NTNU

Norwegian University of
Science and Technology

# Attacking B-SIDH Using Castryck-Decru's Key Recovery Attack on SIDH

**Skuggedal, Georg**

Submission date: June 2023
Main supervisor: Professor, Boyd Colin, NTNU
Co-supervisor: Eriksen, Jonathan Komada, NTNU

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

| | |
|---|---|
| **Title:** | Attacking B-SIDH Using Castryck-Decru's Key Recovery Attack on SIDH |
| **Student:** | Skuggedal, Georg |

**Problem description:**

Isogeny-based cryptography is an emerging and exciting field in modern cryptography that strives to create new and hard cryptographic problems that makes it infeasible for both classical and quantum computers to find our secret keys. This field uses elliptic curve theory, particularly special maps between elliptic curves called isogenies. One notable isogeny-based key encapsulation mechanism is Supersingular Isogeny Key Encapsulation (SIKE), which reached the fourth round of the NIST standardization process. However, in the summer of 2022 Castruck and Decru presented a new devastating key-recovery attack on SIDH, which also breaks SIKE in seconds. The full extent of the affected protocols remains uncertain. One of these protocols, that remains unexplored, is Costello's B-SIDH, which introduces a new way of instantiating isogeny-based cryptography.

The objective of this thesis is to evaluate to what extent does Castruck and Decryu's attack hold in the case of B-SIDH and to determine the extent to which it is effective. In addition, we aim to assess the effectiveness of Castruck and Decryu's attack in retrieving Bob's secret key in B-SIDH for various key sizes. To accomplish this, the thesis will adapt the available implementation of the attack, and make appropriate adjustments to ensure its applicability against B-SIDH.

| | |
|---|---|
| **Approved on:** | 2023-02-24 |
| **Main supervisor:** | Professor, Boyd Colin, NTNU |
| **Co-supervisor:** | Eriksen, Jonathan Komada, NTNU |

# Abstract

The foundation of our public-key cryptography schemes lies in the hardness of specific mathematical problems. Although no efficient solution for these problems exists for classical computers, the advent of quantum computing presents a significant threat. Quantum computers have the potential to solve these mathematical problems, threatening the security of our current public-key cryptography systems. As a consequence, the cryptographic community is actively exploring alternative cryptographic protocols, which rely on other hard mathematical problems robust against both classical and quantum attacks. Two such schemes are the isogeny-based key exchanges called Supersingular Isogeny Diffie-Hellman (SIDH) and the closely related B-SIDH. These two are based on hard problems from the theory of elliptic curves and isogenies, which are rational maps between elliptic curves.

In the summer of 2022 Castryck and Decru presented a new devastating key-recovery attack on SIDH, breaking it in seconds. The attack recovers Bob's secret key by computing isogenies in a higher dimension. In this thesis, we examine to what extent Castryck and Decru's attack holds in the case of B-SIDH and explore the applicability of the attack. Additionally, we modify the practical implementation of the attack to take into account Bob's varying torsion and create a new representation of Bob's secret key using the Chinese Remainder Theorem (CRT).

Our findings indicate that the attack can be effectively applied to B-SIDH under certain conditions, especially when $p+1$ or $p-1$ can be factored into only prime powers of two. When Alice's degree is not a prime power of two, the attack requires computing $(\ell,\ell)$-isogenies for $\ell > 2$, which poses practical challenges. The attack's efficiency on B-SIDH is innately slower due to the necessity of computing isogenies higher degree.

# Sammendrag

Fundamentet i nesten alle offentlige nøkkelkryptografier ligger i vanskeligheten til spesifikke matematiske problemer. Selv om det ikke eksisterer effektive løsninger for disse problemene på klassiske datamaskiner, presenterer kommende kvantemaskiner en betydelig trussel. Kvantemaskiner har potensiale til å løse disse vanskelige problemene, og truer sikkerheten til nåværende offentlige nøkkelkryptografi systemer. Som en konsekvens utforsker det kryptografiske miljøet aktivt etter alternative kryptografiske protokoller, som er avhengige av andre vanskelige matematiske problemer som er robuste mot både klassiske- og kvanteangrep. To slike systemer er isegoni baserte nøkkelutvekslinger, Supersingulær Isogeni Diffie-Hellman (SIDH) og den nært relaterte B-SIDH. Disse to er basert på vanskelige problemer fra teorien om elliptiske kurver og isogenier, som er rasjonelle avbildninger mellom elliptiske kurver.

Sommeren 2022 presenterte Castryck og Decru et nytt ødeleggende nøkkelgjenopprettingsangrep på SIDH, og brøt det på sekunder. Angrepet gjenoppretter Bob sin hemmelige nøkkel ved å beregne isogenier i en høyere dimensjon. I denne oppgaven undersøker vi i hvilken grad Castryck og Decru sitt angrep holder i tilfellet med B-SIDH og utforsker anvendelsen av angrepet. I tillegg endrer vi den praktiske implementasjonen av angrepet for å ta hensyn til Bob sin varierende torsjon og skaper en ny representasjon av Bob sin hemmelige nøkkel ved å bruke det kinesiske restteorem.

Våre funn indikerer at angrepet effektivt kan brukes på B-SIDH under visse omstendigheter. Spesielt når $p+1$ eller $p-1$ kan bli faktorisert til bare primpotenser av to. Når Alice sin grad ikke er en toerpotens, krever angrepet beregningen av $(\ell, \ell)$-isogenier for $\ell > 2$, som forårsaker praktiske utfordringer. Angrepet sin effektivitet på B-SIDH er naturlig tregere på grunn av nødvendigheten av å beregne isogenier av høyere grad.

# Preface

This Master's thesis represents the culmination of my 5 year Master of Science degree in Communications Technology at the Department of Information Security and Communication Technology at the Norwegian University of Science and Technology (NTNU). It stands as a testament to the journey I have undertaken over the past 5 years and reflects my hard work and dedication.

The work presented here reflects my personal endeavor, but it was far from a solitary pursuit. I would like to express my gratitude to my amazing supervisors, Colin Boyd and Jonathan Komada Eriksen, whose guidance and insights have been invaluable. Thank you for lending me your expertise and for answering every dumb question I have had. Working with you has been a real pleasure and good luck in the future.

I would like to thank my family for their unconditional support and belief in my capabilities. Also, my deepest appreciation goes to my wife. Thank you for your patience, support, and love over these years. I can't wait until what the future has in store for us.

Finally, to all my peers, I am humbled by the opportunity to get to know all of you. You are all amazing people and made my stay here in Trondheim for the better. Thank you for all these years and I hope our paths cross again in the future.

*Georg Skuggedal*
*Trondheim 2023*

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **CDH** | Computational Diffie-Hellman. |
| **CIP** | Computational Isogeny Problem. |
| **DDH** | Decision Diffie-Hellman. |
| **DES** | Data Encryption Standard. |
| **DHP** | Diffie-Hellman Problem. |
| **DIP** | Decisional Isogeny Problem. |
| **DLP** | Discrete Logarithm Problem. |
| **NIST** | National Institute of Standards and Technology. |
| **OTP** | One-Time Pad. |
| **SIDH** | Supersingular Isogeny Diffie-Hellman. |
| **SIKE** | Supersingular Isogeny Key Encapsulation. |

# Chapter 1 Introduction

Elliptic curves have long been stars of the cryptographic stage and formed the backbone of many widely used key exchange protocols and digital signature schemes. Since the proposal of the Supersingular Isogeny Diffie-Hellman (SIDH) scheme by Jao and De Feo in 2011 [JF11], the world of isogeny-based cryptography has been attracting substantial attention. This scheme was created to stand resilient as we march into the era of quantum computing. While there are no known efficient quantum algorithms for attacking isogeny-based schemes, in the pursuit of quantum resilience we cannot forget the potential threats from classical computers.

## 1.1 Motivation

Cryptography plays a crucial role in safeguarding our digital information as it travels between devices and across the internet. Even before the birth of the internet, encryption standards like the Data Encryption Standard (DES) were established to scramble data during transit to ensure its confidentiality. However, as technology advances, new encryption standards must be developed to keep pace with the increasing sophistication of potential attacks.

The foundation of encryption schemes lies in the difficulty of specific mathematical problems. Although cryptographic constructions can rarely be proven unconditionally secure, we often rely on the assumption that solving these problems would require an infeasible amount of time. For example, while multiplying two numbers is a simple task, factoring a large number into its prime components presents a much greater challenge. This problem, known as integer factorization, is the basis for the widely used RSA encryption scheme.

The One-Time Pad (OTP) offers provably perfect security but is practically infeasible for widespread use due to its requirements for long, pre-shared keys. In 1976, Diffie and Hellman [DH76] introduced a solution that eliminated the need for pre-arranged keys, giving birth to public key cryptography. The Diffie-Hellman key exchange remains an essential component of many modern internet services. The Diffie-Hellman Problem (DHP) relies on the difficulty of solving the Discrete Logarithm Problem (DLP). Although no efficient algorithm for classical computers exists to solve the DLP, Shor [Sho97] developed an algorithm in 1994 that can solve both DLP and integer factorization using quantum computers.

Although no quantum computer is currently capable of executing Shor's algorithm at scale, the race is on to find new cryptographic problems that cannot be efficiently solved by quantum computers. Post-quantum cryptography research focuses on several approaches, one of which is isogeny-based cryptography. This field aims to develop protocols based on the properties of supersingular elliptic curves and supersingular isogeny graphs. One such protocol, the Supersingular Isogeny Key Encapsulation (SIKE), reached the fourth round of the National Institute of Standards and Technology (NIST) standardization process with high hopes of being standardized. However, in July 2022, Castryck and Decru unveiled a brutal key-recovery attack that shattered the security of SIKE, breaking it in seconds.

The full extent of the damage caused to isogeny-based cryptographic systems by this attack remains uncertain, highlighting the importance of continued research in this area. In this thesis, we will investigate the implications of the attack on another closely related key exchange protocol, B-SIDH, to better understand its impact on the broader field of cryptography.

## 1.2 Research Objective and Questions

In 2019, Costello [Cos20] explore a new way of instantiating isogeny-based cryptography named B-SIDH. In light of Castryck and Decru's new devastating key-recovery attack on SIDH, a new question can be asked. To what extent does Castryck and Decru's attack hold in the case of B-SIDH[1]? Can the attack, with the same efficiency, be used to recover Bob's secret key in B-SIDH? This problem is meant to be answered by using an implementation of the proposed attack [CD23a] as well as any modifications needed to succeed. To achieve this research objective, we aim to answer the following three research questions.

*Research question 1: What properties of SIDH are used in the Castryck and Decrus attack?*

*Research question 2: Can Castryck and Decru's attack be used with the same efficiency on B-SIDH as on SIDH?*

[1]Pronounced 'B-SIDE'

*Research question 3: What modification needs to be done to Castryck and Decru's attack in order the use the same attack on B-SIDH?*

To answer research question 1, we will undertake a comprehensive analysis surrounding the theory underlying SIDH and the specific characteristics of the attack. Our initial focus on SIDH will be a theoretical deep dive into SIDH's framework and provide examples to gain a solid understanding of SIDH's properties.

Subsequently, we turn our attention to the attack in question. An in-depth study of the attack will offer us insight into methodology, effectiveness, and particularly, its interaction with SIDH. Through this dual perspective, we aim to identify the properties of SIDH that the attack utilizes.

The second research question 2, involves the creation of a B-SIDH implementation. Our first point of action will be an explanation of the key exchange where we highlight the differences to SIDH. The subsequent phase is focused on adapting Castryck and Decru's attack to operate effectively on B-SIDH. Throughout this process, we will describe what modifications, from a practical and theoretical point of view, are needed to answer research question 3.

It's worth noting that the concept of efficiency here extends beyond the speed of the attack on B-SIDH, but also includes the scope of the attack's applicability. By practically testing the attack we wish to evaluate its reach and feasibility across varying degrees of complexity.

## 1.3 Outline of Thesis

Including this introduction, this thesis is sectioned into 6 chapters, and an appendix.

**Chapter 2** equips the reader with the essential background knowledge required to comprehend Chapter 3 and the remainder of the thesis. The chapter is systematically structured, beginning with the foundational concepts of algebra and transitioning into an introduction to modern cryptography. In the latter half of the chapter, elliptic curves are introduced, followed by an introduction to isogenies, which concludes the chapter.

**Chapter 3** presents the two key exchange protocols SIDH and B-SIDH on both a high-level and a detailed explanation. The chapter begins by highlighting some key properties before delving into the protocol descriptions and provides some toy examples.

**Chapter 4** presents the key recovery attack developed by Castryck and Decru. The chapter goes more in-depth on certain curves, including relevant theories for the attack. A relevant attacking example is also presented to enhance understanding.

**Chapter 5** discusses what changes were made to the attack B-SIDH and presents the findings during our research.

**Chapter 6** concludes with the key findings derived from the research, while also presenting potential avenues for future work related to the topics explored in this thesis.

**Appendix** provides additional material used in this thesis.

# Chapter 2
# Background

This chapter contains the essential background knowledge required to comprehend Chapter 3 and the remainder of the thesis. In section 2.1 we explore the fundamentals of algebra transitioning into general cryptography concepts in section 2.2. Finally, in section 2.3 and 2.4, we cover the basics of elliptic curves and isogenies preparing for chapter 3.

## 2.1 Algebra Fundamentals

In this section, a quick overview will be given of necessary theorems and definitions from algebra, which are needed to understand later subjects. The proofs of theorems are omitted in this section and we refer to any introduction book to the subject [JBN14]. Before diving into groups, fields, and rings, it's important to establish a foundation in some fundamental concepts of set theory.

– Sets: A set is a collection of distinct objects, considered as an object in its own right. Sets are usually denoted by capital letters and their elements by lowercase letters. The objects in a set are called its elements or members. A set can be finite or infinite depending on the number of its elements.

– Binary operations: A binary operation is a rule that combines two elements from a set to produce a third element within the same set. Common examples of binary operations include addition, subtraction, multiplication, and division in the set of real numbers.

– Mappings (functions): A function is a relation between two sets that associates each element of the first set with exactly one element of the second set. Functions are used to describe relationships between different mathematical objects and are fundamental to the study of algebraic structures.

– Relations: A relation is a set of ordered pairs of elements from two sets. It is a way to describe a specific connection between elements in different sets. An important type of relation in algebra is the equivalence relation, which satisfies three properties: reflexivity, symmetry, and transitivity.

While in arithmetic we have four binary operations available to us, in abstract algebra we often only consider addition and multiplication. Why we do this becomes clearer with an example. $3 - 4$ is the same as $3 + (-4)$. Here one can think of the subtraction operations as adding a negative. Also $3 - (-4)$ is the same as $3 + 4$. In this example subtracting means the same as adding the opposite. But we use the word *inverse* instead of opposite. These examples show that subtraction means the same as addition with inverses. The same argument can be made that division is the same as multiplication with inverses.

One reason we want to do this is that in algebra, the elements we work with may not always be numbers. Some sets contain elements where each element is a permutation, where subtraction and division may be incoherent. It makes more sense to combine one element with an inverse since using inverses is more general.

With these basic concepts in mind, we can now introduce groups, rings, and fields as specific algebraic structures.

### 2.1.1 Groups

The first tool we will explore is the structure called a group. Let's take a look at the definition.

**Definition 2.1.** **A group** can is a non-empty set of elements $G$ and a binary operation on $G$. The binary operation is depicted by a mapping $* : G \times G \to G$. The set and binary operation needs to satisfy the following axioms:

(i) **Associative** For all $a, b, c \in G$ we have $(a * b) * c = a * (b * c)$

(ii) **Identity element** There exists an element $e \in G$ such that $e * a = a$ for all $a \in G$,

(iii) **Inverse element** For every $a \in G$, there exists an inverse $a' \in G$ such that $a * a' = e$

We write a group as a tuple with the symbol $G$ and the binary operations. For example, consider the set of real numbers $\mathbb{R}$, which has the binary operations addition $a + b$ and multiplication $a * b$. We may write $(\mathbb{R} \backslash \{0\}, *)$.

In many instances, we may omit writing the multiplication sign, and instead only write $ab$.

**Definition 2.2. An abelian group** also known as a commutative group, is a group $(G, *)$ that satisfies an additional axiom, called the commutative axiom.

(iv) **Commutative** For all $a, b \in G$ we have $ab = ba$

In other words, the order in which the elements are combined does not affect the result of the operation.

Examples of abelian groups include the additive group of integers $(\mathbb{Z}, +)$, the additive group of real numbers $(\mathbb{R}, +)$, and the multiplicative group of nonzero real numbers $(\mathbb{R} \backslash \{0\}, *)$.

We continue our exploration of groups by talking about subgroups. Essentially, a subgroup is a group that is contained within another group. More formally:

**Definition 2.3. Subgroup** Let $(G, *)$ be a group, then a non-empty subset $H \subset G$ is called a subgroup of G if $(H, *)$ is a group, written as $H < G$.

A subgroup is a subset of a bigger group that shares the same binary operation as the original group. An important thing to remember about subgroups is that they are themselves, groups. So all the properties of groups (associativity, existence of identity element, and inverses) apply to them.

In algebra, we often work with multiple groups and more specifically the relation between two groups. One such mapping or a function between two groups is called *group homomorphism.*

**Definition 2.4. Group homomorphism** Given two groups, $(G, *)$ and $(H, *)$, a mapping $\phi : G \to H$ is a group homomorphism such that for all $a, b \in G$ it holds that $\phi(a * b) = \phi(a) * \phi(b)$

The binary operation $\phi(a * b)$ is the operation in $G$, while the binary operation between $\phi(a) * \phi(b)$ is the operation in $H$.

In other words, a group homomorphism maps elements of one group to elements of another group in such a way that the result of the group operation in the first group corresponds to the result of the group operation in the second group, after applying the homomorphism.

Given a group homomorphism, there is a subgroup we are specifically interested in called the *kernel.* The kernel is the set of elements that are mapped to the identity element of the targeted group.

**Definition 2.5. Kernel** Let $\phi : G \to H$ be a group homomorphism between the two groups G and H. The kernel of $\phi$ is the set $\ker \phi = \{a \in G | \phi(a) = 1_H\}$, where $1_H$ is the identiy of H.

The image of a group homomorphism is the set of all elements in the codomain (target group) that are mapped to by the homomorphism function from the elements of the domain (source group). We define:

**Definition 2.6. Image** Let $G$ and $H$ be two groups, and $\phi : G \to H$ be a group homomorphism. The image of the group homomorphism $\phi$, denoted by $\operatorname{im} \phi$, or simply $\phi(G)$, is the subset of $H$ consisting of all elements $h \in H$ for which there exists an element $g \in G$ such that $\phi(g) = h$.

The image of a group homomorphism $\phi : G \to H$ is always a subgroup $H$. This fact follows from the properties of group homomorphisms, which ensure that the image of $f$ is closed under the group operation in $H$, contains the identity element of $H$, and includes the inverses of its elements.

If the group homomorphism $\phi$ is injective and surjective, then we call $\phi$ a group *isomorphism*. It follows that $\phi$ is an isomorphism if and only if only the identity in $G$ is in the kernel and $\operatorname{im} \phi = H$. In this case, we say that $G$ and $H$ are isomorphic, denoted as $G \cong H$.

If $\phi$ is a homomorphism from $G$ to itself, $H = G$, then $\phi$ is called an endomorphism.

**Definition 2.7. Group order** The order of a group, denoted by $|G|$, refers to the number of elements contained within that group.

A group can have either a finite or an infinite order. A group has a finite order if it has a finite number of elements and the order of a finite group is always a non-negative integer. A group has an infinite order if it has an infinite number of elements. For example, the additive group of integers $(Z, +)$ is an example of an infinite group, as it contains an infinite number of integers.

### 2.1.2 Rings

After the introduction of groups and their properties, we are ready to look at more specific structures; groups with extra features. Additive groups which have a second operation, multiplication. We call these rings.

**Definition 2.8. Ring** is a non-empty set $R$ with two binary operations $+$ (addition), and $\cdot$ (multiplication), and is called a ring if the following axioms hold:

(i) $(R, +)$ is an abelian group.

(ii) **Associative** For all $a, b, c \in G$ we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

(iii) There exist $1 \in R$ such that $1 \cdot a = a$ for all $a \in R$

(iv) **Distributive property** $a \cdot (b + c) = a \cdot b + a \cdot c$, and $(a + b) \cdot c = a \cdot c + b \cdot c$ for all $a, b, c \in R$

For both operations, the set is *closed.* This means if you add any two elements in $R$, you get another element in $R$. Similarly, if you multiply any two elements in $R$, you get a third element in $R$. More formally, if $x, y \in R$, then $x + y \in R$ and $x \cdot y \in R$.

Also note that 1 does not necessarily mean the integer 1, but rather symbolizes the multiplicative identity in $R$. If the operation $\cdot$ is commutative, we way that $R$ is a commutative ring.

**Definition 2.9. Characteristic** Let R be a ring. If there exists $n \in \mathbb{N} \setminus \{0\}$ such that $na = 0$ for all $a \in R$ then the smallest $n$ is called the characteristic of $R$, denoted as $char(R)$. If there is no such $n$, R has characteristic 0.

The characteristic of a ring gives important information about the structure of the ring. This definition also applies to fields which be elaborate on in the following section.

### 2.1.3 Fields

Up until now, we have seen that groups and rings are defined after which operations we are allowed along with other axioms. Loosely speaking if we can add and subtract you have a group. If we can add, subtract and multiply, we have a ring. If we can use all four operations, we have what we call fields.

**Definition 2.10. Field** A set F with the two operations addition and multiplication is a field K if the following three conditions hold

(i) F is an abelian group

(ii) $F \setminus \{0\}$ is an abelian group under multiplication

(iii) The disruptive law holds: $a(b + c) = ab + ac$

There is an infinite number of fields, where the most famous ones are the rational numbers $\mathbb{Q}$, the real numbers $\mathbb{R}$, and the complex numbers $\mathbb{Q}$. These are examples of infinite fields, however, there also exist finite fields or finite prime fields. A finite prime field can be written of the form $\mathbb{Z}/p\mathbb{Z}$ which means integers modulo p, which we will write as $\mathbb{F}_p$.

The finite prime fields and rational numbers $\mathbb{Q}$ is the starting point of all fields. That is if you pick any field $F$, then it will contain one and only one of these fields as a subfield, and we say $F$ is an extension field. The characteristic of a field tells us which prime field it extends.

Since we mostly will be working with finite fields the following theorem will be important to find the characteristic of our field.

**Theorem 2.11.** *Let $F$ be a field, then the characteristic is either $char(F) = p$ where $p$ is a prime number or $char(F) = 0$.*

With groups, we are often interested in subgroups of other groups. However, when it comes to fields, it's often the other way around. We start with a field and add numbers to it, to get larger fields. We then get the definition

**Definition 2.12.** **Extension fields** If $F$ and $E$ are fields with $F \subset E$, we say that E is an **extension** of E

For example $\mathbb{C}$ is an extension of $\mathbb{R}$ which is an extension of $\mathbb{Q}$.

**Definition 2.13.** **Algebraically closed** A field $F$ is algebraically closed if every non-constant polynomial $p(x)$ with coefficients in $F$ has a root in $F$.

An example of an algebraically closed field is the field of complex numbers $\mathbb{C}$ since every non-constant polynomial with complex coefficients has a root in $\mathbb{C}$.

Later in the thesis, we will be interested in quadratic extensions of large prime fields. When $p \equiv 3 (\bmod 4)$ We will represent $\mathbb{F}_{p^2} = \mathbb{F}(i)$ with $i^2 + 1 = 0$ where elements are of the form

$$\mathbb{F}(i) = \{a + bi : a, b \in \mathbb{F}_p\}$$

We can use theorem 2.11 to find the characteristic $char(p^2) = p$ of our field.

**Definition 2.14.** **Algebraic closure** Given a field $F$ its algebraic closure is a larger field that contains $F$ and is algebraically closed.

meaning that every polynomial with coefficients in this larger field has a root in the field. This property allows us to solve polynomial equations without leaving the field. The algebraic closure of a field $F$ is typically denoted by $\bar{F}$

For example, the field of real numbers $\mathbb{R}$ is not algebraically closed, as there are some polynomials with real coefficients that do not have real roots. Consider the equation $x^2 + 1 = 0$. There is no real number $x$ that can satisfy this equation.

However, the field of complex numbers $\mathbb{C}$ is algebraically closed. It contains all the roots of polynomials with real coefficients, including those that are not real. In the case of the equation $x^2 + 1 = 0$, its solutions are $x = i$ and $x = -i$, where $i$ is the imaginary unit, and these solutions belong to the field of complex numbers $\mathbb{C}$.

## 2.2 Cryptography

In this thesis, we will set the following framework for cryptography: Two communicating parties, Alice and Bob, tries to communicate over an insecure channel. Both parties are honest with the intention of sending messages privately. The messages sent are in some way encrypted before being sent into the insecure channel, then decrypted into the original message at the receiving party. A third entity, Eve, acts as an eavesdropper and intercepts every message over the insecure channel.

### 2.2.1 Asymmetric and Symmetric Cryptography

Cryptographic systems can be broadly divided into two kinds. Symmetric cryptography and asymmetric cryptography (also known as public-key cryptography). In symmetric cryptography, the communicating entities, Alice and Bob, share some sort of shared secret to both encrypt and decrypt messages. It is fast and efficient but requires that both parties have a secure method for exchanging the shared secret.

For both of these cryptographic systems, we can define the tuple $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. $\mathcal{K}$ is the key space, i.e. the set of all possible keys. $\mathcal{M}$ is the message space, i.e. the set of all possible messages. $\mathcal{C}$ is the ciphertext-space, i.e. the set of all possible ciphertexts.

A symmetric encryption scheme consist of three algorithms $(\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec)$

- $\mathcal{G}en$ : Key generation algorithm, which returns a random[1] key $k \in \mathcal{K}$.
- $\mathcal{E}nc$ : Encryption algorithm, which takes in a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$ and returns a ciphertext $c \in \mathcal{C}$.
- $\mathcal{D}ec$ : Decryption algorithm, which takes in a key $k \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$ and returns a message $m \in \mathcal{M}$.

[1]With the use of a pseudorandom number generators (PRNGs).

The correctness of this scheme can be defined by $\forall k \in \mathcal{K}, \forall m \in \mathcal{M}$, we have $m = Dec(k, Enc(k, m))$ meaning that encrypting a message $m \in \mathcal{M}$ using a key $k \in \mathcal{K}$ decrypts to itself when using $k$.

Assuming $|\mathcal{K}| \geq |\mathcal{M}|$ one can prove there exists a symmetric encryption scheme perfectly secure [SS14]. However, it is practically infeasible to use key sizes bigger or equal to the size of the message. Therefore in most schemes $|\mathcal{K}| < |\mathcal{M}|$ but $\mathcal{K}$ is still large enough such that trying to decrypt with all possible keys in $\mathcal{K}$ is computationally infeasible.

One major drawback of symmetric encryption is the problem of key distribution. Two entities who have never met cannot share a secret. Public-key encryption aims to solve this, as well as other problems. In contrast to symmetric-key schemes, public-key schemes use two keys, one for encrypting (a public key) and one for decrypting (a private key). The public key can freely be distributed to anyone who wishes to send a message, and the corresponding private key is kept secret.

A public-key encryption scheme consist of three algorithms ($\mathcal{G}en$, $\mathcal{E}nc$, $\mathcal{D}ec$)

- $\mathcal{G}en$ : Key generation algorithm, which returns a public key $pk$ and private key $sk$.
- $\mathcal{E}nc$ : Encryption algorithm, which takes in a public key pk and a message $m \in \mathcal{M}$ and returns a ciphertext $c \in \mathcal{C}$.
- $\mathcal{D}ec$ : Decryption algorithm, which takes in a private key and a ciphertext $c \in \mathcal{C}$ and returns a message $m \in \mathcal{M}$.

The key pair ($pk$, $sk$) have the property that it is computationally infeasible to determine the private key solely from the knowledge of the public key. Further, the correctness can be defined by $\forall (pk, sk) \in \mathcal{K}, \forall m \in \mathcal{M}$ we have that $m = Dec(sk, Enc(pk, m))$.

### 2.2.2 Diffie-Hellman Key Exchange

In 1976, Diffie and Hellman [DH76] presents a key exchange algorithm, where a shared secret between two parties is derived. Note that this is a key exchange algorithm, not a public-key encryption scheme. However, the Diffie-Hellman key exchange can be turned into a public-key encryption scheme called the ElGamal encryption scheme. This thesis will not discuss ElGamal, readers wanting more information is referred to the original paper [Gam85].

We can formulate the Diffie-Hellman key exchange in the following way. Let $G := \langle g \rangle$ be a cyclic group of prime order $q$. Alice and Bob will begin by publicly agreeing to use the base $g$. They will then each choose a secret exponent, $x$ and $y$ such that $x, y \in Z_q$, and compute $A = g^a$ and $B = g^b$. A and B will then be exchanged and both parties will apply their secret exponent by calculating $A^b$ and $B^a$. Alice and Bob will then calculate their shared secret $A^b = (g^a)^b = g^{ab} = (g^b)^a = B^a$.

During the key exchange, attacker Eve gathers all information sent over the open channel and has the following information: $(G, g, q, A, B)$. Eve wishes to derive Alice and Bob's shared secret and if he could find one of the secret exponents it would be trivial to calculate $g^{ab}$. We can formulate the problem Eve faces by showing the following attacking game. A challenger sends $g^x$, where $x \in \mathbb{Z}_q$, to an adversary. If the adversary can find a $x' \in \mathbb{Z}_q$ such that $x' = x$ the adversary wins. This problem is called the Discrete Logarithm Problem (DLP).

### 2.2.3 Shor's Algorithm and Quantum Computers

Quantum computers represent a significant breakthrough in computing technology, as they utilize the principles of quantum mechanics to perform certain calculations that are exponentially faster than normal classical computers. One of the most well-known algorithms that exploit the power of quantum computing is Shor's algorithm, developed by mathematician Shor in 1994 [Sho97]. This algorithm is capable of solving two key problems that underlie the security of many modern cryptographic systems: integer factorization and the discrete logarithm problem.

Shor's algorithm leverages the unique properties of quantum bits (qubits), which, unlike classical bits, can exist in a superposition of states. This enables quantum computers to process information and perform calculations in parallel, leading to significant speedup when solving certain problems [NCG02].

Integer factorization is a crucial problem in number theory and is the foundation for the widely-used RSA cryptosystem. In classical computing, the most efficient algorithms for integer factorization, such as the General Number Field Sieve [LL93], have a sub-exponential running time. This makes it infeasible to factor in large integers, ensuring the security of RSA.

Shor's algorithm, on the other hand, can factor integers in polynomial time on a quantum computer, thus posing a significant threat to the security of RSA. When applied to the discrete logarithm problem, Shor's algorithm can similarly break cryptographic schemes based on this problem, such as the Diffie-Hellman key exchange and the ElGamal cryptosystem.

As of today, no quantum computer is large enough to utilize Shor's algorithm to break any real-world cryptographic system at scale. However, the mere existence of Shor's algorithm and the rapid progress of quantum computing technology, calls for new cryptographic systems which rely on new hard problems resistant to attacks from both classical and quantum computers.

## 2.3 Elliptic Curves

In many of today's cryptographic systems, elliptic curves are frequently used due to their unique properties, such as speed and compactness, which make them highly appealing from an implementation standpoint. Notably, elliptic curves typically permit the use of smaller keys to achieve equivalent security levels compared to other alternatives. This section will explore the well-known fundamental properties of elliptic curves and present relevant examples for later discussions. While proofs will not be provided in this section, readers interested in obtaining further details and proofs are referred to any introduction book on elliptic curves [Sil09].

An elliptic curve is a curve over a field $K$ (denoted by $E/K$) with a distinguished point. This point is usually the point at infinity, which we will represent as $\infty$. Generally, elliptic curves can be described with a homogeneous polynomial in three variables with coefficients in $\bar{K}$, where $\bar{K}$ is the closure of the field we are working over. For a field $K$ where $char(K) \neq 2$ an elliptic curve admits of the form $E : y^2 = f(x)$, with $deg(f) = 3$. This thesis will only consider fields with characteristic not equal to 2 or 3. Consequently, an elliptic curve will be defined as follows:

**Definition 2.15.** An elliptic curve $E$ over a field $K$ is defined by the equation

$$E : y^2 = x^3 + ax + b$$

where $a, b \in K$

In cryptographic applications, the Montgomery curve model is often the preferred choice, due to some special properties that make them particularly efficient to implement on computers. One such property allows Montgomery curves to be expressed using only the $x$-coordinate, enabling *$x$-only arithmetic*, a concept that will be further explored later in this thesis. Montgomery curve is defined as follows

**Definition 2.16.** A Montgomery curve over a field $K$ is defined by the equation

$$E_a : y^2 = x^3 + ax^2 + x$$

where $a \in K$ and $x$, $y$ are symbolic values [Mon87]. From this point, only Montgomery curves will be used and considered.

The genus of a curve is a measure of its *complexity* or *shape*. It is closely related to the number of *holes* the curve visually creates when drawn. By definition, the genus for an elliptic curve is always equal to 1. We will consider genus 2 curves in chapter 4.

Writing down equations for elliptic curves comes with its challenges as the equation is generally not unique. Meaning two curves can be the *same* curve in different coordinated systems, when they are *isomorphic.* In section 2.4, we will delve into a more detailed explanation of isomorphism. A straightforward method to ascertain whether two curves are isomorphic involves computing the *j-invariant.*

**Theorem 2.17.** *Let $E$ and $E'$ be two elliptic curves over a field $K$, then $E$ and $E'$ are isomorphic over $K$ if and only if they have the same j-invariant.*

In another way, comparing the $j$-invariant of two curves is a fast way of checking if all certain criteria for isomorphism are fulfilled. Every elliptic curve has a unique $j$-invariant and for curves in Montgomery form, the $j$-invariant can be calculated as

$$j(E_a) = \frac{256(a^2-3)^3}{(a^2-4)} \tag{2.1}$$

We will emphasize the importance of this property by providing an example, as it becomes relevant in later topics. Given two Montgomery curves $E_{a_1}$ and $E_{a_2}$ over $\mathbb{F}_{71}$ where $a_1 = 1$ and $a_2 = 70$ we have

$$j(E_{a1}) = \frac{256(1^2-3)^3}{(1^2-4)} = 20 = \frac{256(70^2-3)^3}{(70^2-4)} = j(E_{a2})$$

Theorem 2.17 tells us the two curves $E_{a_1}$ and $E_{a_2}$ are isomorphic due to having the same j-invariant.

Given an elliptic curve $E/K$, the set of points carries an abelian group structure, which makes it interesting in cryptography. Start with two points $P$ and $Q$ on the curve $E$, then we can uniquely describe a third point $R$ by the operation of $P + Q$. $P + Q$ can intuitively be described geometrically by drawing a straight line between $P$ and $Q$, as depicted in Figure 2.1. Any straight line will intersect the curve in exactly three (not necessarily distinct) points, by Bezout's theorem [Mil06], where the three points are $P$, $Q$, and $-R$. The sum of two points on the curve lying in a straight vertical line equals the point at infinity, which serves as the identity. Since the curve is symmetrical about the x-axis, any point $P = (P_x, P_y)$ has a point opposite. This operation is called point negation and can be defined with $-P = (P_x, -P_y)$, where we flip the $y$-coordinate.

By the description given, the binary operation $+$ turns $(E, +)$ into an abelian group with $\infty$ as the identity since the following properties are fulfilled.

(i) For all $P, Q \in E$ we have $P + Q = Q + P$

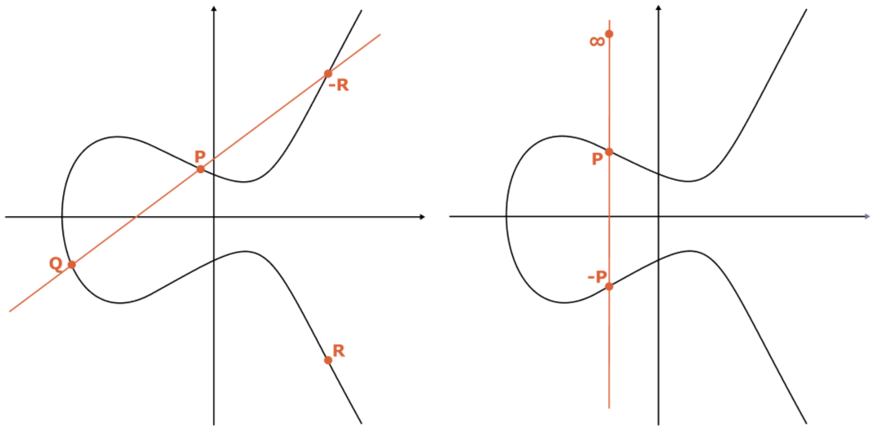


**Figure 2.1:** Left figure illustrates the operation of adding two points $P$ and $Q$ together which represents a third point $R$. The right figure illustrates adding two points lying in a vertical line which equals the point at infinity.

(ii) For all $P \in E$, $P + \infty = P$

(iii) For all $P \in E$, there exist $-P \in E$ such that $P + (-P) = \infty$

(iv) For all $P, Q, R \in E$, $(P + Q) + R = P + (Q + R)$

One operation we are specifically interested in is the operation of adding a point to itself multiple times, called scalar multiplication.

**Definition 2.18. Scalar Multiplication** Given a curve, E, over a finite field $K$, any integer $n$ defines a group homomorphism $\phi : E \to E$ from any elliptic curve E to itself. It consists of summing up $n$ copies of a point using $(E, +)$. Denote as

$$[n]P = \underbrace{P + P + \cdots + P}_{n \text{ times}} \tag{2.2}$$

for some scalar $n$ and a point $P = (P_x, P_y) \in E$ .

This group homomorphism $\phi : E \to E$ is actually an example of an *isogeny*. Later in section 2.4 where we define isogenies we can see from the definiiton that scalar multiplication is an isogeny from the curve to itself. For now the interesting thing to note that is for some number $n$ and point $P$, then $[n]P = \infty$, i.e. the identity element of $E$.

In this thesis, we are exclusively interested in elliptic curves over finite fields, i.e. $E/\mathbb{F}_q$. We are interested in the subset of $\mathbb{F}$-rational points which forms a subgroup of E.

**Definition 2.19. K-rational points** Given an elliptic curve $E$ defined over $K$, the $K$-rational points are the points $(x, y) \in E$ with $x, y \in K$ and the point $\infty$. We denote as $E(K)$.

As $\mathbb{F}_q$ is a finite field, it follows that $E(\mathbb{F}_q)$ contains a finite set of points. Finding how many points there are contained in $E(\mathbb{F}_q)$, which we denote as $\#E(\mathbb{F}_q)$, is not a trivial task, the best-known algorithm is polynomial-time algorithms that find the number of points [LM95]. However, there is a theorem called Hasse's theorem which gives us an upper and lower bound of $E(\mathbb{F}_q)$.

**Theorem 2.20. Hasse's theorem** *Let E be an elliptic curve over finite field $\mathbb{F}_q$ then the order of $E(\mathbb{F})$, denoted by $\#E(\mathbb{F})$ is given by*

$$\#E(\mathbb{F}_q) = q + 1 - t \textit{ where } |t| \leq 2\sqrt{q}$$

### 2.3.1 Supersingular Curves

When it comes to isogeny-based cryptography, we are interested in a particular class of elliptic curves, namely *supersingular elliptic curves.* Supersingular curves have some sought-after different behaviors than their counterpart *ordinary curves.* These curves can be defined in various ways, however, we define them in the following way

**Definition 2.21.** Given an elliptic curve E defined over a finite field of $\mathbb{F}_q$ of characteristic $p$, E is supersingular if and only if $p$ divides $\#E(\mathbb{F}_q) - q - 1$

By applying Hasse's theorem 2.20 the order of $E(\mathbb{F}_p)$, the number of points, simplifies to $\#E(\mathbb{F}_p) = p + 1$ where $q = p$ and $p > 5$. This allows us to effectively choose what the group order of supersingular curves will be and makes it easy to calculate the number of points on a given supersingular curve. This also applies to supersingular curves over extension fields.

Just as ordinary curves are isomorphic when they have the same j-invariant, supersingular curves are also isomorphic when they have the same j-invariant. However, we refer to them as supersingular j-invariants The number of different supersingular j-invariants in a field $\mathbb{F}_p$ can be found using the following theorem:

**Theorem 2.22.** *Let $n$ be the number of distinct supersingular j-invariants in $\mathbb{F}_p$ where $p \neq 2, 3$, then*

$$n = \left\lfloor \frac{p}{12} \right\rfloor + \begin{cases} 0, & \textit{if } p \equiv 1 \ (\textit{mod } 12) \\ 1, & \textit{if } p \equiv 5 \ (\textit{mod } 12) \\ 1, & \textit{if } p \equiv 7 \ (\textit{mod } 12) \\ 2, & \textit{if } p \equiv 11 \ (\textit{mod } 12) \end{cases}$$

*[Sil09, Theorem V.4.1(c)]*

This is a useful property which we will later see in section 3.5.1 where the shared secret in SIDH is one of the j-invariants. Therefore the previous theorem can be used to calculate the upper bound of the shared key space.

### 2.3.2 Quadratic Twists

An elliptic curve $E$ over a not algebraically closed field $K$ has an associated *quadratic twist*. We will explore this by giving a concrete example. Let $E$ be an elliptic curve over finite field $K$ and of the Montgomery form in 2.16, then given a $d \neq 0$ not a square in $K$, a quadratic twist of E is the curve $E^t$, defined by the equation

$$E^t : dy^2 = x^3 + Ax^2 + x$$

The two elliptic curves $E$ and $E^t$ are not isomorphic over $K$. This may seem to contradict theorem 2.17 that two curves sharing the same j-invariant over $K$ are isomorphic. The two curves are rather isomorphic over the field extension $\bar{K}$, however, they will still have the same j-invariant over $\mathbb{F}_{p^2}$.

Now consider the finite field of $\mathbb{F}_{p^2}$ and let $B$ be a square in $\mathbb{F}_{p^2}$ and let $\gamma$ be a non-square in $\mathbb{F}_{p^2}$. Take $\mathbb{F}_{p^4} = \mathbb{F}_{p^2}(\delta)$ with $\sigma^2 = \gamma$ and take the two elliptic curves

$$E_{A,B} : By^2 = x^3 + Ax^2 + x \qquad \text{and} \qquad E^t_{A,\gamma B} : \gamma By^2 = x^3 + Ax^2 + x$$

as models for $E/\mathbb{F}_{p^2}$ and its quadratic twist $E^t/\mathbb{F}_{p^2}$. These two curves are isomorphic over $\mathbb{F}_{p^4}$ where we can write a map

$$\sigma : E_{A,\gamma B}(\mathbb{F}_{p^4}) \to E_{A,B}(\mathbb{F}_{p^4}), \qquad (x, y) \to (x, \delta y) \tag{2.3}$$

which is an isomorphism that leaves the $x$-coordinate unchanged.

Then, there is a point-value correspondence for the $x$-coordinates, given any $u \in \mathbb{F}_{p^2}$, we have three cases:

(i) If $f(u)$ (where $E : y^2 = f(x)$) is a square in $\mathbb{F}_{p^2}$, it corresponds to two points in $E$
(ii) If it's a non-square, it corresponds to two points in $E^t$.
(iii) If $f(u) = 0$, is a point on both curves.

Consider the two points $P_1 = (u_1, -)$ and $P_2 = (u_2, -)$, corresponding to cases $(i)$ and $(ii)$ respectively. From these points, we can construct two isogenies: $\phi_1 : E_{A,B} \to E_{A,B}/\langle P_1 \rangle$ and $\phi_2 : E_{A,\gamma B} \to E_{A,\gamma B}/\langle P_2 \rangle$. However, the evaluation of $\phi_1$ at $P_2$ presents a problem, as these points reside on entirely different curves that are not $\mathbb{F}_{p^2}$-isogenous.

To address this, we elevate to $\mathbb{F}_{p^4}$ and precompose using twisting isomorphism, which results in two new isogenies that are defined over $\mathbb{F}_{p^2}$. We then compose $\phi_1' = (\phi_1 \circ \sigma)$ and $\phi_2' = (\phi_2 \circ \sigma^{-1})$ to yield the isogenies $\phi_1' : E_{A,\gamma B} \to E_{A,B}/\langle \sigma(P_2) \rangle$ and $\phi_2' : E_{A,B} \to E_{A,\gamma B}/\langle \sigma^{-1}(P_1) \rangle$.

Recall Montgomery curves and they allow to be expressed using only the $x$-coordinate, enabling $x$-only arithmetic. Turning our attention back to equation 2.3, the two morphisms $\sigma : (x, -) \to (x, -)$ and $\sigma^{-1} : (x, -) \to (x, -)$, the $x$-coordinate is left unchanged, and we can ignore twisting morphisms completely. So by ignoring the $y$-coordinate, all arithmetic operations can be conducted over $\mathbb{F}_{p^2}$ when we confine ourselves to using only $x$-coordinates.

## 2.4 Isogenies of Elliptic Curves

In this section, we will introduce some special mappings between elliptic curves called isogenies. Isogenies can be looked at as well-behaved maps and have some interesting properties we can use to create key exchanges. We will begin by giving some definitions and theorems before giving some related examples and theories for later.

**Definition 2.23. Isogeny** An isogeny is a map between elliptic curves given by rational functions over a field $K$, denoted by $\phi : E_1 \to E_2$.

The definition over is only partial, meaning we need a few more definitions and theorems to fully grasp the structure of isogenies. This morphism between two elliptic curves preserves the group structure and it's important to note that the identity element of curve $E_1$ maps to the identity of $E_2$. We can state the following theorem

**Theorem 2.24.** *Let $\phi : E_1 \to E_2$ be an isogeny. Then $\phi$ maps the point at $\infty_1$ in $E_1$ to the point at infinity $\infty_2$ in $E_2$, and satisfies $\phi(P + Q) = \phi(P) + \phi(Q)$.*

Notice how the last part of the theorem satisfies the definition of a group homomorphism. So $\phi$ induces a group homomorphism between our two curves $E_1$ and $E_2$. This induces us to look at their kernel i.e. the set of points that maps to the point at infinity.

**Theorem 2.25.** *Let $\phi : E_1 \to E_2$ be a non zero isogeny, then ker $\phi$ is a finite subgroup of $E_1$*

We have introduced general isogenies, however, in this thesis, we will mostly work with a specific type of isogenies called $\ell$-isogenies. An $\ell$-isogeny is an isogeny whose kernel is a cyclic group of order $\ell$. Or more specifically, we can define $\ell$-isogenies as

**Definition 2.26.** **$\ell$-isogeny** An $\ell$-isogeny is an isogeny $\phi : E \to E' = E/G$, where $G \cong \mathbb{Z}/\ell\mathbb{Z}$

$E/G$ should not be confused with the notation $E/K$ where $K$ is a field, however, this shows the group structure of $E'$. This shows that an elliptic curve $E'$ is the image of $\phi$ derived from the kernel $G$.

Composition of isogenies is the process of applying multiple isogenies sequentially. If we have two isogenies $\phi : E \to E'$ and $\psi : E' \to E''$, then the composition of these two isogenies is a new isogeny $\omega : E \to E''$ that is obtained by applying $\psi$ after $\phi$. An important thing about isogeny composition is that it preserves the structure of the elliptic curve, including the group law.

If $\phi$ is $\ell_1$-isogeny and $\psi$ is $\ell_2$-isogeny then $\omega = \phi \circ \psi$ will be a $N$-isogeny where $N = \ell_1 \cdot \ell_2$. Throughout this thesis, $\ell$-isogenies will be referred to as prime degree isogenies, while $N$-isogenies will be a composition of prime degree isogenies.

We have yet to define the degree of an isogeny. Since we mostly will be working $N$-isogenies, we will informally define the degree of an isogeny equal to the size of its kernel, which is $\ell$.

It can be shown that for any $N$-isogeny $\varphi : E \to E'$ of degree $\ell$, has a unique corresponding $\ell$-isogeny called the *dual* isogeny $\hat{\varphi} : E' \to E$. The dual isogeny turns the isogenies between two curves into an equivalence relation we call *isogenous.*

**Definition 2.27.** **Isogenous** Let $E$ and $E'$ be elliptic curves over $K$. If there exists a non-zero isogeny $\phi : E \to E'$, we say that $E$ and $E'$ are *isogenous.*

To the previous definition, we can add that if $E$ is supersingular, it follows that $E'$ is also supersingular. Given the existence of the dual isogeny and this fact, it tells us a lot about the *isogeny class* we are working with. The isogeny class is the set of all elliptic curves that can be reached from a given curve by isogenies.

**Theorem 2.28.** *Let $\mathbb{F}_q$ be a finite field of order $q$. Then all supersingular elliptic curves over $\bar{\mathbb{F}}_q$ are isogenous.*

This theorem tells us that for any field $\mathbb{F}_q$, there exists only one isogeny class containing supersingular curves. i.e. all supersingular elliptic curves belong to the same isogeny class. Further, from the fact that $\phi : E \to E'$ with $E$ as a supersingular curve implies that $E'$ is a supersingular curve, we see that an isomorphism class either contains none or all supersingular curves

In section 2.3 we discussed how we can do group operations on elliptic curves or more specifically adding points. This gives us access to an important subgroup called the *n-torsion subgroup.*

**Definition 2.29. n-torsion subgroup** Let $E$ be an elliptic curve defined over a field $K$, and $n \in \mathbb{N}$. The n-torsion subgroup of $E$ is defined as

$$E[n] = \{P \in E(K) | [n]P = \infty\}$$

$E[n]$ are the points on $E$ of order dividing $n$. The isogeny is called *multiplication-by-n-map* which is an important isogeny we will use a lot later. The multiplication-by-n-map is defined by $\phi : E \to E$, where $\phi(P) = [n]P$ and allows us to easily find isogenies of degree $n$. Notice that $E[n]$ is precisely the kernel of this isogeny.

### 2.4.1 Isogenies From Kernels

So far, we've delved into a multitude of definitions and theorems relating to elliptic curves and isogenies. These concepts, while crucial, may seem quite abstract and disconnected from practical applications, such as their use in a key exchange protocol. In the following section, we take our first stride towards connecting theory with practice, as we delve into how we can generate isogenies from a given kernel.

Lets consider the curve $E : y^2 = x^3 + 12x^2 + x$ over $\mathbb{F}_{17}$ with $j(E) = 10$, which is depicted in Figure 2.2. This curve has 12 rational points, including the point at infinity, listed in Figure 2.2. From the Table, we can easily identify the 2-torsion subgroup consisting of the points $(0, 0), (10, 0)$ and $(12, 0)$. In fact this can be generelized to $(0, 0), (1/\alpha, 0)$ and $(\alpha, 0)$ and holds true for any Montgomery curve.

Take the point $P = (12, 0)$ which we have an order of 2. By definition, we know that multiplying it by 2 or adding the point to itself, yields the point at infinity. If

| point | order |
|---|---|
| $(0,0)$ | 2 |
| $(3,6)$ | 6 |
| $(3,11)$ | 6 |
| $(6,5)$ | 6 |
| $(6,12)$ | 6 |
| $(8,8)$ | 3 |
| $(8,9)$ | 3 |
| $(10,0)$ | 2 |
| $(12,0)$ | 2 |
| $(15,2)$ | 6 |
| $(15,15)$ | 6 |

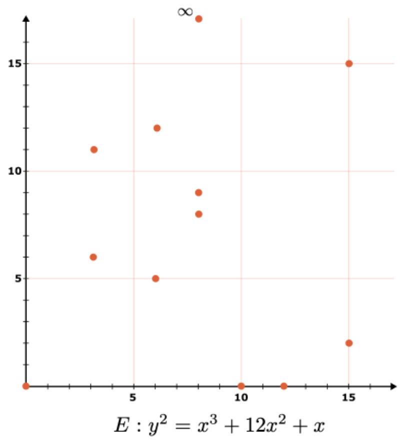


**Figure 2.2:** The table shows the coordinates and order of all rational points on curve E. The figure shows the curve $E : y^2 = x^3 + 12x^2 + x$ over $\mathbb{F}_{17}$ with $j(E) = 10$. The point at infinity is not geometrically correct but rather a representation.

we're to add the point three times to itself, we will come back to the same point, $[3]P = P$, represented in Figure 2.3. We have discovered one of the cyclic subgroups of order 2 on our curve namely $G = \{\infty, (12, 0)\}$. Taking the two other points of order 2 yields the other two cyclic subgroups.

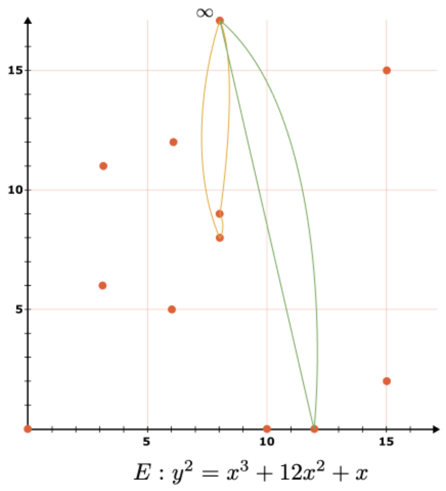


**Figure 2.3:** Visual representation of two cyclic subgroups. The green lines is one of the subgroups of order 2 and the yellow lines are one of the subgroups of order 3.

We can do this with the 3-torsion subgroup as well and find that there are four cyclic subgroups of order 3. In fact, this pattern holds true for any $\ell$ where $p \nmid \ell$. The set of points in the $\ell$-torsion, i.e. the set of points sent to $\infty$ under the multiplication-by-$\ell$ map is such that

$$E[\ell] \cong \mathbb{Z}_\ell \times \mathbb{Z}_\ell$$

forming $\ell + 1$ cyclic subgroups of order $\ell$ when $\ell$ is prime.

A key fact about isogenies is that there is a one-to-one correspondence between $N$-isogenies and finite subgroups of order $N$. Meaning one of the cyclic subgroups $G$ gives rise to a unique isogeny $\phi : E \to E' = E/G$. Vélus formulas [Vél71], a set of formulas, give a way to compute the isogeny from a given kernel. We will not write down Vélu's formulas, the most important thing to note is that these formulas are rational functions, and the degree of the functions is the size of the input.

Let's take the cyclic subgroup we discovered $G = \{\infty, (12, 0)\}$ and input it into Vélus formulas. The formulas give us the image curve $E' : y^2 = x^3 + 12x^2 + 11$ with $j(E') = 6$ and rational function:

$$\phi : x \to \frac{x^2 + 5x + 7}{x + 5}$$

that will take any x-coordinate on $E_a$, not in $\{\infty, (12, 0)\}$, to the corresponding x-coordinate on the image curve $E_{a'}$, as shown in Figure 2.4.

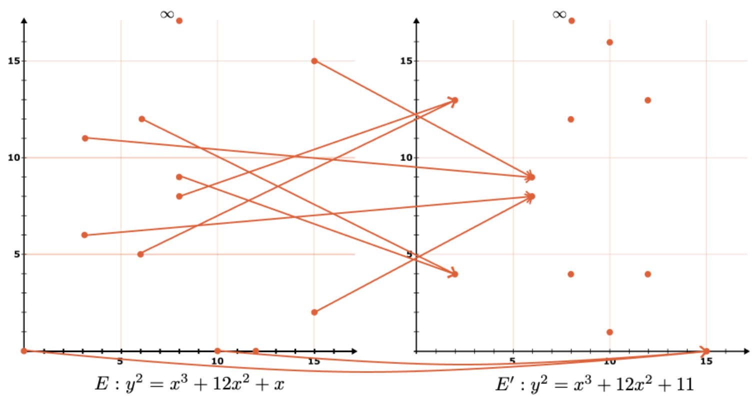


**Figure 2.4:** Isogeny map between two elliptic curves. Taking a point on E, not in the kernel, maps to another point on the image curve.

Given the definition of the kernel, definition 2.5, if we apply the rational function $\phi$ to the points in the kernel, the resulting value should be the identity element of the image, in this case, $E'$. Inserting the two points through the map, indeed yields the identity element of E', namely $\infty$ This behavior aligns perfectly with the definition of a group homomorphism and its kernel, as the kernel forms the set of all elements in the original group that get mapped to the identity in the image.

The key takeaway is that the kernel of an isogeny can uniquely define an isogeny, which we later in chapter 3 will see is an important object for isogeny-based cryptography,

### 2.4.2 The Supersingular Isogeny Problem

Recall to section 2.2.2 about the Diffie-Hellman key exchange where we could formulate a hard problem called the discrete logarithm problem (DLP). We could formulate a similar hard problem called the supersingular isogeny problem: Given two supersingular elliptic curves $E$ and $E'$ defined over a finite field $K$, find any isogeny between them. This problem is conjecturally hard. We can give the following attack game. A challenger sends two supersingular elliptic curves, $E$ and $E'$, to an adversary. If the adversary can find an isogeny $\phi : E \rightarrow E'$, the adversary wins.

# Chapter 3
# SIDH & B-SIDH

In this chapter, we will introduce the Supersingular Isogeny Diffie-Hellman (SIDH) key exchange protocol. We will provide both a high-level overview and a detailed explanation of the protocol by examining a specific example. Additionally, we will discuss the closely related key exchange protocol B-SIDH, which is the primary focus of this thesis.

Throughout sections 3.3 and 3.4, we will use the prime number $p = 2^4 3^3 = 431$ for our examples. The rationale behind selecting primes of this form and the method of choosing it will be explained in section 3.4. In section 3.7 a detailed explanation of the protocol will be given accompanied by an example.

## 3.1 Building Upon Costello

The content of this chapter is largely based on Costello's turotal on SIDH [Cos19] and B-SIDH [Cos20]. Anyone familiar with his papers may notice parallels in the illustrations and examples used, some of which are identical. This is done deliberately to amplify his introduction to SIDH while offering a different point of view on certain properties. In fact, reading this chapter alongside his paper may enhance one's understanding of the topic.

While this chapter might echo Costello's work, our hope is that it gives different insights on certain properties and the focus we provide, especially regarding B-SIDH and the attack by Castryck and Decru. Our perspective aims to shine a different light on the subject, thereby adding value to the existing body of work. We delve deeper into certain aspects, highlighting their relevance to B-SIDH and preparing for understanding Castryck and Decru's attack.

## 3.2 A Brief History of Isogeny-Based Cryptography

In 1976, the original Diffie-Hellman protocol emerged and revolutionized the field of cryptography. This protocol, which is explained in more detail in section 2.2.2, marked the beginning of an era where secret information could be securely shared over public networks.

Fast-forward to 2010, when Stolbunov proposed a variant of the Diffie-Hellman scheme [Sto10], based on the hardness of computing isogenies between ordinary elliptic curves. This was crafted with the aim of being resistant to both classical and quantum attacks. However, Childs and his team [CJS14] discovered a faster method to break Stolbunov's system, indicating that it was not as robust as initially expected.

In 2011, Jao and De Feo stepped into the picture and took Stolbunov's concept to a new level [JF11]. They introduced the Supersingular Isogeny Diffie-Hellman (SIDH) scheme, where they used supersingular instead of ordinary elliptic curves. This switch, however, brought in some challenges due to the noncommutative nature of endomorphisms in supersingular elliptic curves. To work around this, they introduced auxiliary points into the protocol, a novel method that had not been used before.

A new development emerged in 2018, when CSIDH was introduced by a team of researchers, including Castryck, Lange, Martindale, Panny, and Renes [CLM+18]. CSIDH differed from SIDH in its group action, opting for a commutative action instead of a non-commutative one. CSIDH is a promising alternative to SIDH as it has remained unaffected by Castryck and Decru's attack on SIDH.

The conjecture that finding isogenies between two elliptic curves is hard has not only been used for key exchange protocols but also for creating hash functions and signature schemes. The Charles-Goren-Lauter (CGL) hash function was proposed in 2006 [DPB17] as a theoretically collision-resistant cryptographic hash function. Most recently, in 2020, a team consisting of De Feo, Kohel, Smith, Petit, and Wesolowski introduced Supersingular Isogeny Signature (SQISign) [FKL+20], the first practical isogeny-based signature scheme Despite its computation time being relatively longer than other post-quantum alternatives, SQISign stands out due to the remarkably small size of its signatures.

## 3.3 Walking and Drawing the Isogeny Graph

Before delving into the protocol, let's build some intuition on how it works by exploring a concept called *isogeny graph*. As theorem 2.17 states, each elliptic curve has a unique j-invariant and two elliptic curves are isomorphic if and only if they have the same j-invariant. In this context, we can represent each unique j-invariant in our set as a collection of elliptic curves with that specific j-invariant. For our example with $p = 431$, we have a set of 37 j-invariants, due to theorem 2.22. Each unique j-invariant can be represented as a vertex in the isogeny graph. Note that finding all j-invariants for a given p is hard, however for our example, and other small primes, it is trivial to find all j-invariants, especially with a tool like SageMath which we will discuss further in section 3.3.1.

We introduce a new prime $\ell \neq p$, which will bring interesting properties to our graph when combined with our set of j-invariants. For each $\ell$, the vertices in our graph remain the same, but the edges connecting them correspond to $\ell$-isogenies. This means that for different values of $\ell$, we will obtain different graphs with the same vertices but distinct edges. In SIDH, Alice and Bob will each move in their own graph, typically with $\ell = 2$ and $\ell = 3$, respectively.

We can start constructing our isogeny graph by selecting a starting curve $E_a : y^2 = x^3 + (208i + 161)x^2 + x$ with $j(E_a) = 364i + 304$ and representing it as a vertex. By identifying the cyclic subgroups of order 2, we can compute 2-isogenies from $E_a$. We generate the kernel using point $P = (\alpha, 0) \in E_a$ with $\alpha = 350i + 68$ which leads us to the image curve $E_{a'} : y^2 = x^3 + (102i + 423)x^2 + x$ with $j(E_a) = 344i + 190$. We can then connect these vertices by drawing an undirected edge between them. Additionally, two other 2-isogeny kernels on $E_a$ can be connected to our vertex. The kernel generated by $(1/\alpha, 0)$ links $j = 364i + 304$ to $j = 67$, and the kernel generated by $(0, 0)$ to $j = 319$. The progression can be viewed in Figure 3.1.

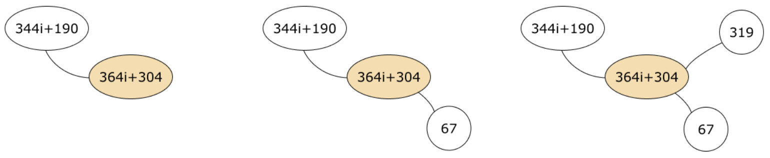


**Figure 3.1:** Drawing the 2-isogeny graph edge by edge by computing 2-isogenies from our starting curve with j-invariant of $364i + 304$.

By continuing this process, we will eventually explore the entire 2-isogeny graph by visiting every j-invariant and drawing each corresponding edge. The complete 2-isogeny graph can be seen in Figure 3.2.

Similarly, we can draw the 3-isogeny graph, but now there are four outgoing edges for every j-invariant. Starting with the same curve $E_a : y^2 = x^3 + (208i + 161)x^2 + x$ with $j(E_a) = 364i + 304$, we take the point $(\beta, \gamma) = (321i + 56, 303i + 174)$ of order 3 on $E_a$. This point serves as the kernel of an isogeny to an image curve $E_{a'} : y^2 = x^3 + 415x^2 + x$ with $j(E_a) = 189$. Plugging the other three 3-torsion subgroups into Velu's formulas yields three additional image curves with j-invariants

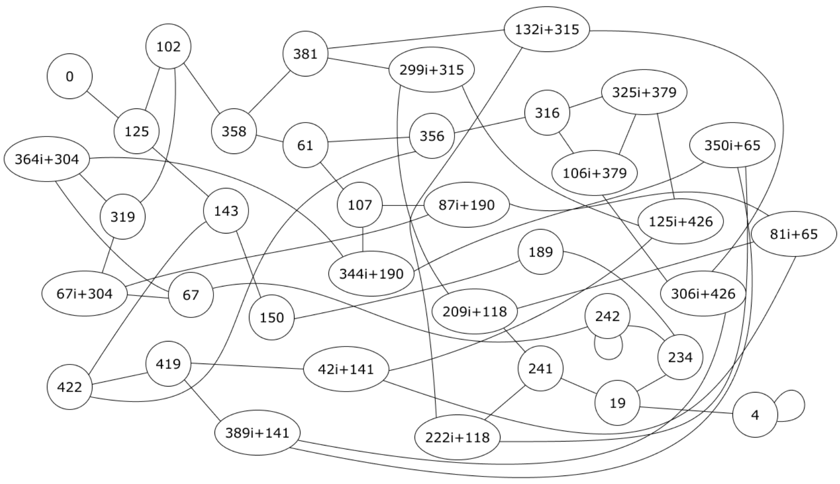


**Figure 3.2:** The full 2-isogeny graph for p = 431. The graph consists of 37 supersingular j-invariants represented as noted and the edges between them correspond to 2-isogenies.

of $j = 19, j = 42i + 141$, and $j = 106i + 379$. We are repeating this process for every node results in the full 3-isogeny graph, as shown in Figure 3.3

### 3.3.1 Discovering the $\ell$-isogeny Graph with SageMath

Using a tool like SageMath, it becomes straightforward to find all supersingular j-invariants and create the corresponding $\ell$-isogeny graph. SageMath includes a function called *isogeny_ell_graph* which returns a graph representing the $\ell$-degree $K$-isogenies between $K$-isomorphism classes of elliptic curves for a field $K$. To discover the full 3-isogeny graph, in SageMath, over the finite field $F_{431}$ and starting curve $E = x^3 + (208i + 161)x^2 + x$, we can do the following.

```
sage: p = 2^4*3^3 -1
sage: F.<i> = GF(p**2, modulus=x^2 + 1)
sage: E = EllipticCurve(F, [0, (208*i +
161), 0, 1, 0])
sage: G = E.isogeny_ell_graph(3, directed=False, label_by_j=True)
sage: G.vertices(sort=True)
['0', '102', '105* i+36', '107', '125', '138* i+357', '14* i+372', '143', '150', '
 166* i+107', '189', '19', '194* i+218', '214* i+389', '217* i+172', '234', '237*
  i+412', '241', '242', '265* i+273', '293* i+64', '316', '319', '326* i+141', '
 336* i+404', '356', '358', '379* i+144', '381', '4', '417* i+386', '419', '422',
  '52* i+92', '61', '67', '95* i+309']
```

**Listing 3.1:** Example of how to find all supersingular j-invariants over a finite field using SageMath

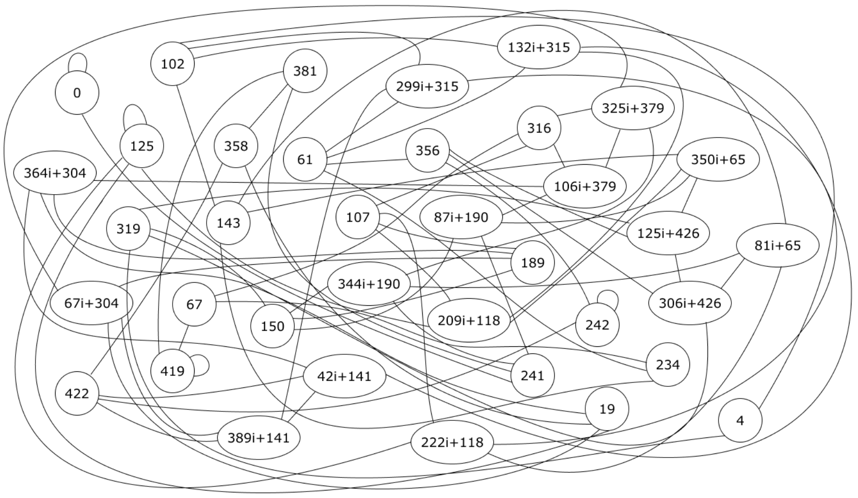


**Figure 3.3:** The full 3-isogeny graph for $p = 431$. The graph consists of 37 supersingular j-invariants represented as noted and the edges between them correspond to 3-isogenies.

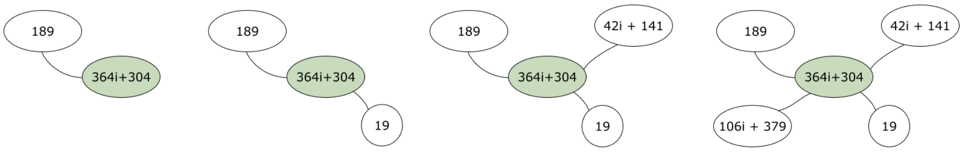


**Figure 3.4:** Drawing the 3-isogeny graph edge by edge by computing 3-isogenies from our starting curve with j-invariant of $364 \cdot i + 304$.

## 3.4 Setting the Stage

Before we are ready to delve into the protocol, we need to set the stage by discussing how SIDH primes are chosen as well as other public parameters. A prime $p$ is chosen of the form

$$p = \ell_A^{e_A} \ell_B^{e_B} f \pm 1$$

where $\ell_A$ and $\ell_B$ are two small, distinct primes and $f$ is a cofactor to ensure that $p$ is a prime. For the majority of instantiations of SIDH, the prime $p$ is of the form

$$p = 2^{e_A} 3^{e_B} f - 1$$

where $e_A$ and $e_B$ are chosen such that $2^{e_A} \approx 3^{e_B}$ and usually $f = 1$ since it is flexible enough to find primes at all security levels. The choices of the two small primes $\ell = 2$ and $\ell = 3$ give us the most efficient instantiations of SIDH. Computing isogenies with Velu's formula is polynomial in the degree of the isogeny. Meaning it is more computational cost to compute isogenies of higher degrees, which facilitates us to compute isogenies of the smallest degree.

It is also worth mentioning that even though the j-invariants in the isogeny graphs are always in $F_{p^2}$, the isogenies corresponding to $\ell \notin \{2, 3\}$ would be computed using points that are not $F_{p^2}$-rational. In other words, points of order $l^e$ for $\ell \notin \{2, 3\}$ are not found in $E(F_{p^2})$, at least for our prime p chosen as above. So to compute these isogenies we would have to perform computations in the larger extension fields of $F_p$ where these points are found.

Our elliptic curve group $E(\mathbb{F}_{p^2})$ is the full $(p + 1)$-torsion for which is isomorphic to $\mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1}$. Thus, primes of the form $p = 2^{e_A}3^{e_B}f - 1$ yields the elliptic curve group

$$E(\mathbb{F}_{p^2}) \cong \mathbb{Z}_{2^{e_A}3^{e_B}} \times \mathbb{Z}_{2^{e_A}3^{e_B}} \times \mathbb{Z}_f \times \mathbb{Z}_f$$

meaning for any $r \in \mathbb{Z}$ with $r|p + 1$, the entire $r$-torsion $E[r] \cong Z_r \times Z_r$ is then contained in $\mathbb{F}_{p^2}$. With this $p$, it follows that the full $2^m$-torsion $E[2^m] \cong Z_{2^m} \times Z_{2^m}$, and the full $3^n$-torsion $E[3^n] \cong Z_{3^n} \times Z_{3^n}$, are both $\mathbb{F}_{p^2}$-rational. Since every isogeny $\phi : E \to E'$ of degree $d$ is in one-to-one correspondence with a kernel subgroup of order $d$ we give the following proposition.

**Proposition 3.1.** *If each isogeny is computed using rational functions of the input curve and the given kernel subgroup, it follows that if both these inputs are* $\mathbb{F}_{p^2}$*-rational, then so is the isogeny computation.*

During the initiation of the protocol, Alice and Bob each select two auxiliary points, which are utilized for creating a secret generator. For Alice, these points have an order of $2^{e_a}$, while for Bob, they have an order of $3^{e_b}$. Any linear combination $[\alpha]P + [\beta]Q$ with $\alpha, \beta \in 2^{e_a}$ can be used to generate all points whose orders are factors of $2^{e_a}$. This ensures that every point we work with lies in $E(\mathbb{F}_{p^2})$.

To further understand the necessity of torsion points, consider the following scheme. Both Alice and Bob choose a secret generator, $S_A$ and $S_B$ respectively, and compute the corresponding isogenies $\phi_A/\langle S_A\rangle$ and $\phi_B/\langle S_B\rangle$ from the starting curve $E$. They then send the new curve to the other party, where each party composes a new isogeny, resulting in a shared curve as depicted in figure 3.4.

$$\begin{tikzcd}
E_0 \arrow[r, "\phi_B"] \arrow[d, "\phi_A"] & E_B \arrow[d, "\phi'_A"] \\
E_A \arrow[r, "\phi'_B"] & E_{AB}
\end{tikzcd}$$

**Figure 3.5:** An isogeny diamond configuration. Both Alice and Bob start at the curve $E_0$ and each computes a secret isogeny. They exchange the curves they landed on and compute a new isogeny where they derive the same curve.

This scheme raises a crucial question: How can Alice evaluate her secret generator $S_A$ at $\phi_B$? Since this is not possible and Bob cannot provide his secret generator for evaluation, an alternative solution must be found. If Bob were to take Alice's basis points and push them through his secret isogeny, Alice could then work with her own torsion subgroup to evaluate at Bob's curve instead of the secret generators itself. In this new scheme, the two parties will not arrive at the same curve, however, the two curves are isomorphic. Remember two isomorphic curves have the same j-invariant, which is enough to make a shared secret.

## 3.5 The Protocol

SIDH is a key agreement scheme in which Alice and Bob establish a shared secret by performing computations involving isogenies between supersingular curves. At a high level, it closely resembles the Diffie-Hellman protocol. We will first provide an overview of the SIDH protocol and then illustrate it with a simplified example. For readers interested in the implementation code or who wish to run the protocol themselves, please refer to GitHub repository

### 3.5.1 Description

**Setup (public parameters).** SIDH is instantiated with $p = 2^{e_A}3^{e_B} - 1$ where $2^{e_A} \approx 3^{e_B}$, and a starting supersingular curve E over $\mathbb{F}_{p^2}$. In addition Alice has two basis points $\{P_A, Q_A\}$ on $E$ of order $2^{e_A}$ and likewise Bob has two basis points $\{P_B, Q_B\}$ on $E$ of order $3^{e_B}$.

**Key generation (Alice).** Alice samples a random $k_A$ and will compute generators of her secret subgroups by computing secret linear combinations of her two basis points $P_A$ and $Q_A$

$$S_A = P_A + [k_A]Q_A \qquad \text{with} \qquad k_A \in [0, 2^{e_A})$$

To calculate her public key, Alice computes her secret isogeny $\phi_A : E \to E_A$, where $E/\langle S_A \rangle$. She accomplishes this through multiple steps by composing $e_A$ isogenies of degree 2, which involves taking $e_A$ steps in the 2-isogeny graph depicted in Figure 3.2. Alice's public key is then represented as the tuple:

$$PK_A = (E_A, P'_B, Q'_B) = (\phi(E), \phi_A(P_B), \phi_A(Q_B))$$

where the first element $E_A = \phi_A(E)$ is the image curve from her starting point after composing $e_A$ isogenies of degree 2. The two other elements Bob's public basis points pushed through Alice's secret isogeny.

**Key generation (Bob).** Bob samples a random $k_B$ and will compute generators of his secret subgroups by computing secret linear combinations of his two basis points $P_B$ and $Q_B$

$$S_B = P_B + [k_B]Q_B \qquad \text{with} \qquad k_B \in [0, 3^{e_B})$$

To compute his public key, he will compute his secret isogeny $\phi_B : E \to E_B$ where $E/\langle S_B \rangle$. Bob will do this in multiple steps by composing $e_B$ isogenies of degree 3. Meaning taking $e_b$ steps in the 3-isogeny graph depicted in Figure 3.3. Bobs public key is then the tuple

$$PK_B = (E_B, P'_A, Q'_A) = (\phi(E), \phi_A(P_B), \phi_B(Q_A))$$

where the first element $E_B = \phi_A(E)$ is the image curve from his starting point after composing $e_B$ isogenies of degree 3. The two other elements Alice's public basis points pushed through Bob's secret isogeny.

**Computing a shared secret.** Upon exchanging public keys, Alice takes her secret integer $k_A$ and, combined with Bob's public key, calculates a new secret subgroup $S'_A = P'_A + [k_A]Q'_A$ on $E_B$. She then computes a new isogeny $\phi'_A : E_A \to E_{AB}$, where $E_{AB} = E_B/\langle S'_A \rangle$. Finally, Alice computes the shared secret $j(E_{AB})$.

Similarly, after exchanging public keys, Bob takes his secret integer $k_B$ and, combined with Alice's public key, calculates a new secret subgroup $S'_B = P'_B + [k_B]Q'_B$

on $E_A$. He then computes a new isogeny $\phi' B : E_B \rightarrow E_{BA}$, where $E_{BA} = E_B/\langle S'_A \rangle$. Bob computes the shared secret $j(E_{BA}) = j(E_{AB})$.

### 3.5.2 SIDH Toy Example

We are now prepared to provide an example of SIDH. For our example, we will utilize the prime $p = 2^4 3^3 - 1 = 431$. According to Theorem 2.22, there are $\lfloor p/12 \rfloor + 2 = 37$ supersingular j-invariants in $\mathbb{F}_{p^2}$. While the size of the set of supersingular j-invariants may appear small, as p increases exponentially, so will the size of the subset. We will adhere to the protocol description provided in 3.5.1.

**Setup (Public parameters)**. We will take the public starting curve used in SIKE

$$E_{a_0} : y^2 = x^3 + a_0 x^2 + x \qquad \text{with} \qquad a_0 = 6 \qquad \text{and} \qquad j(E_{a_0}) = 19$$

We will take the following four basis points,

$$\begin{aligned} P_A &:= (70 \cdot i + 332, 371 \cdot i + 191) && \text{and} && Q_A := (269 \cdot i + 403, 302 \cdot i + 3) \\ P_B &:= (i + 420, 112 \cdot i + 230) && \text{and} && Q_B := (265 \cdot i + 329, 377 \cdot i + 124) \end{aligned}$$

We can assert that $P_A$ and $Q_A$ is of order $2^4$ and $P_B$ and $Q_B$ is of order $3^3$

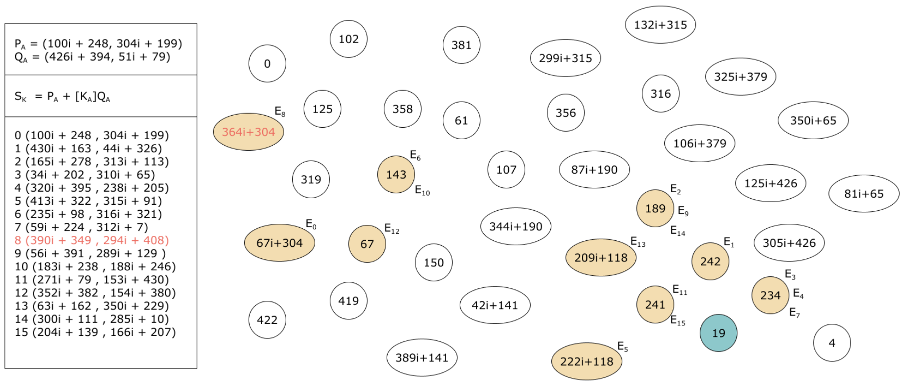
**Key generation (Alice).** Alice begins by choosing a secret

$$k_A := 8$$

from $\{0, 1, ...15\}$. After the secret integer is chosen she will compute her secret generator corresponding to her $k_A$.

$$\begin{aligned} S_A &= P_A + [k_A]Q_A \\ &= (70 \cdot i + 332, 371 \cdot i + 191) + [8](269 \cdot i + 403, 302 \cdot i + 3) \\ &= (361 \cdot i + 332, 371 \cdot i + 240) \end{aligned}$$

This corresponds to a point of order $2^4 = 16$ on our starting curve $E_{a_0}$. In Figure 3.6, we can observe the various nodes Alice would have reached after computing her isogenies for different $k_A$ values. With her secret integer $k_A = 8$, we see that she arrives at the node with a j-invariant of 319. We also observe that $k_A = 7$ would have led to the same j-invariant; however, this does not imply that they will yield identical public and secret keys. It is crucial to remember that we are interested



$P_A$ = (100i + 248, 304i + 199)
$Q_A$ = (426i + 394, 51i + 79)

$S_K$ = $P_A$ + [$K_A$]$Q_A$

0 (100i + 248 , 304i + 199)
1 (430i + 163 , 44i + 326)
2 (165i + 278 , 313i + 113)
3 (34i + 202 , 310i + 65)
4 (320i + 395 , 238i + 205)
5 (413i + 322 , 315i + 91)
6 (235i + 98 , 316i + 321)
7 (59i + 224 , 312i + 7)
8 (390i + 349 , 294i + 408)
9 (56i + 391 , 289i + 129 )
10 (183i + 238 , 188i + 246)
11 (271i + 79 , 153i + 430)
12 (352i + 382 , 154i + 380)
13 (63i + 162 , 350i + 229)
14 (300i + 111 , 285i + 10)
15 (204i + 139 , 166i + 207)

**Figure 3.6:** The set of yellow vertices represents the j-invariants Alice can land on after her key generation.

in the path taken and the isogenies computed. It turns out, the real secret is the isogenies why made along the way.

Alice will now start to compute her public key. Remember Alice will do this by computing $e_A = 4$ number of 2-isogenies, by using a combination of multiplication-by-2, also called point doubling, and inputting the cyclic subgroup of order 2 into Vélus formulas. We will present the first isogeny computed in a detailed manner before the next isogenies are somewhat abbreviated.

*Compute $\phi_0$. input:* $S_A := (361 \cdot i + 332, 371 \cdot i + 240)$, $P_B := (i + 420, 112 \cdot i + 230)$ *and* $Q_B := (265 \cdot i + 329, 377 \cdot i + 124)$ $S_A$ is of order 16, however, we can use the point doubling operation three times to produce a point $R_A = [8]S_A = (373, 0)$, which has order 2 on $E_{a_0}$. Inputting $R_A$ into Vélus formulas gives us

$$\phi_0 : E_{a_0} \to E_{a_1} \qquad \text{with} \qquad a_1 = 142 \cdot i \qquad \text{and} \qquad j(E_{a_1}) = 234.$$

it also gives the rational map $\phi_0 : x \to \frac{x^2+58 \cdot x - 85}{x+58}$, which we will send $S_A$, $P_B$ and $Q_B$ through

$$S_A = \phi_0(S_A) = (332 \cdot i + 232, 420 \cdot i + 253)$$
$$P_B = \phi_0(P_B) = (316 \cdot i + 269, 51 \cdot i + 141)$$
$$Q_B = \phi_0(Q_B) = (193 \cdot i + 265, 310 \cdot i + 249)$$

*Compute $\phi_1$* $S_A := (332 \cdot i + 232, 420 \cdot i + 253)$ $S_A$ is of order 8 and we will use the point doubling operation two times to produce a point $R_A = [4]S_A = (72, 0)$, which has order 2 on $E_{a_1}$. Inputting $R_A$ into Vélus formulas gives us

$$\phi_1 : E_{a_1} \to E_{a_2}, \qquad \text{with} \qquad a_2 = 120 \qquad \text{and} \qquad j(E_{a_2}) = 242.$$

it also gives the rational map $\phi_1 : x \to \frac{x^2-72\cdot x+33}{x-72}$, which we will send $S_A$, $P_B$ and $Q_B$ through

$$\begin{aligned} S_A &= \phi_0(S_A) = (139 \cdot i + 190, 352 \cdot i + 399) \\ P_B &= \phi_0(P_B) = (162 \cdot i + 335, 26 \cdot i + 156) \\ Q_B &= \phi_0(Q_B) = (165 \cdot i + 293, 230 \cdot i + 108) \end{aligned}$$

*Compute* $\phi_2$. $S_A := (139 \cdot i + 190, 352 \cdot i + 399)$ is of order 4 and we will use the point doubling operation one time to produce a point $R_A = [2]S_A = (190, 0)$, which has order 2 on $E_{a_2}$. Inputting $R_A$ into Vélus formulas gives us

$$\phi_2 : E_{a_2} \to E_{a_3}, \qquad \text{with} \qquad a_3 = 221 \qquad \text{and} \qquad j(E_{a_3}) = 67.$$

it also gives the rational map $\phi_2 : x \to \frac{x^2-190\cdot x+74}{x-190}$, which we will send $S_A$, $P_B$ and $Q_B$ through

$$\begin{aligned} S_A &= \phi_0(S_A) = (278 \cdot i + 190, 0) \\ P_B &= \phi_0(P_B) = (192 \cdot i + 372, 62 \cdot i + 355) \\ Q_B &= \phi_0(Q_B) = (139 \cdot i + 1, 215 \cdot i + 194) \end{aligned}$$

*Compute* $\phi_3$. $S_A := (278 \cdot i + 190, 0)$ which is already of order two on $E_{a_2}$. Inputting $R_A$ into Vélus formulas gives us

$$\phi_3 : E_{a_3} \to E_{a_4}, \quad \text{with} \quad a_4 = (223 \cdot i + 270) \quad \text{and} \quad j(E_{a_4}) = 364 \cdot i + 304.$$

it also gives the rational map $\phi_1 : x \to \frac{x^2+(153\cdot i-190)\cdot x+(23\cdot i+161)}{x+(153\cdot i-190)}$, which we will send $P_B$ and $Q_B$ through

$$\begin{aligned} P_B &= \phi_0(P_B) = (324 \cdot i + 185, 11 \cdot i + 166) \\ Q_B &= \phi_0(Q_B) = (379 \cdot i + 90, 406 \cdot i + 317) \end{aligned}$$

Alice's secret $2^4$-isogeny is the composition of the four 2-isogenies computed. $\phi_A : E_{a_0} \to E_{a_4}$, with $\phi_A = (\phi_3 \circ \phi_2 \circ \phi_1 \circ \phi_0)$, see Figure 3.7. Her public key $PK_A$ is then

$$\begin{aligned} PK_A &= (E_{a_4}, \phi_A(P_B), \phi_A(Q_B)) \\ PK_A &= (364 \cdot i + 304, (324 \cdot i + 185, 11 \cdot i + 166), (379 \cdot i + 90, 406 \cdot i + 317)) \end{aligned} \tag{3.1}$$

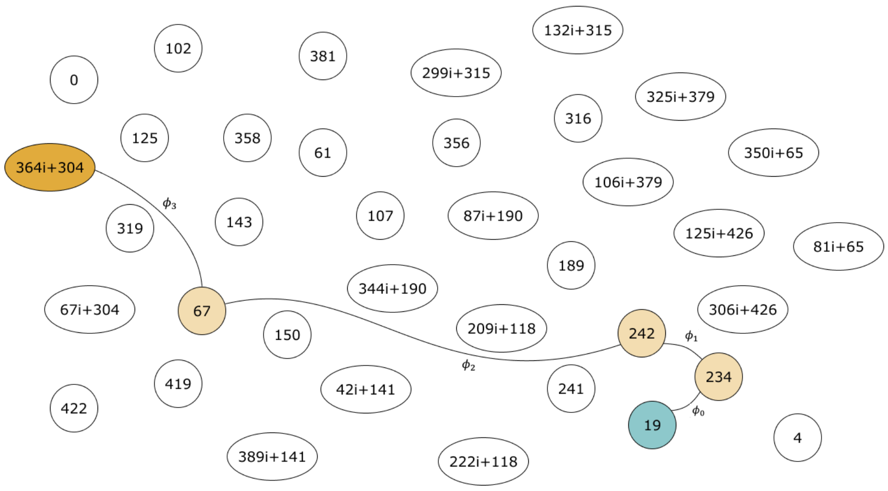


**Figure 3.7:** Alice's key generation path. She starts on the public curve with j-invariant 19, her secret key is the isogeny $\phi_B = (\phi_3 \circ \phi_2 \circ \phi_1 \circ \phi_0)$. The destination node with j-invariant 364i+304 becomes part of her public key.

**Key generation (Bob).** Bob begins by choosing a secret

$$k_B := 3$$

from $\{0, 1, \ldots, 27\}$. After the secret integer is chosen he will compute his secret generator corresponding to his $k_B$.

$$\begin{aligned} S_B &= P_B + [k_B]Q_B \\ &= (i + 420, 112 \cdot i + 230) + [8](265 \cdot i + 329, 377 \cdot i + 124) \\ &= (12 \cdot i + 176, 50 \cdot i + 423) \end{aligned}$$

This corresponds to a point of order $3^3 = 27$ on our starting curve $E_{a_0}$. Bob will now start to compute his public key. Remember Bob will do this by computing $e_B = 3$ number of 3-isogenies, by using a combination of multiplication-by-3, also called point tripling, and inputting the cyclic subgroup of order 3 into Vélus formulas.

*Compute* $\phi_0$*. input:* $S_B := (12 \cdot i + 176, 50 \cdot i + 423)$*,* $P_A := (70 \cdot i + 332, 371 \cdot i + 191)$ *and* $Q_A := (269 \cdot i + 403, 302 \cdot i + 3)$ $S_B$ is of order 27, however, we can use the point doubling operation two times to produce a point $R_B = [9]S_B = (208, 30 \cdot i)$, which has order 3 on $E_{a_0}$. Inputting $R_B$ into Vélus formulas gives us

$$\phi_0 : E_{a_0} \to E_{a_1} \qquad \text{with} \qquad a_1 = 130 \qquad \text{and} \qquad j(E_{a_1}) = 125.$$

it also gives the rational map $\phi_0 : x \to \frac{x^3+15\cdot x^2+108\cdot x-141}{x^2+15\cdot x+164}$, which we will send $S_B$, $P_A$ and $Q_A$ through

$$S_B = \phi_0(S_B) = (100 \cdot i + 152, 376 \cdot i + 118)$$
$$P_A = \phi_0(P_A) = (100 \cdot i + 410, 170 \cdot i + 296)$$
$$Q_A = \phi_0(Q_A) = (239 \cdot i + 391, 371 \cdot i + 13)$$

*Compute* $\phi_1$*.* $S_B := (100{\cdot}i{+}152, 376{\cdot}i{+}118)$ is of order 9, however, we can use the point doubling operation one time to produce a point $R_B = [9]S_A = (113{\cdot}i{+}236, 398{\cdot}i{+}407)$, which has order 3 on $E_{a_1}$. Inputting $R_B$ into Vélus formulas gives us

$$\phi_1 : E_{a_1} \to E_{a_2} \qquad \text{with} \qquad a_1 = 55 \cdot i + 21 \qquad \text{and} \qquad j(E_{a_1}) = 222 \cdot i + 118.$$

it also gives the rational map $\phi_0 : x \to \frac{x^3+(205\cdot i-41)\cdot x^2+(-199\cdot i-157)\cdot x+(144\cdot i-164)}{x^2+(205\cdot i-41)\cdot x+(-108\cdot i-173)}$, which we will send $S_B$, $P_A$ and $Q_A$ through

$$S_B = \phi_0(S_B) = (154 \cdot i + 391, 135 \cdot i + 228)$$
$$P_A = \phi_0(P_A) = (425 \cdot i + 406, 414 \cdot i + 127)$$
$$Q_A = \phi_0(Q_A) = (219 \cdot i + 164, 327 \cdot i + 236)$$

*Compute* $\phi_2$*.* $S_B := (154 \cdot i + 391, 135 \cdot i + 228)$ which is already of order 3 on $E_{a_2}$. Inputting $R_B$ into Vélus formulas gives us

$$\phi_2 : E_{a_2} \to E_{a_3} \qquad \text{with} \qquad a_1 = 275 \cdot i + 299 \qquad \text{and} \qquad j(E_{a_1}) = 107.$$

it also gives the rational map $\phi_0 : x \to \frac{x^3+(123\cdot i+80)\cdot x^2+(-175\cdot i-210)\cdot x+(115\cdot i-60)}{x^2+(123\cdot i+80)\cdot x+(179\cdot i-135)}$, which we will send $P_A$ and $Q_A$ through

$$P_A = \phi_0(P_A) = (347\cdot i+270, 118\cdot i+17)$$
$$Q_A = \phi_0(Q_A) = (35\cdot i+394, 79\cdot i+323)$$

Bob's secret $3^3$-isogeny is the composition of the 3 3-isogenies computed. $\phi_B : E_{a_0} \to E_{a_3}$, with $\phi_B = (\phi_2 \circ \phi_1 \circ \phi_0)$, see figure 3.8. Her public key $PK_B$ is then

$$PK_B = (E_{a_3}, \phi_A(P_A), \phi_A(Q_A))$$
$$PK_B = (273\cdot i+355, (347\cdot i+270, 118\cdot i+17), (35\cdot i+394, 79\cdot i+323)) \quad (3.2)$$

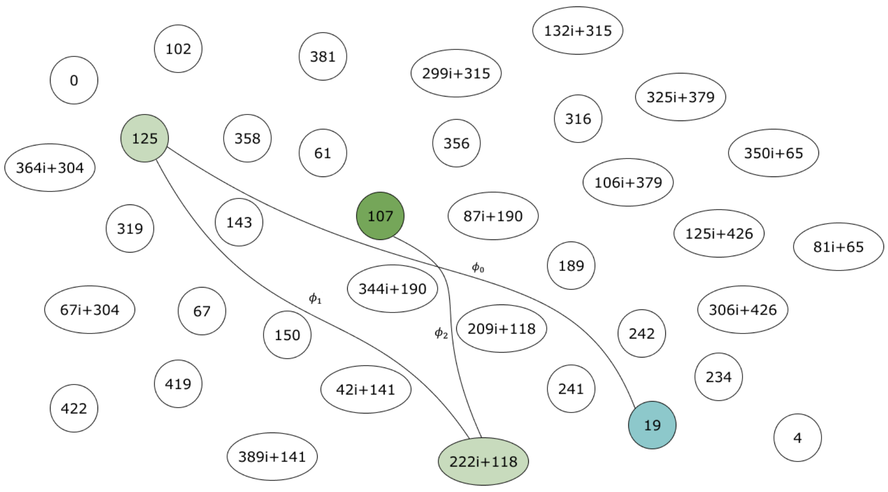


**Figure 3.8:** Bob's key generation path. He starts in the public curve with j-invariant 19, his secret key is the isogeny $\phi_B = (\phi_2 \circ \phi_1 \circ \phi_0)$. The destination node with j-invariant 107 becomes part of his public key.

**Alice's shared secret computation.** Alice begins with the curve output from Bob's public key in 3.2, setting her new starting curve as $E_{a_0}$, where $a_0 = 273\cdot i+355$ and j-invariant $j(E_{a_0}) =$. Similar to her key generation process, Alice's initial step involves computing a secret generator on $E_{a_0}$ that corresponds to her $k_A$:

$$\begin{aligned} S_A &= \phi_B(P_A) + [k_A]\phi_B(Q_A) \\ &= (347 \cdot i + 270, 118 \cdot i + 17) + [8](35 \cdot i + 394, 79 \cdot i + 323) \\ &= (8 \cdot i + 369, 360 \cdot i + 87) \end{aligned}$$

Alice will proceed just as before, performing four new steps in the 2-isogeny graph, with her new generator. Saving computation she will no longer need to push any basis points through the isogeny. We will summarize the four 2-isogenies computations as follow:

$$\begin{array}{llllll} \phi_0 : E_{a_0} \to E_{a_1} & \text{with} & a_1 = 273 \cdot i + 355 & \text{and} & j(E_{a_1}) = 87 \cdot i + 190 \\ \phi_1 : E_{a_1} \to E_{a_2} & \text{with} & a_2 = 259 \cdot i + 162 & \text{and} & j(E_{a_2}) = 67 \cdot i + 304 \\ \phi_2 : E_{a_2} \to E_{a_3} & \text{with} & a_3 = 17 \cdot i + 3 & \text{and} & j(E_{a_3}) = 67 \\ \phi_3 : E_{a_3} \to E_{a_4} & \text{with} & a_4 = 376 & \text{and} & j(E_{a_4}) = 242 \end{array}$$

Alice's shared secret computation is depicted in Figure 3.9

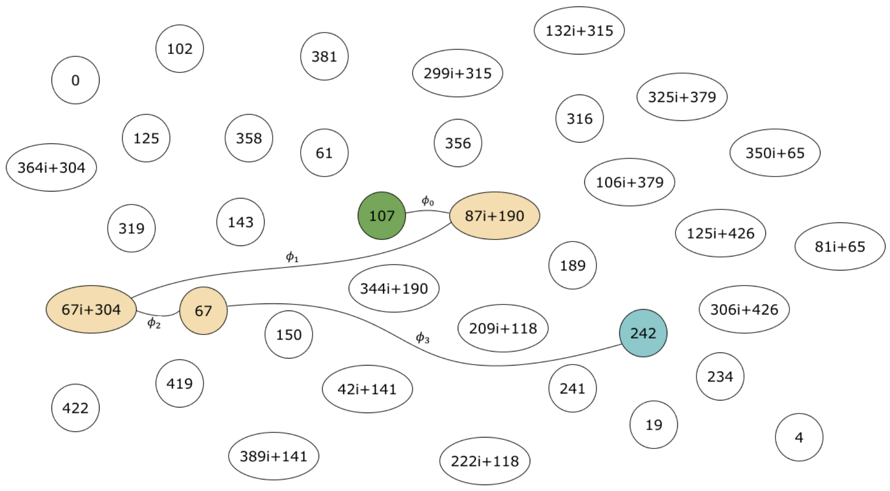


**Figure 3.9:** Alice's shared secret computation. She starts on the curve from Bob's public key with j-invariant 107. She then uses her secret key to compute the remaining walk to the node with j-invariant 242. This is the shared secret.

**Bob's shared secret computation.** Similarly, as Alice, Bob begins with the curve output from Alice's public key in 3.1, setting his new starting curve as $E_{a_0}$, where $a_0 = 364 \cdot i + 304$. Bob's first step is again to compute a secret generator on $E_{a_0}$ that corresponds to her $k_B$:

$$\begin{aligned} S_B &= \phi_A(P_B) + [k_B]\phi_B(Q_B) \\ &= (324 \cdot i + 185, 11 \cdot i + 166) + [3](379 \cdot i + 90, 406 \cdot i + 3173) \\ &= (334 \cdot i + 430, 337 \cdot i + 6) \end{aligned}$$

Bob will proceed just as before, performing three new steps in the 3-isogeny graph, with his new generator. Saving computation he will no longer need to push any basis points through the isogeny. We will summarize the three 3-isogenies computations as follow:

$$\begin{array}{llllll} \phi_0 : E_{a_0} \to E_{a_1} & \text{with} & a_1 = 322 \cdot i + 97 & \text{and} & j(E_{a_1}) = 42 \cdot i + 141 \\ \phi_1 : E_{a_1} \to E_{a_2} & \text{with} & a_2 = 14 \cdot i & \text{and} & j(E_{a_2}) = 422 \\ \phi_2 : E_{a_2} \to E_{a_3} & \text{with} & a_3 = 376 & \text{and} & j(E_{a_3}) = 242 \end{array}$$

Bob's shared secret computation is depicted in Figure 3.10. Revisiting 3.9 we see that both Alice and Bob landed on a curve with j-invariant 242.

## 3.6 Supersingular Isogeny Key Encapsulation

One of SIDH's underlying assumptions is that given the four basis points in the public parameters, and the image points in the public keys do not help a *passive adversary* [1] solve the supersingular isogeny problem we described earlier in 2.4.2. Up until the Castryck and Decru attack, this assumption remained valid. However, Galbraith, Petit, Shani, and Ti [GPST16] showed that if static keys are used and an adversary is active, the entire secret can be learned through repeated interactions equal to the bit length of the key.

This has led to the development of the SIKE protocol, which applies a generic transformation to SIDH, created by Hofheinz, Hövelmanns, and Kiltz [HHK17], which

[1] A passive adversary observes the key exchange in action, collecting information without interfering. An active adversary, on the other hand, interferes with the system's operation, potentially altering data or processes to gain an advantage

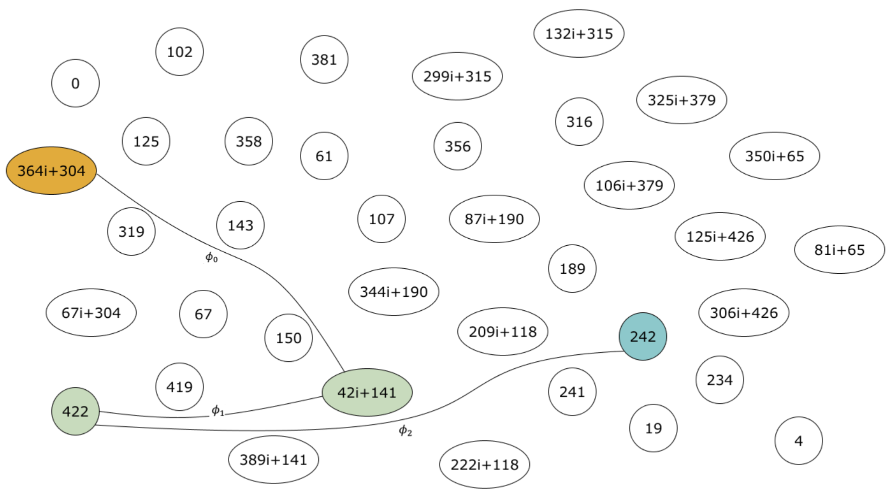


**Figure 3.10:** Bob's shared secret computation. He starts on the curve from Alice's public key with j-invariant $364i + 304$. He then uses his secret key to compute the remaining walk to the node with j-invariant 242. This is the shared secret.

safely allows Alice to use a long-term static secret. In this protocol, Bob calculates the true shared secret before sending any information to Alice, using Alice's fixed public key and a secret key derived from a cryptographic hash function. This hash function uses Alice's public key and a random value as input. Bob also encapsulates the random value by XOR-ing it with a hash of the shared secret and sends this to Alice along with his usual public key.

Alice can then use Bob's public key and her secret key to compute the shared secret and recover Bob's initial random value. She can then recalculate Bob's secret and check that Bob's public key is exactly as it should be. If the check fails, Alice can presume Bob is acting maliciously and output garbage, ensuring that Bob learns nothing about Alice's secret if he tampers with the protocol. More details on the SIKE protocol can be found in [SAJA21].

SIKE was submitted to the NIST standardization process on post-quantum cryptography [NIST]. The protocol reached the 4th round and after Castrcyck and Decru released their attack, SIKE officially withdrew their submission [SIKE].

## 3.7 B-SIDH

In 2019, Costello [Cos20] introduced a novel approach to implementing SIDH, referred to as B-SIDH. This method shares a strong resemblance with SIDH, not only by name but with the primary distinctions stemming from the instantiation process, where Alice and Bob each operate with distinct sets of supersingular curves. In this section, we will first elucidate the differences and the instantiation process, followed by providing a new toy example.

Revisiting section 3.4, we recall that Jao and De Feo selected primes in the form of $p = 2^{e_a}3^{e_b}f - 1$, where Alice's degree is $2^{e_A}$ and Bob's degree is $N = 3^{e_B}$. Until now, both Alice and Bob have operated over the $(p+1)$-torsion, which can be conceptualized as one side of a coin, referred to as the A-side. We now introduce the other side of the coin, the $(p-1)$-torsion, which we will call the B-side. A visual representation of the A-side and B-side can be observed in Figure 3.11.

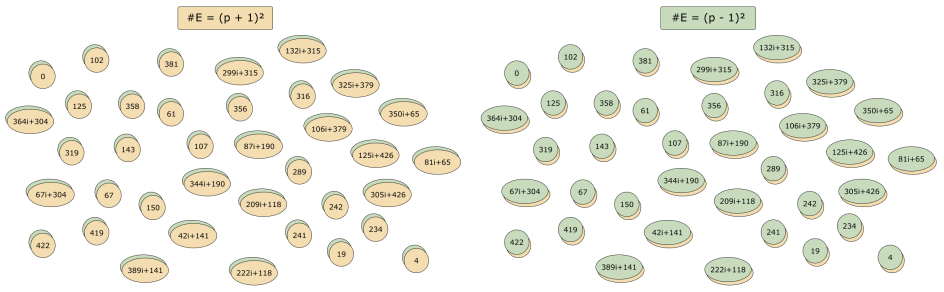


**Figure 3.11:** Illustration of the two sides Alice and Bob will work over. The yellow vertices is representing the A-side $(p+1)$ while the green vertices is representing the B-side $(p-1)$.

With a starting curve $E$, by taking the quadratic twist yields a different $\mathbb{F}_{p^2}$-isomorphism class with group structure

$$E^t(\mathbb{F}_{p^2}) \cong \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$$

Every such supersingular curve with group structure $\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$ is the quadratic twist if a supersingular curve with group structure $\mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1}$, and vice versa. It is essential to recognize that although two quadratic twists are not isomorphic over $\mathbb{F}_{p^2}$, they still have the same j-invariant in $\mathbb{F}_{p^2}$. Similarly, just as any factor $r$ of $p+1$ led to the complete rational $r$-torsion in $E(\mathbb{F}_{p^2})$, any factor $s$ of $p-1$ results in a full rational $s$-torsion in $E^t(\mathbb{F}_{p^2})$.

For a given prime $p$, we use $M$ and $N$ to represent the coprime degrees of Alice and Bob's secret isogenies. Alice's degree, $M$, is always defined such that $M|p+1$, causing Alice to work over the A-side, while Bob's degree, $N$, is always defined such that $N|p-1$, causing Bob to work over the B-side. Ideally, we want $N \approx M \approx p$ for optimal parameter selection.

In SIDH, Alice calculates a secret isogeny of degree $2^{e_A}$ by composing $e_A$ isogenies of degree 2, i.e., taking $e_A$ steps in the 2-isogeny graph. However, in B-SIDH, we cannot solely rely on a single prime power; instead, the isogeny degree will generally be the product of multiple primes. For example, with $p = 161$, we have $M = 80$ and $N = 81$. Alice computes an isogeny of degree $M = 80 = 2^4 \cdot 5$ by composing four 2-isogenies and one 5-isogeny, i.e., taking four steps in the 2-isogeny graph and one step in the 5-isogeny graph.

This approach is possible due to the general conjecture that the hardness of the $L$-isogeny problem depends on the size of $L$, rather than its factorization. This means security is determined by the number of destination nodes Alice and Bob can both reach.

Jao and De Feo chose primes in the form $p = 2^m 3^n - 1$, as computing small-degree isogenies is currently faster. The same principle applies to B-SIDH, so acquiring suitable parameters involves finding primes $p$ for which both $p+1$ and $p-1$ contain factors large enough to achieve a high-security level, while also being smooth enough for efficiency. An integer's smoothness is determined by the size of its prime factors; for example, a 7-smooth number is one where every prime factor is at most 7.

### 3.7.1 Enabled by X-Only Arithmetic

For Alice and Bob to freely work with points coming from the $(p+1)$-torsion and $(p-1)$-torsion one would think we have to lift the entire protocol to $\mathbb{F}_{p^4}$. While it is true the protocol will be lifted to $\mathbb{F}_{p^4}$ Alice and Bob can still work entirely in $\mathbb{F}_{p^2}$, thanks to the use of twisting isomorphisms and the nature of Montgomery curves that allow ignoring y-coordinates and certain curve coefficients.

The secret points chosen by Alice from the $(p+1)$-torsion of E and Bob from the $(p-1)$-torsion of $E^t$ can be transformed by two isogeny maps $\phi_1$ and $\phi_2$. When these are lifted to $\mathbb{F}_{p^4}$ and precomposed with the twisting morphisms, we get isogenies that work well over $\mathbb{F}_{p^4}$. However, due to the property of the $\phi$ map, which we explored in 2.3.2, the actual implementation can ignore the y-coordinates and work essentially in $\mathbb{F}_{p^2}$.

This way, the SIDH protocol can remain mostly unchanged in its implementation while benefiting from the efficiency gains of x-only arithmetic on Montgomery curves.

### 3.7.2 Choosing a Friendly Prime p

We briefly mentioned choosing good friendly primes is crucial to have a fast and efficient application of B-SIDH. We want to find primes p where both $M|p+1$ and $N|p-1$ are as smooth as possible and large enough to reach a required security level. Costello [Cos20] discusses various methods for identifying friendly primes and we will mention some of them here.

When choosing which primes it is important to keep in mind two things:

– One party has to compute some or all factors in $p+1$ and the other has to compute some or all factors in $p-1$.
– Computing higher-degree isogenies requires more computational resources.

Finding primes where both parties have smooth order is not always feasible and as a result, we often have to compromise and burden one party with more computation to allow the other party to work more efficiently.

However, in some practical applications, this may not be a problem. A server oftentimes will perform the protocol at a much greater volume than an individual client. In this case, burdening the client to speed up the server's operations could be an acceptable trade-off.

For the purpose of our discussion, we'll classify primes into three groups, mainly based on Alice's order $M$. The rationale behind this focus will become clearer in the subsequent chapter, as it relates directly to the nature of the attack we'll be discussing.

**Primes where Alice's order is $2^m$.**

These primes can be written in the form

$$p = 2^m \cdot c - 1$$

enables Alice to work with an order of $2^m$. Primes resulting in 2 dividing $p+1$ are already popular in much of the Elliptic Curve Cryptography (ECC) literature, e.g., Mersenne and Ridinghood primes. Although these primes are fast and efficient for Alice, their scarcity means that $p-1$ is unlikely to be smooth.

Taking $c = 1$ yields Mersenne primes for which $m = \{31, 61, 127, 521\}$ being of interest. While $m = 31, 61$ offers no cryptographic security, they are relevant for developing the necessary changes to the attack in the following chapter. Alice can utilize $2^e$-isogenies for any $e \leq m$, allowing her to adjust her security level according to requirements.

Bob can compute L-isogenies for any $L|p-1$. For example, when $m = 127$, $p - 1 = 2 \cdot 3^3 \cdot 7^2 \cdot 19 \cdot 43 \cdot 73 \cdot 127 \cdot 337 \cdot 5419 \cdot 92737 \cdot 649657 \cdot 77158673929$, and Bob can choose any of the factors.

**Primes where Alice's order is $2^m 3^n$**

These are primes of the form

$$p = 2^f \cdot 3^e \cdot c - 1$$

From a practical standpoint, selecting primes in this manner is not ideal, as we generally prefer Alice to mostly work with 2-factors and Bob to mostly work with 3-factors to maximize efficiency for both parties. Nevertheless, to fully comprehend the scope of the attack, it is helpful to choose primes such that Alice's order is $2^m 3^n$. Finding these primes is straightforward, but in most cases, Bob's order will not be smooth.

**Primes where Alice's order is $\ell_1^{k_1} \ell_2^{k_2} \dots \ell_i^{k_i}$**

These are primes of the form

$$p = \ell_1^{k_1} \ell_2^{k_2} \dots \ell_i^{k_i} - 1$$

We can generalize these primes to where the order for both Alice and Bob is as smooth as possible. One of these primes is the 382-bit prime:

$$\begin{aligned} p :=& 0x277AF122D68C175343851A906212321112FB72C2AAB291357 \\ & 9000000000000000000000000000000000000000000001 \end{aligned} \tag{3.3}$$

with

$$\begin{aligned} M &= 3^{115} \cdot 7 \cdot 13 \cdot 312 \cdot 157 \cdot 241 \text{ and} \\ N &= 2^{188} \cdot 11 \cdot 17 \cdot 29 \cdot 73 \cdot 193, \end{aligned}$$

These sizes offer security comparable to SIKEp434.

### 3.7.3 Handling Large $\ell$-degree Isogenies

We have not delved deeply into the runtime of isogeny calculations thus far, primarily because computing small-degree isogenies is relatively fast. However, B-SIDH involves much larger isogenies compared to SIDH. When calculating $\ell$-isogenies with Velu's formulas, where $\ell$ is a prime, the process requires $\mathcal{O}(\ell)$ field operations. Considering the primes mentioned in Section 3.7.2, some prime factors are quite large, which can result in isogeny calculations taking several seconds or even minutes. For example, when working with the prime $p = 2^{127} - 1$ discussed earlier, Bob needs to compute a 37-bit degree isogeny.

There are some practical strategies to speed up isogeny calculations. In 2020, Bernstein, De Feo, Leroux, and Smith [BFLS20] introduced a significant improvement for computing large prime-degree isogenies. They demonstrated a method that requires only $\mathcal{O}(\sqrt{\ell})$ field operations for $\ell$-isogenies. In recent times SageMath implemented this feature using an algorithm called $\sqrt{élu}$ [2] to compute isogenies of elliptic curves in time $\mathcal{O}(\sqrt{\ell})$. We will take a closer look at the usage of $\sqrt{élu}$ in section 5.3.

Costello [Cos20] suggests two methods for accelerating isogeny calculations in practice: *parallelization* and *precomputation*. An algorithm presented in [CH17], lends itself to parallelization almost perfectly. For large enough $\ell$-isogenies one can shorten the final multiplications and squarings down to $log(\ell)$.

For large prime-degree isogenies, Bob can shorten the runtime significantly if the procedure allows for the storage of offline precomputation. For example, where Bob is the one generating ephemeral public keys and his largest degree isogeny is a $\ell$-isogeny, he could precompute all of the $\ell + 1$ possible image curves. At runtime, he could simply choose the isogeny corresponding to his secret key.

### 3.7.4 The Protocol

As previously mentioned, B-SIDH largely resembles SIDH, with the primary distinction being the instantiation process. We will offer a high-level overview of the B-SIDH protocol and demonstrate it using a simple toy example [3].

#### Description

**Setup (public parameters).** B-SIDH is instantiated with a given prime p, and M and N will be used to denote two coprime degrees of Alice and Bob's secret isogenies. Alice's degree $M$ will be defined such that $M|p + 1$ and Bob's $N$ such that $N|p - 1$.

Alice begins with a supersingular curve $E$ defined over the field $\mathbb{F}_{p^2}$, while Bob starts with $E's$ quadratic twist, denoted as $E^t$. Alice's and Bob's curves are not

[2] $\sqrt{élu}$ is a symbol for the name Squareroot Vélu

[3] The implementation code can be found in https://github.com/georgsku/BSIDH-attack/blob/main/bsidh.sage

isomorphic over $\mathbb{F}_{p^2}$, however, they do share the same j-invariant in $\mathbb{F}_{p^2}$. Furthermore, Alice has two basis points $\{P_A, Q_A\}$ on curve $E$, both of order $M$, and Bob has two basis points $\{P_B, Q_B\}$ on curve $E^t$, both of order $N$.

**Key generation (Alice).** Alice samples a random $k_A$ and will compute generators of her secret subgroups by computing secret linear combinations of her two basis points $P_A$ and $Q_A$

$$S_A = P_A + [k_A]Q_A \qquad \text{with} \qquad k_A \in [0, M)$$

To calculate her public key, Alice computes her secret isogeny $\phi_A : E \to E_A$, where $E/\langle S_A \rangle$. She accomplishes this through multiple steps by composing isogenies of the prime factors of $M$. Meaning where $M = \ell_1^{k_1}\ell_2^{k_2}\ldots\ell_i^{k_i}$ she will take $k_1$ steps in the $\ell_1$-graph, $k_2$ steps in the $\ell_2$-graph and so on. Alice's public key is then represented as the tuple:

$$PK_A = (E_A, P'_B, Q'_B) = (\phi(E), \phi_A(P_B), \phi_A(Q_B))$$

where the first element $E_A = \phi_A(E)$ is the image curve from her starting point after composing a secret isogeny of degree $M$. The two other elements are Bob's public basis points pushed through Alice's secret isogeny.

**Key generation (Bob).** Bob samples a random $k_b$ and will compute generators of his secret subgroups by computing secret linear combinations of her two basis points $P_B$ and $Q_B$

$$S_B = P_B + [k_b]Q_B \qquad \text{with} \qquad k_B \in [0, N)$$

To calculate his public key, Bob computes his secret isogeny $\phi_B : E \to E_B$, where $E/\langle S_B \rangle$. He accomplishes this through multiple steps by composing isogenies of the prime factors of $N$. Meaning where $N = \ell_1^{k_1}\ell_2^{k_2}\ldots\ell_i^{k_i}$ she will take $k_1$ steps in the $\ell_1$-graph, $k_2$ steps in the $\ell_2$-graph and so on. Bob's public key is then represented as the tuple:

$$PK_B = (E_B, P'_A, Q'_A) = (\phi(E^t), \phi_A(P_A), \phi_A(Q_A))$$

where the first element $E_B = \phi_A(E^t)$ is the image curve from his starting point after composing a secret isogeny of degree $N$. The two other elements are Alice's public basis points pushed through Bob's secret isogeny.

**Computing a shared secret.** Upon exchanging public keys, Alice takes her secret integer $k_A$ and, combined with Bob's public key, calculates a new secret subgroup $S'_A = P'_A + [k_A]Q'A$ on $E_B$. She then computes a new isogeny $\phi' A : E_A \to E_{AB}$, where $E_{AB} = E_B/\langle S'A\rangle$. Finally, Alice computes the shared secret $j(E_{AB})$.

Similarly, after exchanging public keys, Bob takes his secret integer $k_B$ and, combined with Alice's public key, calculates a new secret subgroup $S'_B = P'B + [k_B]Q'B$ on $E_A$. He then computes a new isogeny $\phi'_B : E_B \to E_{BA}$, where $E_{BA} = E_B/\langle S'A\rangle$. Bob computes the shared secret $j(E_{BA}) = j(E_{AB})$.

### 3.7.5 B-SIDH Toy Example

For our B-SIDH example, we will still utilize the prime $p = 2^4 3^3 - 1 = 431$. Remember Alice will work over $(p + 1)$, which means her order will be $N = 2^4 \cdot 3^3$, and Bob will work over $p - 1$ resulting in order $M = 5 \cdot 43$. We will adhere to the protocol description provided in 3.7.4.

It is important to note that a modification is introduced in the implementation in order to save time. The implementation operates over $\mathbb{F}_{p^4}$ to circumvent the use of x-only arithmetic, as SageMath does not currently support this feature. Although this approach results in slower performance, it is not a concern for this thesis, as the outcome remains the same.

Operating over $\mathbb{F}_{p^4}$ means a few things change. Some $j$-invariants will get another representation, however, these can be mapped back to $\mathbb{F}_{p^2}$ using the Table in 3.1. We will also omit the $y$-coordinate, denoted by $-$, in order to avoid long tuples and simulate how B-SIDH with x-only arithmetic works. Lastly, Bob will not start on the quadratic twist but on the same curve as Alice. We are working in $\mathbb{F}_{p^4} = \mathbb{F}_p(\omega)$ where $\omega^4 + 2\omega^2 + 323\omega + 7 = 0$.

**Setup (Public parameters).** We will utilize the public starting curve used in SIKE, which means both Alice and Bob will start on

$$E_{a_0} : y^2 = x^3 + a_0 x^2 + x \qquad \text{with} \qquad a_0 = 6 \qquad \text{and} \qquad j(E_{a_0}) = 19$$

We will take the following four basis points,

| $\mathbb{F}_{p^4} \to \mathbb{F}_{p^2}$ |
|---|
| $313 \cdot \omega^3 + 93 \cdot \omega^2 + 254 \cdot \omega + 287 \to 209 \cdot i + 118$ |
| $118 \cdot \omega^3 + 338 \cdot \omega^2 + 177 \cdot \omega + 380 \to 222 \cdot i + 118$ |
| $12 \cdot \omega^3 + 217 \cdot \omega^2 + 18 \cdot \omega + 297 \to 344 \cdot i + 190$ |
| $134 \cdot \omega^3 + 340 \cdot \omega^2 + 201 \cdot \omega + 209 \to 106 \cdot i + 379$ |
| $143 \cdot \omega^3 + 395 \cdot \omega^2 + 430 \cdot \omega + 322 \to 364 \cdot i + 304$ |
| $151 \cdot \omega^3 + 396 \cdot \omega^2 + 11 \cdot \omega + 228 \to 306 \cdot i + 426$ |
| $199 \cdot \omega^3 + 402 \cdot \omega^2 + 83 \cdot \omega + 371 \to 389 \cdot i + 141$ |
| $232 \cdot \omega^3 + 29 \cdot \omega^2 + 348 \cdot \omega + 342 \to 42 \cdot i + 141$ |
| $175 \cdot \omega^3 + 399 \cdot \omega^2 + 47 \cdot \omega + 331 \to 132 \cdot i + 315$ |
| $288 \cdot \omega^3 + 36 \cdot \omega^2 + \omega + 286 \to 67 \cdot i + 304$ |
| $297 \cdot \omega^3 + 91 \cdot \omega^2 + 230 \cdot \omega + 118 \to 325 \cdot i + 379$ |
| $280 \cdot \omega^3 + 35 \cdot \omega^2 + 420 \cdot \omega + 193 \to 125 \cdot i + 426$ |
| $313 \cdot \omega^3 + 93 \cdot \omega^2 + 254 \cdot \omega + 287 \to 209 \cdot i + 118$ |
| $353 \cdot \omega^3 + 98 \cdot \omega^2 + 314 \cdot \omega + 16 \to 350 \cdot i + 65$ |
| $419 \cdot \omega^3 + 214 \cdot \omega^2 + 413 \cdot \omega + 83 \to 87 \cdot i + 190$ |

**Table 3.1:** Mapping j-invariants in $\mathbb{F}_{p^4}$ to $\mathbb{F}_{p^2}$. Omega is a root of the irreducible polynomial $x^4 + 2 \cdot x^2 + 323 \cdot x + 7$.

$$\begin{gathered} P_A := (136 \cdot \omega^3 + 17 \cdot \omega^2 + 204 \cdot \omega + 123, -) \\ Q_A := (218 \cdot \omega^3 + 135 \cdot \omega^2 + 327 \cdot \omega + 222, -) \\ \text{and} \\ P_B := (380 \cdot \omega^3 + 263 \cdot \omega^2 + 139 \cdot \omega + 292, -) \\ Q_B := (336 \cdot \omega^3 + 42 \cdot \omega^2 + 73 \cdot \omega + 120, -) \end{gathered}$$

We can assert that $P_A$ and $Q_A$ order devides $p+1$ and $P_B$ and $Q_B$ order devides $p-1$

**Key generation (Alice).** Alice begins by choosing a secret

$$k_A := 314$$

from $\{0, 1, ...N\}$. After the secret integer is chosen she will compute her secret generator corresponding to her $k_A$.

$$\begin{aligned} S_A &= P_A + [k_A]Q_A \\ &= (136 \cdot \omega^3 + 17 \cdot \omega^2 + 204 \cdot \omega + 123, -) \\ &+ [314](218 \cdot \omega^3 + 135 \cdot \omega^2 + 327 \cdot \omega + 222, -) \\ &= (346 \cdot \omega^3 + 151 \cdot \omega^2 + 88 \cdot \omega + 386, -) \end{aligned}$$

Which corresponds to a point of order $M$ on our starting curve $E_{a_0}$. Alice will create her public key by computing a secret isogeny of degree $N = 2^4 3^3$. She will accomplish this by composing 4 numbers of 2-isogenies, and 3 numbers of 3-isogenies. In other words, she will take 4 steps in the 2-isogeny graph, Figure 3.2, and 3 steps in the 3.3. We will present the first isogeny computed in a detailed manner before the next isogenies are abbreviated.

## 2-isogenies

*Compute* $\phi_0$. *input:* $S_A := (346 \cdot \omega^3 + 151 \cdot \omega^2 + 88 \cdot \omega + 386, -)$, $P_B := (136 \cdot \omega^3 + 17 \cdot \omega^2 + 204 \cdot \omega + 123, -)$ *and* $Q_B := (218 \cdot \omega^3 + 135 \cdot \omega^2 + 327 \cdot \omega + 222, -)$ $S_A$ is of order $2^4 3^3$, however, we can use a combination of the point doubling operation and point tripling to produce a point $R_A = [2^3 3^3]S_A = (0, -)$, which has order 2 on $E_{a_0}$. Inputting $R_A$ into Vélus formulas gives us

$$\phi_0 : E_{a_0} \to E_{a_1} \qquad \text{with} \qquad a_1 = 0 \qquad \text{and} \qquad j(E_{a_1}) = 4$$

it also gives the rational map $\phi_0 : x \to \frac{x^2+1}{x}$, which we will send $S_A$, $P_B$ and $Q_B$ through

$$\begin{aligned} S_A &= \phi_0(S_A) = (247 \cdot \omega^3 + 408 \cdot \omega^2 + 155 \cdot \omega + 80, -) \\ P_B &= \phi_0(P_B) = (43 \cdot \omega^3 + 167 \cdot \omega^2 + 280 \cdot \omega + 130, -) \\ Q_B &= \phi_0(Q_B) = (40 \cdot \omega^3 + 5 \cdot \omega^2 + 60 \cdot \omega + 39, -) \end{aligned}$$

$$\begin{aligned} &\phi_1 : E_{a_1} \to E_{a_2} && \text{with} && a_2 = (378 \cdot \omega^3 + 155 \cdot \omega^2 + 136 \cdot \omega + 138) \\ & && \text{and} && j(E_{a_2}) = 19 \\ &\phi_2 : E_{a_2} \to E_{a_3} && \text{with} && a_3 = 39 \\ & && \text{and} && j(E_{a_3}) = 241 \\ &\phi_3 : E_{a_3} \to E_{a_4} && \text{with} && a_4 = (37 \cdot \omega^3 + 274 \cdot \omega^2 + 271 \cdot \omega + 273) \\ & && \text{and} && j(E_{a_4}) = 209 \cdot i + 118 \end{aligned}$$

**3-isogenies**

$$\begin{aligned}
\phi_4 : E_{a_4} \to E_{a_5} \quad &\text{with} & a_5 &= 284 \cdot \omega^3 + 251 \cdot \omega^2 + 426 \cdot \omega + 408 \\
&\text{and} & j(E_{a_5}) &= 132 \cdot i + 315 \\
\phi_5 : E_{a_5} \to E_{a_6} \quad &\text{with} & a_6 &= 63 \cdot \omega^3 + 385 \cdot \omega^2 + 310 \cdot \omega + 23 \\
&\text{and} & j(E_{a_6}) &= 102 \\
\phi_6 : E_{a_6} \to E_{a_7} \quad &\text{with} & a_7 &= 406 \\
&\text{and} & j(E_{a_7}) &= 143
\end{aligned}$$

Alice's secret $M$-isogeny is the composition of the four 2-isogenies and three 3-isogenies computed. $\phi_A : E_{a_0} \to E_{a_7}$, with $\phi_A = (\phi_7 \circ \phi_6 \circ \phi_5 \circ \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1 \circ \phi_0)$, see Figure 3.12. Her public key $PK_A$ is then

$$\begin{aligned}
PK_A &= (E_{a_7}, \phi_A(P_B), \phi_A(Q_B)) \\
PK_A &= (406, (183 \cdot \omega^3 + 400 \cdot \omega^2 + 59 \cdot \omega + 274), \\
&\quad (213 \cdot \omega^3 + 296 \cdot \omega^2 + 104 \cdot \omega + 84, -))
\end{aligned} \tag{3.4}$$

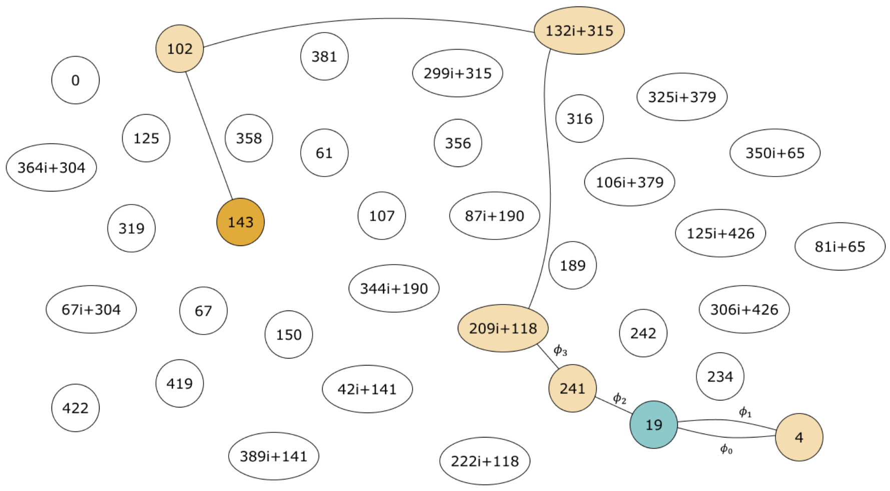


**Figure 3.12:** Alice's key generation path. She starts on the public curve with j-invariant 19, her secret key is the isogeny $\phi_B = (\phi_6 \circ \phi_5 \circ \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1 \circ \phi_0)$. The destination node with j-invariant 140 becomes part of her public key.

**Key generation (Bob).** Bob begins by choosing a secret

$$k_A := 91$$

from $\{0, 1, ...M\}$. After the secret integer is chosen he will compute his secret generator corresponding to his $k_B$.

$$\begin{aligned} S_B &= P_B + [k_B]Q_B \\ &= (380 \cdot \omega^3 + 263 \cdot \omega^2 + 139 \cdot \omega + 292, -) \\ &+ [91](336 \cdot \omega^3 + 42 \cdot \omega^2 + 73 \cdot \omega + 120, -) \\ &= (416 \cdot \omega^3 + 52 \cdot \omega^2 + 193 \cdot \omega + 257, -) \end{aligned}$$

Which corresponds to a point of order $N$ on our starting curve $E_{a_0}$. Bob will create his public key by computing a secret isogeny of degree $N = 5 \cdot 43$. he will accomplish this by composing a 5-isogeny and a 43-isogeny. In other words, he will take one step in the 5-isogeny graph and then another step in the 43-isogeny graph.

**5-isogeny**

$$\begin{aligned} \phi_0 : E_{a_0} \to E_{a_1} \qquad &\text{with} \qquad a_1 = (31 \cdot \omega^3 + 381 \cdot \omega^2 + 262 \cdot \omega + 402) \\ &\text{and} \qquad j(E_{a_1}) = 325 \cdot i + 379 \end{aligned}$$

**43-isogeny**

$$\begin{aligned} \phi_1 : E_{a_1} \to E_{a_2} \qquad &\text{with} \qquad a_2 = (194 \cdot \omega^3 + 132 \cdot \omega^2 + 291 \cdot \omega + 373) \\ &\text{and} \qquad j(E_{a_2}) = 87 \cdot i + 190 \end{aligned}$$

Bob's secret $N$-isogeny is the composition of the 5-isogeny and 43-isogeny computed. $\phi_A : E_{a_0} \to E_{a_2}$, with $\phi_A = (\phi_1 \circ \phi_0)$, see Figure 3.13. His public key $PK_B$ is then

$$\begin{aligned} PK_B &= (E_{a_2}, \phi_B(P_A), \phi_B(Q_A)) \qquad (3.5) \\ PK_B &= ((194 \cdot \omega^3 + 132 \cdot \omega^2 + 291 \cdot \omega + 373), \\ &\quad (17 \cdot \omega^3 + 56 \cdot \omega^2 + 241 \cdot \omega + 256, -), (264 \cdot \omega^3 + 33 \cdot \omega^2 + 396 \cdot \omega + 311, -)) \end{aligned}$$

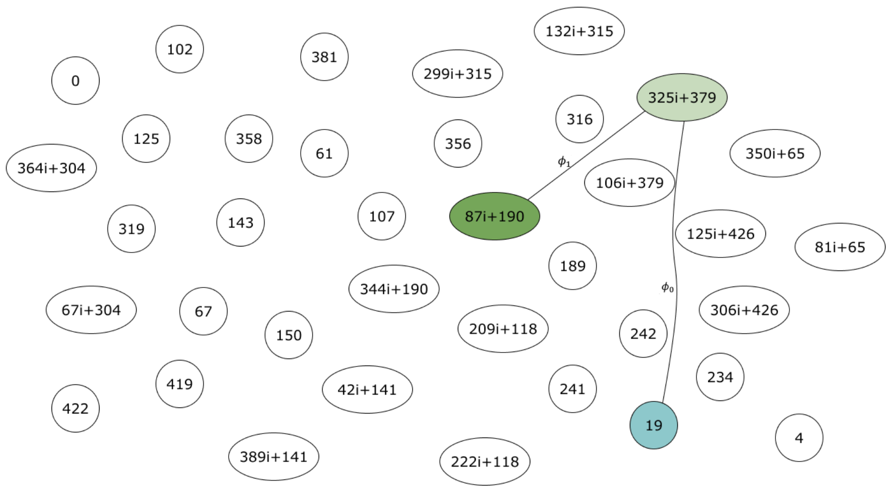


**Figure 3.13:** Bob's key generation path. He starts on the public curve with j-invariant 19, his secret key is the isogeny $\phi_B = (\phi_1 \circ \phi_0)$. The destination node with j-invariant $87i + 190$ becomes part of his public key.

**Alice's shared secret computation.** Alice begins with the curve output from Bob's public key in 3.5, setting her new starting curve as $E_{a_0}$, where $a_0 = (194 \cdot \omega^3 + 132 \cdot \omega^2 + 291 \cdot \omega + 373)$ and j-invariant $j(E_{a_0}) = 87 \cdot i + 190$. Similar to her key generation process, Alice's initial step involves computing a secret generator on $E_{a_0}$ that corresponds to her $k_A$:

$$\begin{aligned} S_A &= \phi_B(P_A) + [k_A]\phi_B(Q_A) \\ &= (17 \cdot \omega^3 + 56 \cdot \omega^2 + 241 \cdot \omega + 256, -) + [314](264 \cdot \omega^3 + 33 \cdot \omega^2 + 396 \cdot \omega + 311, -) \\ &= (108 \cdot \omega^3 + 229 \cdot \omega^2 + 162 \cdot \omega + 425, -) \end{aligned}$$

Alice will proceed just as before, performing four new steps in the 2-isogeny graph and three steps in the 3-isogeny graph, with her new generator. Saving computation she will no longer need to push any basis points through the isogeny.

**2-isogenies**

$$\begin{aligned}
&\phi_0 : E_{a_0} \to E_{a_1} && \text{with} && a_1 = (231\cdot\omega^3 + 406\cdot\omega^2 + 131\cdot\omega + 226) && \text{and} && j(E_{a_1}) = 81\cdot i + 65\\
&\phi_1 : E_{a_1} \to E_{a_2} && \text{with} && a_2 = (259\cdot\omega^3 + 194\cdot\omega^2 + 173\cdot\omega + 22) && \text{and} && j(E_{a_2}) = 42\cdot i + 141\\
&\phi_2 : E_{a_2} \to E_{a_3} && \text{with} && a_3 = (317\cdot\omega^3 + 309\cdot\omega^2 + 260\cdot\omega + 63) && \text{and} && j(E_{a_3}) = 125\cdot i + 426\\
&\phi_3 : E_{a_3} \to E_{a_4} && \text{with} && a_4 = (286\cdot\omega^3 + 359\cdot\omega^2 + 429\cdot\omega + 38) && \text{and} && j(E_{a_4}) = 325\cdot i + 379
\end{aligned}$$

**3-isogenies**

$$\begin{aligned}
&\phi_4 : E_{a_4} \to E_{a_5} && \text{with} && a_5 = 328\cdot\omega^3 + 41\cdot\omega^2 + 61\cdot\omega + 34 && \text{and} && j(E_{a_5}) = 67\cdot i + 304\\
&\phi_5 : E_{a_5} \to E_{a_6} && \text{with} && a_6 = 259\cdot\omega^3 + 194\cdot\omega^2 + 173\cdot\omega + 215 && \text{and} && j(E_{a_6}) = 389\cdot i + 141\\
&\phi_6 : E_{a_6} \to E_{a_7} && \text{with} && a_7 = (259\cdot\omega^3 + 194\cdot\omega^2 + 173\cdot\omega + 22) && \text{and} && j(E_{a_7}) = 42\cdot i + 141
\end{aligned}$$

Alice's shared secret computation is depicted in Figure 3.14

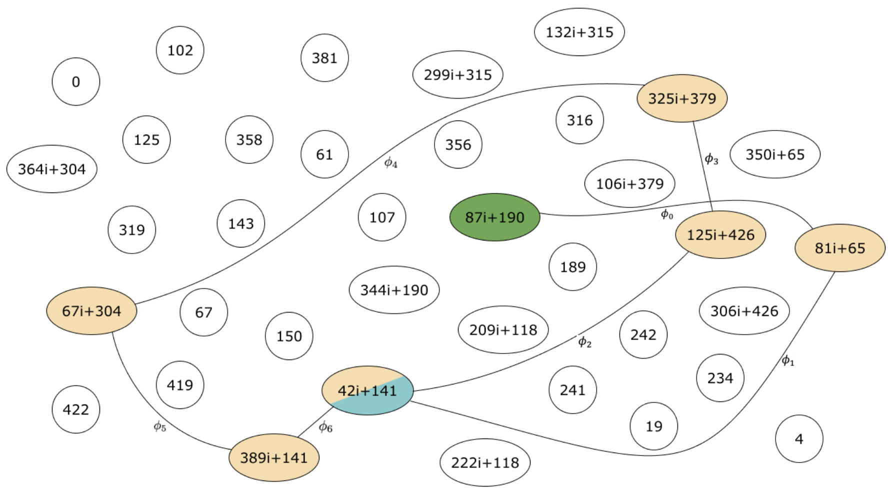


**Figure 3.14:** Alice's shared secret computation. She starts on the curve from Bob's public key with j-invariant $87i + 190$. She then uses her secret key to compute the remaining walk to the node with j-invariant $42i + 141$. This is the shared secret.

**Bob's shared secret computation.** Similarly, as Alice, Bob begins with the curve output from Alice's public key in 3.1, setting his new starting curve as $E_{a_0}$, where $a_0 = 364\cdot i + 304$. Bob's first step is again to compute a secret generator on $E_{a_0}$ that corresponds to her $k_B$:

$$
\begin{aligned}
S_B &= \phi_A(P_B) + [k_B]\phi_B(Q_B) \\
&= (324 \cdot i + 185, 11 \cdot i + 166) + [91](379 \cdot i + 90, 406 \cdot i + 3173) \\
&= (346 \cdot \omega^3 + 151 \cdot \omega^2 + 88 \cdot \omega + 146, -)
\end{aligned}
$$

Bob will proceed just as before, performing three new steps in the 3-isogeny graph, with his new generator. Saving computation he will no longer need to push any basis points through the isogeny. We will summarize the three 3-isogenies computations as follow:

$$
\begin{array}{llllll}
\phi_0 : E_{a_0} \to E_{a_1} & \text{with} & a_1 = 322 \cdot i + 97 & \text{and} & j(E_{a_1}) = 42 \cdot i + 141 \\
\phi_1 : E_{a_1} \to E_{a_2} & \text{with} & a_2 = 14 \cdot i & \text{and} & j(E_{a_2}) = 422 \\
\phi_2 : E_{a_2} \to E_{a_3} & \text{with} & a_3 = 376 & \text{and} & j(E_{a_3}) = 242
\end{array}
$$

Bob's shared secret computation is depicted in Figure 3.15. Revisiting 3.14 we see that both Alice and Bob landed on a curve with j-invariant $42i + 141$.

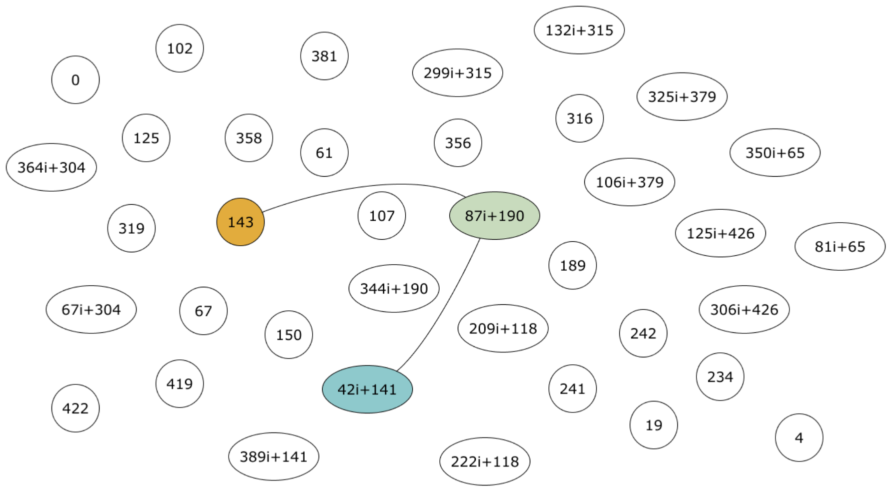


**Figure 3.15:** Bob's shared secret computation. He starts on the curve from Alice's public key with j-invariant 143. He then uses his secret key to compute the remaining walk to the node with j-invariant $42i + 141$. This is the shared secret.

| ID | p | $\ell_{Alice}^{max}$ | $\ell_{Bob}^{max}$ | Key gen | | Shared secret | |
|---|---|---|---|---|---|---|---|
| | (bits) | (integer) | (integer) | Alice | Bob | Alice | Bob) |
| 1 | 253 | 76667 | 51193 | 11.82s | 14.14s | 10.41s | 13.25s |
| 2 | 255 | 47353 | 38201 | 8.065s | 14.11s | 6.419s | 13.02s |
| 3 | 255 | 2 | 51040879 | 3.925s | 149.58s | 3.791s | 94.42s |
| 4 | 247 | 7901 | 7621 | 10.63s | 11.01s | 9.715s | 9.425s |
| 5 | 247 | 53 | 709153 | 9.892s | 17.14s | 9.468s | 13.11s |
| 6 | 247 | 37 | 745309897 | 11.68s | 196.5s | 11.27s | 123.3s |

**Table 3.2:** Table for how long it takes to run through our implementation of the B-SIDH protocol. The examples are labeled by ID and the full primes can be seen in Table 3.3. The third and fourth columns are the biggest integer in Alice and Bob factors respectively. Columns 5-8 are the timings for the key generation and the derivation of the shared secret.

### 3.7.6 B-SIDH Running Time

As a next step, we shall execute the B-SIDH protocol using our own implementation to measure the duration of key generation and shared secret derivation. It's important to highlight that our aim here is not to strive for optimal speed, but rather to understand the performance characteristics of different prime numbers within this context. As preiously stated in 3.7.5, operating over $\mathbb{F}_{p^4}$ and not use $x$-only arithemtic will significantly slow the performance.

The selection of primes used in this timing excercise has been drawn from Costello's paper on B-SIDH [Cos20, Table 1]. This selection of primes gives a good varying degree of smooth order, for both Alice and Bob.

In Table 3.2 we see the different running times for the different examples. The first thing to notice is that the running time is heavily dependent on the degree of the isogeny. The two most clear examples are the primes with IDs 3 and 6. In both examples, Alice's biggest isogeny degree is a relatively small number, while Bob is burdened with a high degree of isogeny.

A second observation is that there is a tradeoff, either accelerate one part, in this case, Alice, and burdened the other, or more evenly distribute the computational time.

A final thing to note is that the time for deriving the shared secret is less than the time for generating the public keys. As previously mentioned in the SIDH and B-SIDH examples, during deriving the shared secret we do not have to push the basis points through the isogeny. Which is a notable difference in running time.

| ID | Prime |
|---|---|
| 1 | 6076565381793920588579541156299400888300934131913673683193138303184607223137608869474563035671800932229002362355713 |
| 2 | 11402780996313137804419565692258934141207562497476991733713707020990899136527 |
| 3 | 53380256806900813486663140872422666278845625642838258109389995943373349847039 |
| 4 | 49763539558022470300850808459551889029922028956791596168707089459100476506111 |
| 5 | 124911187139035370912055024190553051978425375284430516021455114291199999999 |
| 6 | 188098835761489939757482570291811148273499283258225940944664269318258687 |
| 7 | 160301004300541798749117895782440681403932506173733050251729195751312457727 |
| 8 | 981557143716422697151123767627082955525245092564873268976355659484863281249 |

Table 3.3: The full primes used in Table 3.2 labeled by ID.

Chapter 4

# Castryck and Decru's Key Recovery Attack

In this chapter, we will explore Castryck and Decru's key recovery attack. We will start by delving into the theory of genus 2 curves and their Jacobians, followed by a high-level overview of the attack. We will then discuss Kani's theorem and the construction of auxiliary isogenies before providing a more detailed explanation.

## 4.1 Hard Problems are Difficult to Find

The security of cryptographic schemes often hinges on the formulation of the underlying problems. If these problems can be solved with high probability and within a feasible timeframe, the scheme can be considered broken. To gain a better understanding of the attack on SIDH, it is useful to first examine the problems upon which its security is based. In this section, we will explore the well-known Diffie-Hellman problems and formulate two analogous problems specific to SIDH.

### 4.1.1 Computational and Decision Diffie-Hellman Problem

The Computational Diffie-Hellman (CDH) problem and the Decision Diffie-Hellman (DDH) problem are two related cryptographic problems that pertain to the security of the Diffie-Hellman key exchange protocol. Both problems are concerned with the difficulty of solving specific mathematical challenges, and their hardness assumptions are the foundation for the security of many cryptographic schemes.

The CDH problem can be formulated as follows: given a cyclic group $G$, a generator $g$ of $G$, and the elements $g^a$ and $g^b$, where $a$ and $b$ are random secret integers, compute the shared secret $g^{ab}$. The CDH problem is considered hard since there is no known efficient algorithm for solving it. The security of the previously mentioned, Diffie-Hellman key exchange, relies on the assumption that the CDH problem is difficult to solve.

The DDH problem, on the other hand, deals with distinguishing between two types of elements in the cyclic group $G$. Given a cyclic group $G$, a generator $g$ of $G$, the elements $g^a$, $g^b$, and an element $h$ of $G$, the task is to decide whether $h$ is equal to the shared secret $g^{ab}$ or not. If it is computationally infeasible to distinguish between the case where $h = g^{ab}$ and the case where $h$ is a random element of $G$, the DDH problem is considered hard.

Clearly DDH is weaker than CDH because if we can solve CDH we know the answer to the DDH question with certainty. The hardness of the CDH and DDH problems is crucial for the security of many cryptographic schemes, as it serves as a foundation for the assumed difficulty of breaking the underlying mathematical structures.

The relationship between breaking the Decisional Isogeny Problem (DIP) and breaking the CDH problem lies in the fact that the solution to the DIP can be directly used to solve the CDH problem. Once an attacker obtains the exponent $x$ through DLP-breaking techniques, they can calculate $g^{ab}$ by raising $g^a$ to the power of $x$. This effectively means that breaking the DLP implies the ability to break the CDH problem as well. From this, we can create a chain $DDH \to CDH \to DLP$ where solving one on the right implies breaking the ones to the left.

### 4.1.2 Computational and Decisional Isogeny Problem

Similar to Diffie-Hellman, we can formulate one computational and one decisional problem related to isogenies and SIDH. We will define these two problems and explain how solving them can break SIDH.

The Computational Isogeny Problem (CIP) can be described as follows: given two elliptic curves $E$ and $E'$ over a field $K$, find an isogeny of degree $\ell^k$ between them. This problem is not limited to prime powers but also applies to isogenies with degrees that are products of many coprimes, as in B-SIDH. Analogous to the CDH problem, we consider the CIP problem to be hard if there is no known efficient algorithm for solving it.

An efficient algorithm for solving the CIP problem would enable the breaking of SIDH easily. In the SIDH protocol, both the starting curve $E$ and Bob's curve $E'$ are public, and finding an isogeny of degree $\ell^k$ between them would reveal Bob's secret isogeny.

The DIP involves deciding on the existence of an isogeny, rather than distinguishing between two types of elements, as in the DDH problem. We define the DIP problem as follows: given two elliptic curves $E$ and $E'$ over a field $K$, determine if there exists an isogeny of degree $\ell^i$ between them for $0 < i < k$. Although it is not as straightforward as the CIP problem, one can argue that solving the DIP problem would also break SIDH.

Let's consider an isogeny from $E$ to $E'$ of degree $\ell^k$ and assume we have an oracle that can flawlessly answer the DIP problem in a feasible amount of time. We can then easily construct the entire isogeny $\ell^k$ by concatenating $k$ isogenies of degree $\ell$:

$$E = \underbrace{E_0 \to E_1}_{\ell+1\text{ - options}} \to E_2 \to E_3 \to \ldots \to E_{k-1} \to E_k = E'$$

For the first isogeny in our chain, we have $\ell + 1$ options. By computing these isogenies, we can ask the oracle if there exists an isogeny of degree $\ell^{k-1}$ from $E_1 \to E_k$. With high probability, there will only be one correct answer, and we can again consider a new isogeny $E_1 \to E_2$ which also has $\ell + 1$ options. In this way, we can construct the entire $\ell^k$ isogeny.

The number of options we need to consider at each step depends on the size of $\ell$, but in SIDH, these values are small. Alice's secret isogeny is of degree $2^a$, meaning only three options, and Bob's secret isogeny is of degree $3^b$, with only four options. Therefore, constructing an oracle that can answer the DIP problem in a feasible timeframe would break SIDH.

Adjusting our focus slightly, let's now consider an extension of the DIP problem that is particularly pertinent to SIDH. In this scenario, along with our elliptic curves $E$ and $E'$, we also have access to the additional torsion points. Now, the DIP problem expands to determining the existence of an isogeny of degree $\ell^i$, $0 < i < k$ using these torsion points. Although this seems to be a minor shift, the implications are significant. As it turns out, being able to answer this adjusted DIP problem effectively breaks the security of SIDH and is precisely what Castryck and Decru accomplished.

## 4.2 Even More Curves

Earlier in section 2.3 when we defined elliptic curves, we said that the genus of an elliptic curve is always 1. In order to discuss the attack we need to broaden our set of curves and take a look at hyperelliptic curves, which has genus $g > 1$, and their Jacobians.

The following sections involve complicated mathematical concepts and are frankly beyond the capabilities of this thesis. Therefore in order to simplify the explanation and provide a high-level overview, we will gloss over many details related to the theory of hyperelliptic genus 2 curves and their Jacobians. This is also necessary because delving into the intricate mathematical definitions and concepts involved would require significant additional complexity and a much longer background theory. Our goal with this section is not to explain these concepts in great detail but rather to give a high-level overview and explain why we need this.

### 4.2.1 Hyperellptic Curves

A hyperelliptic curve is a curve of genus $g > 1$. Often genus 1 and genus 2 curves can visually be distinguished. Figure 4.1 shows two examples of a genus 1 and a genus 2 curve where we can observe, the number of donuts the curve creates represents the genus of the curve. While genus curves may not always admit a circle when drawn, a genus 1 curve can never look like the curve on the right in Figure 4.1.

More formally, if $char(K) \neq 2$, a hyperelliptic curve of genus 2 curves admits an equation of the form $\mathcal{C} : y^2 = f(x),$ with $deg(f) \in 5, 6$ and the discriminant is not 0 [Gal12, ch. 10]. The set of points of the curve $C$ is given by

$$\mathcal{C}(\bar{K}) = \{(u, v) \in \bar{K}^2 | v^2 f(u)\} \cup \begin{cases} \{\infty\} & \text{if } deg(f) = 5 \\ \{\infty_+, \infty_-\} & \text{if } deg(f) = 6, \end{cases}$$

A point $P \in \mathcal{C}(\bar{K})$ is $K$-rational of its coordinates are in $K$. The set of $K$-rational points is denoted by $\mathcal{C}(K)$.

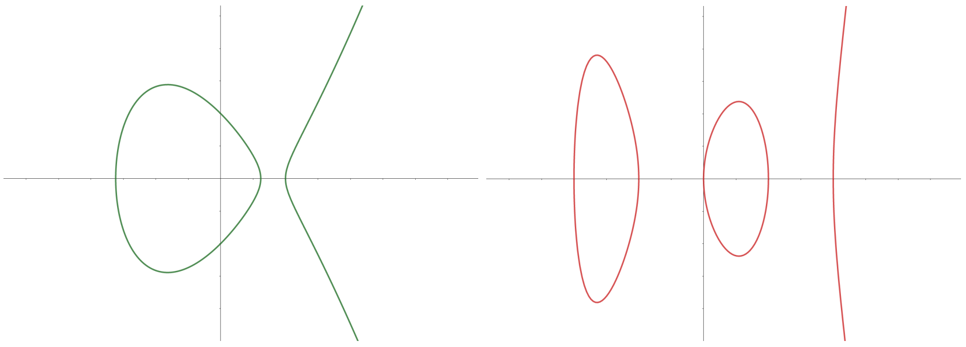


**Figure 4.1:** Examples of genus 1 and genus 2 curves. Visually these two curves are easily distinguishable by the number of circles the curve creates.

lets give an example with $\mathcal{C} : y^2 = x(x^2 - 1)(x^2 - 4)$ over $\mathbb{F}_7$. The curve has precisely 6 $\mathbb{F}_7$-rational points, the set of points is:

$$\mathcal{C}(\mathbb{F}_7) = \{\infty, (0, 0), (0, 1), (-1, 0), (2, 0), (-2, 0)\}$$

An important thing to note is that in contrast to elliptic curves, the set $\mathcal{C}(\bar{K})$ is not a group. Which means there is no group law on the set of points. Therefore we want another class we can work with namely the divisor class group of the curve.

A divisor on a curve $C/K$ is a formal linear combination of the curve's points (with integer coefficients). In other words, a divisor D is an expression of the form:

$$D = n_1 \cdot P_1 + n_2 \cdot P_2 + \ldots + n_k \cdot P_k$$

where $P_i$ are distinct points on the curve, and $n_k$ are integers associated with each point. The degree of a divisor is the sum of its coefficients, i.e., $deg(D) = n_1 + n_2 + \ldots + n_k$.

Divisors can be added and subtracted, meaning the *the divisor group* has an abelian group structure, denoted $Div_C$. The group of $K$-rational devisors is denoted by $Div_C(K)$.

Two divisors $D, D' \in Div_C$ on a curve C are said to be equivalent $(D \sim D')$ if their difference $D - D'$ is a principal divisor [MZoW+96, definition 36]. Being equivalent is an equal relation on the group of divisors. A principal divisor is a divisor that can be associated with a rational function f on the curve C.

In other words, given such $D$ and $D'$, then $D$ is equivalent to $D'$ if there exists an $f$ such that $D - D' = div(f)$, where $div(f)$ is the principal divisor associated with $f$.

The set of equivalence classes of divisors is denoted by $Pic_C(K)$ and forms a group known as the Picard group of the curve. The set of degree-0 divisors is denoted by $Pic^0_C(K)$.

### 4.2.2 The Jacobian of a Genus 2 Curve

Although the points on a genus 2 curve don't form a group, we can work with their Jacobians. The Jacobian $\mathcal{J}(\mathcal{C})$ of genus 2 curve $\mathcal{C}$ over $K$ is the 2-dimensional abelian variety with a special property called principal polarization, i.e. an isomorphism to its dual and is therefore considered a principally polarized abelian surface (p.p.a.s.). In general, these p.p.a.s. come in two types: *irreducible*, which are the Jacobians of genus-2 curves, and *reducible*, which are products of two elliptic curves. Jacobians have a group structure, such that over each field $K \subset L \subset \bar{K}$, we have $\mathcal{J}(\mathcal{C})(L) = Pic^0_{\mathcal{C}}(L)$.

We can represent elements of a Jacobian using the Picard group. This allows us to use degree-0 divisors on the curve to represent elements of the Jacobian. Take $\mathcal{J}(\mathcal{C})(K) = Pic^0_{\mathcal{C}}(K)$, and for any $R \in \mathcal{J}(\mathcal{C})(K) \backslash \{0\}$, there exist unique points

$P_1, P_2 \in \mathcal{C}(\bar{K})$ so that

$$R = [P_1 + P_2 - D_\infty], \text{ with } D_\infty = \begin{cases} 2 \cdot \infty & \text{if } deg(f) = 5 \\ \infty_+ + \infty_- & \text{if } deg(f) = 6 \end{cases} \tag{4.1}$$

For each element in the Jacobian, there exist two unique points on the curve C.

Representing elements in the Jacobian, may not always be ideal, and another way of presenting it is the *Mumford presentation.* Let $R \in \mathcal{J}(\mathcal{C})(K)$ and consider the presentation in 4.1, i.e. $R = [P_1 + P_2 - D_\infty]$ and for simplicity we write the points $P_1 = (u_1, v_1)$ and $P_2 = (u_2, v_2)$ as affine points. We define

$$\begin{aligned} a &= (x - u_1)(x - u_2) \in K[x] \\ b &= b_1 x + b_0, \text{ so that } b(u_1) = v_1 \text{ and } b(u_2) = v_2. \end{aligned}$$

Then (a, b) is called the Mumford presentation of R and we denote $R = J(a, b)$.

Let $y^2 = x(x^2 - 1)(x^2 - 4)$ over $\mathbb{F}_7$ and consider $(a, b) = (x^2 + x - 2, 0)$. This means $a = (x - 1)(x + 2)$ using the representation above, $u_1 = 1, u_2 = -2$ and $v_1 = b(1) = 0$, $v_2 = v(-2) = 0$. This means that the two points $P_1 = (-2, 0)$ and $P_2 = (1, 0)$ represents an element in the jacobian, $R = J(a, b) = [(1, 0) + (-2, 0) - 2 \cdot \infty]$

Just as with elliptic curves, are we particularly interested in the $N$-torsion subgroup. For an integer $N$, the $N$-torsion subgroup of a p.p.a.s is defined as $A[N] = \{P \in A | [N] \cdot P = 0\}$, where $A$ represents a p.p.a.s. Let $G \subset A[N]$ be a *maximal N-isotropic* subgroup, which is a technical term that needs to satisfy certain criteria [DK23, section 2.2]. Then there exist a unique p.p.a.s $A'$ together with an isogeny $\phi : A \to A'$ with kernel $ker(\phi) = G$. We will refer to the isogeny as an $(N, N)$-isogeny.

There are four different types of $(N, N)$-isogenies based on the domain and codomain of the isogeny. We can categorize the following isogenies:

i Generic case: $\phi : Jac(\mathcal{C}) \to Jac(\mathcal{C}')$.

ii Splitting case: $\phi : Jac(\mathcal{C}) \to E_1' \times E_2'$.

iii Gluing case: $\phi : E_1' \times E_2' \to Jac(\mathcal{C}')$.

iv Product case: $\phi : E_1 \times E_2 \to E_1' \times E_2'$.

In cryptography, the generic case is most common when working with p.p.a.s. however, we will later see that the Gluing and Splitting case is particularly interesting.

### 4.2.3 Richelot Isogenies

Richelot isogenies are $(2,2)$-isogenies between jacobians of genus-2 curves. These isogenies are of type generic case as in 4.2.2, where the kernel $\phi$ can be represented as the group $\ker\phi = \{0, P, Q, P+Q\}$. Without going into too much detail we are interested in the byproduct which is the domain curve $H$ which can be represented by the equation

$$H : y^2 = G_1(x)G_2(x)G_3(x)$$

The important thing to note is that $G_i$ are factors of the defining polynomials of the hyperelliptic curves, making the computation of $(2,2)$-isogenies fast. For further details and explicit formulas, we refer to the original paper [Smi06].

## 4.3 Castryck-Decru Attack

In this section, we will start with a high-level overview of the attack before delving into the specific components that particularly interest us. The complete attack is described in Castryck and Decru's paper [CD23b] and has been implemented in SageMath code [CD23a].

The attack aims to recover Bob's secret key $k_b$ by exploiting several properties of the SIDH protocol and parameters. There are three crucial pieces of information utilized in the attack:

1. *The exchanged data includes the images of the auxiliary points.*

   This data is included in the public keys making it always accessible.

2. *The secret isogeny $\phi_B$ has a fixed and known degree.*

   The degree of the secret isogeny is determined by the prime. In the case of SIKE, these are predetermined making it trivial to find the degree. Even if the two parties agree upon a secret prime and degree, it would be easy to find the degree given the public keys.

3. *The initial curve $E_0$ has a known endomorphism ring.*

   The starting curve $E_0$ in SIKE is always the same, making the endomorphism ring known. Maino and Martindale [MM22] and Robert [Rob23] have also described subexponential and polynomial-time attacks that do not require knowledge of the starting curve's endomorphism ring. This makes this item obsolete, however, for the version of the attack we will be using this is a requirement.

A crucial aspect of SIDH is that $\phi$ is not computed directly but as a composition of isogenies of degree 3. Recall the 3-isogeny graph in Figure 3.4; Bob computes a sequence of curves $E_0 \to E_1 \to E_2 \to \cdots \to E_B$ connected by 3-isogenies. The attack emulates SIDH's approach by recovering Bob's secret key digit by digit through a series of queries to an oracle, rather than computing the secret isogeny itself.

This oracle is precisely the oracle we discussed earlier in 4.1.2, who manages to answer the Decisional Isogeny Problem (DIP) in the case of SIDH [1]. Bob's secret isogeny $\phi : E_0 \to E_B$ is constructed by b isogenies of degree 3. By guessing one of the 3 isogenies from $E_0 \to E_1$ we can ask the oracle and the oracle determines if there exists an isogeny of degree $\ell^{b-1}$ from $E_1 \to E_B$ and returns true or false and transfer the torsion information. When we have made the correct guess, we can consider the next isogeny $E_1 \to E_2$.

Referring back to the 3-isogeny graph in Figure 3.4, each vertex has four connecting edges. Since one of the edges represents the path already taken, there are only three remaining steps, which we represent as one of the three ternary digits $0, 1, 2$. This implies that we need only two queries to the oracle at most. The path taken and the resulting ternary number can be converted to the secret key; for example, $2010211120_3 = 4216210_{10}$. Upon obtaining Bob's secret key, one can follow the SIDH protocol to compute the secret isogeny and shared j-invariant.

Castryck and Decru's true ingenuity lies in their ability to create an oracle that verifies whether the chosen path is correct and answers DIP using only public data. This oracle is known as the *glue-and-split* oracle.

Before exploring the attack in more detail, we need to briefly discuss three concepts: *Kani's theorem*, *auxiliary isogenies*, and the construction of Bob's secret key.

### 4.3.1 Kani's Theorem

Kani's theorem [Kan97] asserts that there is a special relationship between the groups of points of two specific elliptic curves, $C$ and $E$. This relationship, denoted as $\psi : C[N] \to E[N]$, helps us map between these two sets of points. The Castryck-Decru attack fundamentally relies on the existence of this $\psi$, along with another genus

[1] Castryck and Decru have not solved DIP for all instances, only in the case of SIDH where additional info is provided i.e. torsion points.

2 isogenies, or $(N, N)$-isogenies, which is mapping between two pairs of supersingular elliptic curve products.

Recall the isogeny diamond configuration in Figure 3.4. This isogeny diamond configuration is constructed by Alice's and Bob's secret isogenies as well as the isogeny What Kani's theorem tells us, is that it is possible to construct an explicit $(N, N)$-isogeny:

$$\Phi : E_B \times E_A \to E_0 \times E_{AB}, \quad \ker(\Phi) = \langle(\phi_B(P_B), \phi_A(P_A), \phi_B(Q_A), \phi_A(Q_A))\rangle$$

where in the case of SIDH, $N = 2^a + 3^b$.

Our next consideration is Bob's isogeny. We ask: can we devise a new isogeny and a corresponding curve to generate an alternative isogeny diamond configuration? Specifically, we wish to exploit Alice's torsion points to establish an *auxiliary isogeny*, denoted by $\gamma$, which could potentially reveal something about Bob's isogeny. How this auxiliary isogeny is constructed will be discussed in Section 4.3.2. Consider this new diamond configuration:

$$\begin{tikzcd}
E_A \arrow[d, "\phi'_B"] & E_0 \arrow[l, "\phi_A"'] \arrow[r, "\gamma"] \arrow[d, "\phi_B"] & \boxed{X_0} \arrow[d, "\phi'_B"] \\
E_{AB} & E_B \arrow[l, "\phi'_A"'] \arrow[r, "\gamma'"] & \boxed{X_B}
\end{tikzcd}$$

Let $E_0$ and $E_B$ represent the starting curve and Bob's curve, respectively. $E_B$ is the codomain of Bob's secret isogeny $\phi_B$ with degree $N_B$. The new auxiliary isogeny $\gamma : E_0 \to X_0$ should be constructed such that the degree is $N_A - N_B$.

A necessary condition is that $2^a > 3^b$. If this condition is not met, one could attack Alice's secret isogeny instead. However, this requires us to compute $(3, 3)$-isogenies which are significantly slower and pose practical challenges. To address this, one could make initial guesses and consider $N_B = 3^{b-\beta_i}$, ensuring that $2^a > 3^b$ holds.

Kani's theorem then tells us for some configuration of $\gamma$, then there is unique relation such that $\varphi : X_0[2^a] \to E_B[2^a]$ and a genus 2 $(2^a, 2^a)$-isogeny that maps between pairs of supersingular curves:

$$\Phi : X_0 \times E_B \to E_0 \times X_B, \quad \text{with} \ker(\Phi) \quad = \langle(\gamma(P_A), \phi_B(P_A), \gamma(Q_A), \phi_B(Q_B))\rangle$$

Castryck and Decru's exploitation of Kani's theorem does not directly compute Bob's secret isogeny. Instead, it is used as a computational tool for answering DIP, in the case of SIDH, using the torsion points. By considering the 4 different options of Bob's first isogeny $E_0 \to E_1$, we can use Kani's theorem to decide which option is correct, therefore learning something about Bob's secret isogeny.

### 4.3.2 Auxiliary Isogenies

In order to fulfill the isogeny diamond configuration in 4.3.1 we need to create auxiliary isogeny $\gamma$. An auxiliary isogeny is a supplementary isogeny that we aim to compute using the auxiliary points in the SIDH protocol. We seek an auxiliary isogeny of degree $N = N_A - N_B$. However, without extra information, constructing this isogeny is not straightforward. This is where one of the critical pieces of information comes into play, specifically item 3. In SIKE, the starting curve is $E_0 : y^2 = x^3 + 6x^2 + x$, for which the endomorphism ring, $End(E_0)$, is known. In particular, this curve has endomorphism $2i$, which satisfies $(2i)^2 = [-4]$.

With the known endomorphism, we can construct an isogeny of degree $N$ whenever we can represent the degree in the form $N = u^2 + 4v^2$, which is feasible if all prime factors of the form $q = 4k + 3$ have an even exponent in $N$. When this condition is met, we can construct the $N$-isogeny $\gamma = [u]x + [v] \cdot 2i$.

### 4.3.3 Constructing Bob's Secret Key

Recall the construction of the secret generator in SIDH, during the key generation in 3.5.1, which involves the public generators $P_B, Q_B$ of $E_0[3^b]$ and a secret integer $k_B \in [0, 3^b\rangle$. The kernel of $\phi$ is given by $(P_B + [k_B]Q_B)$. Bob's secret integer can be represented as:

$$k_B = k_1 + k_2 3^{\beta_1} + \cdots + k_r 3^{\beta_{r-1}}, \qquad k_i \in [0, 3^{\beta_i - \beta_{i-1}} - 1)$$

When we say we are guessing Bob's secret integer digit by digit, we are guessing the $k_i$ values. For $k_1$, we can observe that:

$$\ker \phi_1 = \langle 3^{b-\beta_1} P_B + k_1 3^{b-\beta_1} Q_B \rangle \tag{4.2}$$

For $k_2$, we have:

$$\ker \phi_2 = \langle 3^{b-\beta_2} P_B + (k_1 + k_2 3^{\beta_1}) 3^{b-\beta_2} Q_B \rangle$$

For $k_i$, we have:

$$\ker \phi_3 = \langle 3^{b-\beta_i} P_B + (k_1 + k_2 3^{\beta_1} + \cdots + k_i 3^{\beta_{i-1}}) 3^{b-\beta_i} Q_B \rangle$$

### 4.3.4 The Glue and Split Oracle

The glue and split oracle takes two supersingular elliptic curves as input and determines whether or not it takes us to a product of two elliptic curves. The oracle can be summarized in three steps, and we will explain each step in a way to give further intuition of how the attack works. For further proofs and details, we refer to the original paper [CD23b, p. 8.1].

In the first step, the oracle computes a $(2, 2)$-isogeny of the product of the two curves $X_0$ and $E_B$ by a subgroup $G = \langle (P_A, P_B), (Q_A, Q_B) \rangle$, by gluing them into the Jacobian, as the gluing case in 4.2.2.

$$\text{Step1} : X_0 \times E_B \to Jac(\mathcal{C}_1)$$

The next step is computing $a - 2$ $(2, 2)$-isogenies between jacobians of genus two curves. These isogenies are called Richelot isogenies where each kernel is a subgroup generated by a pair of order two elements of the Jacobian.

$$\text{Step2} : \underbrace{Jac(\mathcal{C}_1) \to Jac(\mathcal{C}_2) \to \ldots \to Jac(\mathcal{C}_i)}_{a - \alpha_i - 2}$$

It's in the last step where our oracle determines if we made the right guess for $k_i$. If the last $(2, 2)$-isogeny takes us back to a product of elliptic curves, we made the right guess. Practically this is done by verifying whether or not the determinant $\delta = 0$ vanishes which happens if and only if the codomain is a product of elliptic curves instead of the Jacobian of a genus two curve.

$$\text{Step3} : Jac(\mathcal{C}_i) \to E_0 \times X_B$$

Our final $(N, N)$-isogeny chain will look like the following:

$$\underbrace{X_0 \times E_B \to Jac(\mathcal{C}_1) \to Jac(\mathcal{C}_2) \to \ldots \to Jac(\mathcal{C}_i) \to E_0 \times X_B}_{a - \alpha_i}$$

### 4.3.5 Attacking Algorithm

We can outline the attack process as a six-step algorithm, with each iteration denoted by $i$. Keep in mind that this presentation may not be the most optimal approach, as some steps or iterations could allow for shortcuts. Nevertheless, this representation captures the attack's fundamental structure, making it easier to comprehend.

**Step 1:** Choose $\beta_i \geq 1$ minimal such that there exists some $\alpha_i \geq 0$ for which $c_i = 2^{a-\alpha_i} - 3^{b-\beta_i}$ is of the form $u_i^2 + 4v_i^2 > 0$

**Step 2:** Make a guess for $k_i \in \{0, 1, 2\}$ [2]

**Step 3:** Use $u_i$ and $v_i$ to construct an auxiliary isogeny $\gamma = [u_i] + [v_i] \cdot 2i$

**Step 4:** Determine the $3^{\beta_i}$-isogeny $\tau : E_{start} \to C_i$ with kernel $\gamma(\ker \kappa_i)$, where $\ker \phi_i$ is as in equation 4.2.

**Step 5:** Compute the points $P_{C_i}, P_{Q_i} \in C_i$ where $P_{C_i} = 2^{\alpha_i} k_i \gamma(P_a)$ and $Q_{C_i} = 2^{\alpha_i} k_i \gamma(Q_a)$

**Step 6:** Check whether or not the subgroup $\langle (P_{C_i}, 2^{\alpha_i} P), (Q_{C_i}, 2^{\alpha_i} Q) \rangle$ splits.

### 4.3.6 Step Size

When talking about step size we are referring to the gap between the integers $\beta_0, \beta_1, \beta_2, \ldots, \beta_r = b$. This gap should be as small as possible as it limits the number of possible guesses in each iteration. For a one-step iteration, we only have 3 possible guesses but for a two-step iteration, we need $3 \cdot 3 = 9$ possible guesses.

A necessary condition for keeping one step is ensuring that $b - \beta_i$ is odd. If $b - \beta_i > 0$ is odd then $c_i = 2^{a-\alpha_i} - 3^{b-\beta_i} \equiv 3 \bmod 4$ cannot be of the form $u_i^2 + 4v_i^2$.

### 4.3.7 Alternative Visual Representation of the Attack

Consider the rooted tree graph in Figure 4.2. With Castryck and Decru's attack, we want to find out the explicit path Bob took during his key generation. By starting at the root node we have three possibilities, given the step size is 1. If the step size is 2 we have to consider all the 9 possibilities by taking two steps down in the tree graph.

When taking the step we consult our oracle who tells us if we made the right step in the graph and lower the number of possibilities. In this thesis, we will then say we have *recovered* the isogeny. Visually this means we have found out which step in the graph is correct and which branch the destination node lies in.

[2]We only have to make a guess for two of digits, as if two fail we can assume the third is the correct digit.

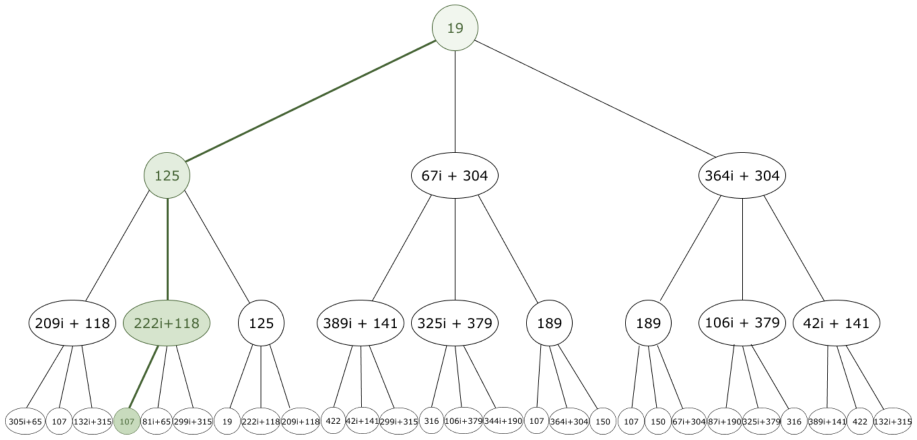


**Figure 4.2:** A visual representation of the path Bob takes during the key generation. The root node is the starting j-invariant both Alice and Bob start on while the internal nodes are the intermediate j-invariants Bob visits. The leaf nodes are 27 possible destinations Bob can land on for $p = 2^4 3^3$. The green nodes and edges are the path Bob takes during the example 3.5.2.

## 4.4 Attacking SIDH Toy Example

In this section, we will demonstrate an attack on SIDH using a toy example. By working with a simplified version of the problem, we aim to provide a clearer understanding of the attack's mechanisms and the key concepts involved. This hands-on approach will offer valuable insights into the protocol's vulnerabilities and the strategies Castryck and Decru employed to break SIDH. Keep in mind that the essential ideas and techniques used in the attack remain the same, regardless of the scale or complexity of the problem. Increasing the scale would merely result in more iterations of the same core attack process, further emphasizing this toy example's relevance and educational value.

We will attack the example showcased in Section 3.5.2, where Bob's secret key is $k_b = 3$. Remember that we recover Bob's secret integer in ternary form, digit by digit, resulting in $k_b = [0, 1, 0]$ with the first digit being the least significant bit. We will go through each iteration of the algorithm in Section 4.3.5, explaining each step and providing further analysis where necessary.

**Setup** During the attack we need Alice's basis points to create the auxiliary isogeny and Bob's basis points to check if it splits. Also recall $p = 2^4 3^3 - 1$ where $a = 4$

and $b = 3$.

$$P_A := (70 \cdot i + 332, 371 \cdot i + 191) \quad \text{and} \quad Q_A := (269 \cdot i + 403, 302 \cdot i + 3)$$
$$P_B := (i + 420, 112 \cdot i + 230) \quad \text{and} \quad Q_B := (265 \cdot i + 329, 377 \cdot i + 124)$$

We will initiate a list of the ternary digits we have recovered $sk_{Bob} = [-, -, -]$

**Iteration 1** We begin by guessing the first digits of Bob's secret key.

Step 1 We find that for $\beta_1 = 2$ we have a $\alpha_1 = 0$ for which $C_1 = 2^{4-0} - 3^{3-2} = 13$. This can be written of the form $C_1 = u_i^2 + 4v_i^2 = 3^2 + 4(-1)^2 = 5$

*Our first step size is $\beta_1 = 2$ which means we will make guesses for two of Bob's digits. This means we have $3 \cdot 3 = 9$ possibilities in the first iteration or we have to consider all the possible nodes two steps down in Figure 4.2*

Guess 1 $k_1, k_2 = \{0, 0\}$

Step 3 We can construct our auxiliary isogeny $\gamma = [3] + [-1] \cdot 2i$

Step 4 The $3^2$-isogeny $\tau_1 : E_{start} \to C_1$ with kernel $\gamma(\ker \kappa_1)$, where $\ker \kappa_1 = 3^{3-2} P_B + (0 \cdot 3^0 + 0 \cdot 3^1) \cdot 3^{4-1} Q_B$

Step 5 Compute the points $P_{C_1}, Q_{C_1} \in C_1$ where $P_{C_1} = 2^0 \tau_1 \gamma(P_A)$ and $Q_{C_1} = 2^0 \tau_1 \gamma(Q_A)$

*We have created a new curve $C_1$ with $a_0 = (42 * i + 420)$ and the two new points are: $P_{C_1} = (53 * i + 276, 171 * i + 405)$ and $Q_{C_1} = (51 * i + 37, 410 * i + 26)$*

Step 6 Check whether or not the subgroup $\langle (P_{C_1}, 2^{\alpha_0} P_B), (Q_{C_i}, 2^{\alpha_0} Q_B) \rangle$ splits.

*We want to glue the curves $C_1$ and $E_{start}$ into the jacobian of a genus 2 curve, via the (2,2)-subgroup provided above. Going through step 1 and 2 in 4.3.4 our isogeny chain will look like:*

$$C_1 \times E_{start} \to Jac(C_2) \to Jac(C_3)$$

*from the last Jacobian we will check if it splits, by determining the determinant. The determinant does not vanish which means it does not split, and we made the wrong guess.*

Guess 2 $k_1, k_2 = \{1, 0\}$

Step 3 We can construct our auxiliary isogeny $\gamma = [3] + [-1] \cdot 2i$

Step 4 The $3^2$-isogeny $\tau_2 : E_{start} \to C_2$ with kernel $\gamma(\ker \kappa_2)$, where $\ker \kappa_1 = 3^{3-2} P_B + (1 \cdot 3^0 + 0 \cdot 3^1) \cdot 3^{4-1} Q_B$

Step 5 Compute the points $P_{C_2}, Q_{C_2} \in C_2$ where $P_{C_2} = 2^0 \tau_2 \gamma(P_A)$ and $Q_{C_2} = 2^0 \tau_2 \gamma(Q_A)$

Step 6 Check whether or not the subgroup $\langle(P_{C_2}, 2^{\alpha_0}P_B), (Q_{C_2}, 2^{\alpha_0}Q_B)\rangle$ is splits.
*We glue the two curves $C_2$ and $E_{start}$ into the jacobian of a genus 2 curve, via the (2,2)-subgroup provided above. The determinant does not vanish which means it does not split, and we made the wrong guess.*

Guess 3 $k_1, k_2 = \{2, 0\}$

Step 3 We can construct our auxiliary isogeny $\gamma = [3] + [-1] \cdot 2i$

Step 4 The $3^2$-isogeny $\tau_3 : E_{start} \to C_3$ with kernel $\gamma(\ker\kappa_3)$, where $\ker\kappa_3 = 3^{3-2}P_B + (2 \cdot 3^0 + 0 \cdot 3^1) \cdot 3^{4-1}Q_B$

Step 5 Compute the points $P_{C_3}, Q_{C_3} \in C_3$ where $P_{C_3} = 2^0\tau_3\gamma(P_A)$ and $Q_{C_3} = 2^0\tau_3\gamma(Q_A)$

Step 6 Check whether or not the subgroup $\langle(P_{C_3}, 2^{\alpha_0}P_B), (Q_{C_3}, 2^{\alpha_0}Q_B)\rangle$ is splits.
*We glue the two curves $C_3$ and $E_{start}$ into the jacobian of a genus 2 curve, via the (2,2)-subgroup provided above. The determinant does not vanish which means it does not split, and we made the wrong guess.*

Guess 4 $k_1, k_2 = \{0, 1\}$

Step 3 We can construct our auxiliary isogeny $\gamma = [3] + [-1] \cdot 2i$

Step 4 The $3^2$-isogeny $\tau_4 : E_{start} \to C_1$ with kernel $\gamma(\ker\kappa_4)$, where $\ker\kappa_4 = 3^{3-2}P_B + (0 \cdot 3^0 + 1 \cdot 3^1) \cdot 3^{4-1}Q_B$

Step 5 Compute the points $P_{C_4}, Q_{C_4} \in C_4$ where $P_{C_4} = 2^0\tau_4\gamma(P_A)$ and $Q_{C_4} = 2^0\tau_4\gamma(Q_A)$

Step 6 Check whether or not the subgroup $\langle(P_{C_4}, 2^{\alpha_0}P_B), (Q_{C_4}, 2^{\alpha_0}Q_B)\rangle$ is splits.
*We glue the two curves $C_4$ and $E_{start}$ into the jacobian of a genus 2 curve, via the (2,2)-subgroup provided above. The determinant does indeed vanish which means it does split, and we made the right guess and we can update our recovered digits $sk_{Bob} = [0, 1, -]$.*

**Iteration 2** Given that we have managed to retrieve two out of the three digits of Bob's key $sk_{Bob} = [0, 1, -]$, it is faster to brute force the last digit by computing the isogeny from $E_{start}$

Guess 1 $k_3 = 0$

- *bobskey* $= (0 \cdot 3^0 + 1 \cdot 3^1 + 0 \cdot 3^2) = 3$
- The $3^3$-isogeny $\phi : E_{start} \to E'$ with $ker(\phi) = P_B + bobskey \cdot Q_B$
- Check if $j(E') = j(E_B)$
  *For $k_3 = 0$, $j(E') = j(E_B)$ is indeed true which means we have the last correct digit.*

**Finish** We have successfully attacked SIDH by recovering Bob's key digit by digit $sk_{Bob} = [0, 1, 0] = 3$.

## 4.5 Other Attacks on SIDH

Other attacks have been inspired by or further developed by the Castryck-Decru attack. We will briefly take a look at a few of these.

### 4.5.1 Generalization by Martindale, Maino, and Robert

During the development of Castryck and Decru's attack, Martindale and Maino [MM22] discovered an attack that makes no use of any special endomorphisms on $E_0$. The work was done independently but inspired by an earlier joint project. The attack has subexponential complexity which is slower than Castryck and Decru's attack, however significantly reduces the security of SIDH and SIKE in the case of an unknown endomorphism ring on the starting curve.

The main idea behind the attack is constructing an elliptic curve $E$, an isogeny $\phi : E \to E_0$, and a polarized isogeny $\Phi$ originating from the abelian surface $E \times E_A$, where $E_A$ is the codomain of Alice's secret isogeny. $E \times E_A$ is constructed such that one of its components reveals the dual of Alice's secret isogeny $\phi_A : E_0 \to E_A$.

Robert [Rob23] dramatically improved the theoretical result by giving a polynomial-time attack with an arbitrary starting curve.

### 4.5.2 Direct Computation

Oudompheng [OP22a], and Wesolowski [Wes] have all observed that Kani's machinery allows for direct key recovery. Which is significantly faster than Castryck and Decru's decisional approach. After Oudompheng's work, a modification to the Castryck-Decru attack has been introduced. After the first digit has been successfully guessed, the secret isogeny can directly be calculated from the result of the (2,2)-isogeny chain.

# Chapter 5 Attacking B-SIDH

In this chapter, we discuss to what extent Castryck and Decru's attack hold in the case of B-SIDH. We will begin by discussing what properties of B-SIDH require modifications to the attack. We present a new construction of Bob's secret key and show how a method for finding how many queries that are required to the oracle. Then the attacking algorithm will be presented with three examples whereas one is a toy example.

In section 5.3 we discuss how we can speed up the attack by exploring various techniques and optimizations. We conclude the chapter by discussing if B-SIDH can be considered broken.

## 5.1 Preparing to Attack B-SIDH

In this section, we will shift our focus to attacking B-SIDH. To do this, we need to update some terminology and provide further explanations. We will use $M$ to refer to Alice's order, or the party who works over $p+1$, and $N$ to refer to Bob's order, or the party who works over $p-1$. The variables $a$ and $b$ will now denote the number of factors, including the multiples, in $M$ and $N$, respectively.

We will begin by discussing some of the differences and adaptations required to attack B-SIDH. As discussed in Section 3.7, most of the differences in B-SIDH arise from the use of twisted torsion for both parties. Consequently, this introduces challenges when carrying out the attack. We will explore the implications of Alice and Bob using twisted torsion and the resulting consequences.

### 5.1.1 Bob's Twisted Torsion

When Bob employs a varying degree of torsion, it follows that his secret integer can no longer be represented as a list of ternary digits. The solution to this problem will be discussed in Section 5.1.6, and consequently, we can no longer make the same guesses of $k_i \in [0, 1, 2]$. Our guesses will now be dynamic and correspond to the torsion part we are recovering.

Generalizing Bob's order to $N = \ell_1^{k_1} \ell_2^{k_2} \dots \ell_i^{k_i}$, when trying to find a valid $c_i$ in step 2, we apply the same logic, yielding $c_i = M - \ell_1^{k_1 - \beta_{1i}} \ell_2^{k_2 - \beta_{2i}} \dots \ell_i^{k_i - \beta_{ki}}$, then $\beta_i = \beta_{1i} + \beta_{2i} + \dots \beta_{ki}$. In other words, we aim to remove as few factors in $N$ as possible, with $\beta_i$ representing the number of factors we remove. Given that we only need to remove one factor, we make the guess for $k_i \in [0, \ell_i\rangle$. If we need two factors, then $k_i$ will be the tuple $k_i = (k_1, k_2)$, where $k_1 \in [0, \ell_1\rangle$ and $k_2 \in [0, \ell_2\rangle$.

Let $\Lambda$ denote the order we have recovered. The next challenge arises when trying to find the $\Lambda$-isogeny $\tau : E_{start} \to C_i$ in step 4. The isogeny $\tau$ can no longer be computed as a chain of 3-isogenies but rather as a chain of isogenies of the degree $\Lambda$. Apart from some changes in the code, this is not a major problem, as finding isogenies for a given degree with a given kernel is a straightforward task for small or smooth enough order.

### 5.1.2 Alice's Twisted Torsion

How efficiently we can attack B-SIDH heavily depends on how our prime $p$ is chosen or specifically how Alice's order factors. We will revisit and use the three categories we introduced in section 3.7.2, in order to explain different properties and the challenges that arise. The type of isogenies we need to compute is directly correlated to Alice's degree, due to Kani's theorem stating a unique genus 2 $(N, N)$-isogeny. The unique genus 2 $(N, N)$-isogeny can be computed as a chain of $(\ell, \ell)$-isogenies of Alice's factors.

#### Alice's order is $2^f$

When Alice's order is a power of two, we only need to compute $(2, 2)$-isogenies. Due to the work of Smith's thesis [Smi06] these isogenies, called Richelot isogenies, are efficient. Oudompheng and Pope, [OP22b], reimplemented this in SageMath using explicit formulas and some work around SageMath limitations. It is also worth mentioning Kunzweiler published an efficient implementation of $(2^n, 2^n)$-isogenies [Kun22]. However, this implementation is not utilized in the implementation of the attack in SageMath.

#### Alice's order is $2^f \cdot 3^e$

When Alice's order contains a power of threes, we need to compute (3,3)-isogenies. We will explore possibilities of $(3, 3)$-isogenies in 5.1.3.

**Alice's order is $\ell_1^{k_1} \cdot \ell_2^{k_2} \cdot \ldots \cdot \ell_i^{k_i}$**

When Alice's order has no restrictions and is the product of different coprime $\ell$, we need to compute $(\ell, \ell)$-isogenies. We will explore possibilities of $(\ell, \ell)$-isogenies in 5.1.4.

### 5.1.3 (3, 3)-Isogenies

We previously introduced a specific type of isogenies called Richelot isogenies, which are $(2, 2)$-isogenies between Jacobians of genus-2 curves. As discussed above, to attack B-SIDH, when Alice's order contains three powers, we need the ability to compute (3,3)-isogenies. Bruin, Flynn, and Testa [BFT14] provide the parametrization of genus 2 curves whose Jacobians have a (3, 3)-torsion subgroup with rational generators. They also supply the corresponding isogeny formula. However, these formulas require over 37,500 multiplications to evaluate a single isogeny at a point.

Following more recent work by Decru and Kunzweiler [DK23], these formulas have been simplified and reduced by 94%, offering a significantly faster and more efficient computation of $(3^n, 3^n)$-isogenies. Moreover, they deduce explicit formulas for evaluating splitting and gluing as the two cases in 4.2.2. Their implementation managed to retrieve Alice's secret isogeny in 11 seconds for the SIKEp751 parameters, which represented the highest security level.

Unfortunately, this implementation is only available in Magma and not SageMath, which is required for this thesis. Translating this into the SageMath would be outside the scope of this thesis. Oudompheng and Pope [OP22b, Section 3.3] explains that translating Castryck and Decru's attack into SageMath indeed posed several challenges that necessitated assistance from many within the cryptographic community. Comparable challenges may arise for anyone undertaking the task of translating the implementation.

The time required for translating Decru and Kunzweiler's implementation can only be speculated. Nevertheless, this is certainly an area for future work and is well worth further investigation. Resulting this task lies outside of the scope of this thesis.

### 5.1.4 ($\ell$, $\ell$)-Isogenies

Finding a way of computing ($\ell$, $\ell$)-isogenies would be the most appropriate approach. However, this approach does not come without its own challenges. Computing $(\ell, \ell)$-isogenies requires huge polynomials and takes a lot of time unless one finds explicit formulas like Richelot isogenies in 4.2.3 or Bruin, Flynn, and Testa [BFT14]. When the degree of the isogeny increases so does the polynomials.

Cosset and Robert [CR11] explore and give a method for computing $(\ell,\ell)$-isogenies. Their way is by converting Momford coordinates of Jacobians of genus-2 curves into another representation. In combination with other provided algorithms provides a method for computing $(\ell,\ell)$-isogenies polynomial time.

To be able to see if we can combine this with Richolot isogenies, we need to see the definition of a $(\ell,\ell)$-isogeny. To form an $(\ell,\ell)$-isogeny we need a *symplectic basis* which is a set of four points $(R_1, R_2, S_1, S_2)$ in the Jacobian In the following definition, the term symplectic basis will be used which will be explained after the definition.

**Definition 5.1.** Let $(R_1, R_2, S_1, S_2)$ be a symplectic basis for $J(C)[\ell]$, then for any $a, b, c \in \mathbb{Z}/\ell\mathbb{Z}$, the group

$$G = \langle R_1 + aS_1 + bS_2, R_2 + bS_1 + cS_2 \rangle$$

defines a $(\ell,\ell)$-isogeny.

Finding a symplectic basis $J(C)[\ell]$ requires us to identify four points in $J(C)[\ell]$ that form a group basis over $\mathbb{Z}/\ell\mathbb{Z}$. Meaning every point in $J(C)[\ell]$ can be written uniquely as a $\mathbb{Z}/\ell\mathbb{Z}$-linear combination of the basis points.

Recall an element in $J(C)$ is represented by two points $P_1, P_2 \in C(K)$. After computing a Richelot isogeny we have four points on $C$, which is not enough to fully represent a symplectic basis.

Even if we managed to combine these approaches a big factor would be the time it requires to compute big $(\ell,\ell)$-isogenies. Cosset and Robert [CR11, Section 5.6] together with Bisson present an implementation that requires almost two hours for computing an isogeny of degree $\ell = 1321$ over a small finite field of 16-bits and without any field extensions. After testing with their provided examples, computing a single (3,3)-isogeny takes 0.448 seconds. Comparing this to computing a Ricghelot isogeny, a (2,2)-isogeny, takes on average 0.00504 seconds, which is almost 90 times slower.

This approximation of time is only evaluating a single isogeny and doesn't take into account the time its takes for converting Mumford coordinates of Jacobians to theta coordinates. Practically this may be even slower than estimated.

While the work by Cosset and Robert gives us the isogenies of the generic type, we still need gluing of elliptic curves and splitting of Jacobians. Castryck and Decru [CD23b, Section 11.1] refer to work by Kuhn [Kuh88] and also point out away from

$\ell = 2, 3$ there is no known straightforward decision algorithm to verify whether an $(\ell, \ell)$-subgroup of a given Jacobian of a genus 2 curve results in a product of elliptic curves.

A possible solution to the gluing problem is ensuring that Alice's order contains a factor of 2, which is most often the case. Then always compute the (2,2)-isogeny when gluing elliptic curves, since this is known and efficient. For the splitting case, Castryck and Decru mention the seemingly easiest way, is to see if theta constants fail to create a genus 2 curve.

**Remark 5.2.** While there are theoretical solutions to these issues, as far as we know, there exist no implementations we can use. Implementing these would require an extensive amount of research and is something we have to leave for future work. *Consequently, we have to limit ourselves to only attack primes where Alice's order is a power of two.*

### 5.1.5 Can We At Least Lower the Security Requirements?

One might naturally wonder if it would be possible to relax the security constraints by merely computing $(2, 2)$-isogenies, and then attempting to guess the remaining isogenies through a brute-force approach. Regrettably, this method proves unfeasible. As an example, consider the prime number discussed earlier in section 3.3, where Alice's order is given by $N = 2^{188} \cdot 11 \cdot 17 \cdot 29 \cdot 73 \cdot 193$. In an attempt to guess one of Bob's digits and consult the glue-and-split oracle, it becomes necessary to compute the entire $(N, N)$-isogeny chain to ascertain whether or not it splits. Therefore, we cannot confirm the correctness of the guessed digit without the ability to compute isogenies of higher degrees.

### 5.1.6 Representing Bob's Secret Key with the Chinese Remainder Theorem

As previously mentioned in Section 5.1.1, we can no longer represent Bob's secret integer as a list of ternary digits. In B-SIDH, Bob's order is the product of different coprime numbers, including multiples. Therefore, we need a way to represent Bob's secret integer that is not dependent on a fixed degree.

Taking into account that all our factors or divisors are pairwise coprime and the digit we are guessing lies in the range $0 \leq k_i < \ell_i$, this problem can be addressed using the Chinese Remainder Theorem. If we know every remainder $k_i$, then there exists a unique integer $K$, such that $0 \leq K < N$, and $K \equiv k_i (\bmod\, \ell_i^{\ell_i})$.

Given Bob's order $M = n_1 \cdot n_2 \ldots \cdot n_i$, where $n_1 \cdot n_2 \ldots \cdot n_i$ are integers greater than 1 and all $n_i$ are pairwise coprime, we can represent Bob's secret integer as follows:

$$K = k_1 \cdot \frac{M}{n_1} \cdot y_1 + k_2 \cdot \frac{M}{n_2} \cdot y_2 + \ldots + k_i \cdot \frac{M}{n_i} \cdot y_i$$

where

$$\begin{array}{ll} k_1 (\mathrm{mod}\ n_1) & \frac{M}{\ell_1} y_1 \equiv 1 (\mathrm{mod}\ \ell_1) \\ k_2 (\mathrm{mod}\ n_2) & \frac{M}{\ell_2} y_2 \equiv 1 (\mathrm{mod}\ \ell_2) \\ \vdots & \\ k_i (\mathrm{mod}\ n_i) & \frac{M}{\ell_i} y_i \equiv 1 (\mathrm{mod}\ \ell_i) \end{array}$$

If one of Bob's factors has multiplies, we can solve it as follows. Take $M = 4095 = 3^2 \cdot 5 \cdot 7 \cdot 13$, we can represent Bob's integer as:

$$k_b = (k_{11} + k_{12} 3) \cdot \frac{M}{3^2} \cdot y_1 + k_2 \cdot \frac{M}{5} \cdot y_2 + k_3 \cdot \frac{M}{7} \cdot y_3 + k_4 \cdot \frac{M}{13} \cdot y_4$$

where

$$\begin{array}{lll} K \equiv k_1 (\mathrm{mod}\ 3^2) & \frac{M}{\ell_1} y_1 \equiv 1 (\mathrm{mod}\ 3^2) & y_1 = 2 \\ k_2 (\mathrm{mod}\ 5) & \frac{M}{\ell_2} y_2 \equiv 1 (\mathrm{mod}\ 5) & y_1 = 4 \\ k_3 (\mathrm{mod}\ 7) & \frac{M}{\ell_3} y_3 \equiv 1 (\mathrm{mod}\ 7) & y_1 = 2 \\ k_4 (\mathrm{mod}\ 13) & \frac{M}{\ell_4} y_4 \equiv 1 (\mathrm{mod}\ 13) & y_1 = 9 \end{array}$$

We can observe that for our first iteration, we get

$$\ker \phi_1 = \langle \Lambda \cdot P_B + (k_1 \cdot \frac{M}{n_1} \cdot y_1) \cdot \Lambda \cdot Q_B \rangle$$

After finding $k_1$, we can proceed to find $k_2$

$$\ker \phi_2 = \langle \Lambda \cdot P_B + (k_1 \cdot \frac{M}{n_1} \cdot y_1 + k_2 \cdot \frac{M}{n_2} \cdot y_2) \cdot \Lambda \cdot Q_B \rangle$$

The general equation $k_i$ is:

$$\ker \phi_i = \langle \Lambda \cdot P_B + (k_1 \cdot \frac{M}{n_1} \cdot y_1 + k_2 \cdot \frac{M}{n_2} \cdot y_2 + \ldots + k_i \cdot \frac{M}{n_i} \cdot y_i) \cdot \Lambda \cdot Q_B \rangle \tag{5.1}$$

### 5.1.7 Number of Queries to Oracle

As presented in chapter 4 we make a guess for each digit of Bob's key, comparable to an efficient brute force attack. For SIDH one can easily calculate the upper bound of the total number of attempts we need to find Bob's key. For each digit we guess we need to at worst make two queries to our oracle, the digits $\{0, 1\}$, and if both fail, we can assume 2 is the correct digit. Our upper bound of queries will then be $\mathcal{O}(2b)$.

In B-SIDH we are in a much worse spot as our isogenies are of higher degrees. Bob's order can be generalized to $\ell_1^{k_1}\ell_2^{k_2}\ldots\ell_i^{k_i}$ and by applying the same logic the upper bound of queries needed, denoted by $\#Q$:

$$\#Q = k_1(\ell_1 - 1) + k_2(\ell_2 - 1) + \ldots + k_i(\ell_i - 1) \tag{5.2}$$

This will always be bigger than $2b$ when $\ell_1^{k_1}\ell_2^{k_2}\ldots\ell_i^{k_i} \approx 3^b$ and most $\ell_i > 3$. Bob's degree in SIDH is $3^b$ which is the smallest prime factor he can work over.

Let us take the prime in 3.3 and use the equation in 5.2 we get a total of:

$$\#Q = 115 \cdot (3-1) + (7-1) + (13-1) + 2 \cdot (31-1) + (157-1) + (241-1) = 704$$

Comparing this to it's equivalent in SIDH namely SIKEp434 where $p = 2^{216}3^{137}-1$, the number of queries is:

$$\#Q = 2 \cdot 137 = 274$$

We can already observe there is a severe difference in total queries between SIDH and B-SIDH.

One crucial thing to note is this only applies if, for each iteration, our step size is $\beta_i = 1$. In the case where

$$C_i = M - N \equiv 3 \bmod 4$$

it cannot be of the form $u_i^2 + 4v_i^2$. Therefore it may be necessary to take a step size of two, i.e. remove two factors. Let $\ell_1$ and $\ell_2$ be the two factors we remove, then have a total of $\ell_1 \cdot \ell_2 - 1$ guesses.

### 5.1.8 B-SIDH Attacking Algorithm

By using the attacking algorithm in 4.3.5 as a base, we can create a new six-step attacking algorithm suited for attacking B-SIDH. Let $N = \eta_1^{k_1-\alpha_{1i}}\eta_2^{k_2-\alpha_{2i}}\ldots\eta_j^{k_i-\alpha_{ki}}$

and $M = \ell_1^{k_1-\beta_{1i}} \ell_2^{k_2-\beta_{2i}} \dots \ell_i^{k_i-\beta_{ki}}$ and denote $N'$ and $M'$ as the remaining factors in $N$ and $M$.

**Step 1:** Choose $\beta_i = \beta_{1i} + \beta_{2i} + \dots \beta_{ki}$ minimal such that there exists some $\alpha_i \geq 0$ where $\alpha_i = \alpha_{1i} + \alpha_{2i} + \dots \alpha_{ki}$, for which $C_i = N' - M'$ is of the form $u_i^2 + 4v_i^2 > 0$. Denote $\ell_l > 0$ as the prime factor we will recover from Bob's order and $\eta_n \geq 0$ as the factor from Alice.

**Step 2:** Make a guess for $k_i \in [0, \ell_j\rangle$

**Step 3:** Use $u_i$ and $v_i$ to construct an auxiliary isogeny $\gamma = [u_i] + [v_i] \cdot 2i$

**Step 4:** Determine the $\Lambda \cdot \ell_l$-isogeny $\tau_i : E_{start} \to C_i$ with kernel $\gamma(\ker \phi_i)$, where $\ker \phi_i$ as equation 5.1.

**Step 5:** Compute the points $P_{C_i}, P_{Q_i} \in C_i$ where $P_{C_i} = \frac{N}{N'} k_i \tau(P_a)$ and $Q_{C_i} = \frac{N}{N'} k_i \tau(Q_a)$

**Step 6:** Check whether or not the subgroup $\langle (P_{C_i}, \frac{N}{N'} P_B), (Q_{C_i}, \frac{N}{N'} Q_B) \rangle$ splits.

## 5.2 Attacking B-SIDH

In this section, we shall proceed to attack B-SIDH with three distinct examples. The first example consists of a simplified toy example. This toy example will enable us to delve deeper into the details of the attack mechanism while highlighting the notable modifications from the SIDH attack. In the subsequent stages, we will engage with two more complex examples. Prior to the two more complex examples, we will conduct a preliminary analysis to estimate the time requirements and provide discussion where deemed necessary.

### 5.2.1 Toy Example

For the toy example, we will use $p = 2^{13} - 1 = 8191$ which gives Alice order $N = 2^{13}$ and Bob $M = 3^2 \cdot 5 \cdot 7 \cdot 13$. We are working in $\mathbb{F}_{p^4} = \mathbb{F}_p(\omega)$ where $\omega^4 + 3\omega^2 + 8183\omega + 17$. Using the new representation in 5.1.6, Bob's key can be written as, $k_b = (k_{11} + k_{12} \cdot 3) \cdot \frac{M}{3^2} \cdot y_1 + k_2 \cdot \frac{M}{5} \cdot y_2 + k_1 \cdot \frac{M}{7} \cdot y_3 + k_1 \cdot \frac{M}{13} \cdot y_4$. We will go through each iteration of the algorithm in Section 5.1.8, explaining each step and providing further analysis where necessary.

**Setup** During the attack we need Alice's basis points to create the auxiliary isogeny and Bob's basis points to check if it splits. Running through the B-SIDH protocol with the parameters shown above yields us these basis points:

$$P_A := (2578 \cdot \omega^3 + 2121 \cdot \omega^2 + 5156 \cdot \omega + 1220, -)$$
$$Q_A := (5746 \cdot \omega^3 + 6602 \cdot \omega^2 + 3301 \cdot \omega + 7002, -)$$
and
$$P_B := (751 \cdot \omega^3 + 3004 \cdot \omega^2 + 1502 \cdot \omega + 161, -)$$
$$Q_B := (2330 \cdot \omega^3 + 1129 \cdot \omega^2 + 4660 \cdot \omega + 6415, -)$$

From Bob's public key, we note that his public curve is the curve $E_B$ where $a = (4722 \cdot \omega^3 + 2506 \cdot \omega^2 + 1253 \cdot \omega + 3254)$ and j-invariant of $4251 \cdot i + 2640$. We will initiate a dictionary isogenies degrees we have recovered $sk_{Bob} = \{3^2 : [-,-], 5 : -, 7 : -, 13 : -\}$

**Iteration 1** We begin by guessing the first digit of Bob's secret key.

Step 1 We find that for $\beta_1 = 1$ or removing 5 from M, we have a $\alpha_1 = 0$ for which $C_1 = 2^{13} - 3^2 \cdot 7 \cdot 13 = 7373$. This can be written of the form $C_1 = u_i^2 + 4v_i^2 = (-77)^2 + 4 \cdot 19^2 = 7373$

*Our first step size is $\beta_1 = 1$ which means we will make a guess for one of Bob's factors, in particular the 5-isogeny. This means we have* 5 *possibilities in the first iteration.*

Guess 1 $k_1 = 0$

Step 3 We can construct our auxiliary isogeny $\gamma = [-77] + [19] \cdot 2i$

Step 4 The $\Lambda \cdot 5$-isogeny $\tau_1 : E_{start} \to C_1$ with kernel $\gamma(\ker \phi_1)$, where $\ker \phi_1 = \Lambda P_B + (0 \cdot \frac{M}{5} \cdot y_1) \cdot \Lambda Q_B$

*For the time being* $\Lambda = 1$

Step 5 Compute the points $P_{C_1}, Q_{C_1} \in C_1$ where $P_{C_1} = 2^0 \tau_1 \gamma(P_A)$ and $Q_{C_1} = 2^0 \tau_1 \gamma(Q_A)$

*We have created a new curve $C_1$ with $a_1 = 3384$ and the two new points are: $P_{C_1} = (3429 \cdot \omega^3 + 5525 \cdot \omega^2 + 6858 \cdot \omega + 6107, -)$ and $Q_{C_1} = (7973 \cdot \omega^3 + 7319 \cdot \omega^2 + 7755 \cdot \omega + 2533, -)$*

Step 6 Check whether or not the subgroup $\langle (P_{C_1}, 2^{13} \cdot P_B), (Q_{C_i}, 2^{13} \cdot Q_B) \rangle$ splits.

*We want to glue the curves $C_1$ and $E_{start}$ into the jacobian of a genus 2 curve, via the (2,2)-subgroup provided in step 6. Going through step 1 and 2 in 4.3.4 our isogeny chain will look like:*

$$C_1 \times E_{start} \to Jac(C_2) \to Jac(C_3) \to \ldots \to\to Jac(C_12)$$

*from the last Jacobian we will check if it splits, by determining the determinant. It turns out the first guess we made, the determinant does vanish which means it does split, and we can update our recovered digits $sk_{Bob} = \{3^2 : [-,-], 5 : 0, 7 : -, 13 : -\}$.*

**Iteration 2** After first iteration we have so far recovered $sk_{Bob} = \{3^2 : [-,-], 5 : 0, 7 : -, 13 : -\}$, and update our recovered factor variable $\Lambda = 5$

Step 1 We find that for $\beta_2 = 2$ or removing two factors from M', namely 3 and 3, we have a $\alpha_2 = 0$ for which $C_2 = 2^{13} - 7 \cdot 13 = 8101$. This can be written of the form $C_2 = u_2^2 + 4v_2^2 = (-1)^2 + 4 \cdot (-45)^2 = 8101$

*For this iteration we have to make a guess for two factors which means we have $3 \cdot 3$ possibilities in this iteration.*

Guess 1 $k_{21}, k_{22} = \{0, 0\}$

Step 3 We can construct our auxiliary isogeny $\gamma = [-1] + [-45] \cdot 2i$

Step 4 The $\Lambda(3 \cdot 3)$-isogeny $\tau 21 : E_{start} \to C_2$ with kernel $\gamma(\ker \phi_2)$, where $\ker \phi_2 = \Lambda P_B + (0 \cdot \frac{M}{5} \cdot y_1 + (0 + 0 \cdot 3) \cdot \frac{M}{3^3} \cdot y_2) \cdot \Lambda Q_B$

Step 5 Compute the points $P_{C_2}, Q_{C_2} \in C_2$ where $P_{C_2} = 2^0 \tau_2 \gamma(P_A)$ and $Q_{C_2} = 2^0 \tau_2 \gamma(Q_A)$
*We have created a new curve $C_2$ with $a_2 = (384 \cdot \omega^3 + 1536 \cdot \omega^2 + 768 \cdot \omega + 7485)$ and the two new points are: $P_{C_1} = (6734 \cdot \omega^3 + 2363 \cdot \omega^2 + 5277 \cdot \omega + 7039, -)$ and $Q_{C_1} = (3851 \cdot \omega^3 + 7213 \cdot \omega^2 + 7702 \cdot \omega + 2851, -)$*

Step 6 Check whether or not the subgroup $\langle (P_{C_2}, 2^{13} \cdot P_B), (Q_{C_2}, 2^{13} \cdot Q_B) \rangle$ splits.
*We want to glue the curves $C_2$ and $E_{start}$ into the jacobian of a genus 2 curve, via the (2,2)-subgroup provided in step 6. Going through step 1 and 2 in 4.3.4 our isogeny chain will look like:*

$$C_2 \times E_{start} \to Jac(C_2) \to Jac(C_3) \to \ldots \to\to Jac(C_12)$$

*from the last Jacobian we will check if it splits, by determining the determinant. It does not split and we have to continue guessing*

$$\vdots \qquad\qquad \vdots$$

Guess 5 $k_{21}, k_{22} = \{1, 1\}$

Step 3 We can construct our auxiliary isogeny $\gamma = [-1] + [-45] \cdot 2i$

Step 4 The $\Lambda(3 \cdot 3)$-isogeny $\tau 21 : E_{start} \to C_2$ with kernel $\gamma(\ker \phi_2)$, where $\ker \phi_2 = \Lambda P_B + (0 \cdot \frac{M}{5} \cdot y_1 + (1 + 1 \cdot 3) \cdot \frac{M}{3^3} \cdot y_2) \cdot \Lambda Q_B$

Step 5 Compute the points $P_{C_2}, Q_{C_2} \in C_2$ where $P_{C_2} = 2^0 \tau_2 \gamma(P_A)$ and $Q_{C_2} = 2^0 \tau_2 \gamma(Q_A)$
*We have created a new curve $C_2$ with $a_2 = (4775 \cdot \omega^3 + 2718 \cdot \omega^2 + 1359 \cdot \omega + 3641)$ and the two new points are: $P_{C_1} = (581 \cdot \omega^3 + 2324 \cdot \omega^2 + 1162 \cdot \omega + 3930, -)$ and $Q_{C_1} = (7139 \cdot \omega^3 + 3983 \cdot \omega^2 + 6087 \cdot \omega + 6828, -)$*

Step 6 Check whether or not the subgroup $\langle (P_{C_2}, 2^{13} \cdot P_B), (Q_{C_2}, 2^{13} \cdot Q_B) \rangle$ splits.

*We want to glue the curves $C_2$ and $E_{start}$ into the jacobian of a genus 2 curve, via the (2,2)-subgroup provided in step 6. Going through step 1 and 2 in 4.3.4 our isogeny chain will look like:*

$$C_2 \times E_{start} \to Jac(C_2) \to Jac(C_3) \to \ldots \to\to Jac(C_12)$$

*from the last Jacobian we will check if it splits, by determining the determinant. This configuration does split! And we can update our recovered digits* $sk_{Bob} = \{3^2 : [1,1], 5 : 0, 7 : -, 13 : -\}$.

**Iteration 3** After second iteration we have so far recovered $sk_{Bob} = \{3^2 : [1,1], 5 : 0, 7 : -, 13 : -\}$, and update our recovered factor variable $\Lambda = 3^2 \cdot 5$

Step 1 We find that for $\beta_3 = 1$ or removing 13 from M', we have a $\alpha_2 = 0$ for which $C_2 = 2^{13} - 7 = 8185$. This can be written of the form $C_2 = u_2^2 + 4v_2^2 = (-83)^2 + 4 \cdot (18)^2 = 8185$

*For this iteration we have to make a guess for one factor which means we have* 13 *possibilities.*

Guess 1 $k_3 = 0$

Step 3 We can construct our auxiliary isogeny $\gamma = [-83] + [18] \cdot 2i$

Step 4 The $\Lambda \cdot 13$-isogeny $\tau_3 : E_{start} \to C_3$ with kernel $\gamma(\ker \phi_3)$, where $\ker \phi_3 = \Lambda P_B + (0 \cdot \frac{M}{5} \cdot y_1 + (1 + 1 \cdot 3) \cdot \frac{M}{3^3} \cdot y_2 + 0 \cdot \frac{M}{13} \cdot y_3) \cdot \Lambda Q_B$

Step 5 Compute the points $P_{C_3}, Q_{C_3} \in C_3$ where $P_{C_3} = 2^0 \tau_3 \gamma(P_A)$ and $Q_{C_3} = 2^0 \tau_3 \gamma(Q_A)$

*We have created a new curve $C_3$ with* $a_3 = (5627 \cdot \omega^3 + 6126 \cdot \omega^2 + 3063 \cdot \omega + 4635)$ *and the two new points are:* $P_{C_3} = (2021 \cdot \omega^3 + 8084 \cdot \omega^2 + 4042 \cdot \omega + 5236, -)$ *and* $Q_{C_3} = (4489 \cdot \omega^3 + 1574 \cdot \omega^2 + 787 \cdot \omega + 6496, -)$

Step 6 Check whether or not the subgroup $\langle (P_{C_3}, 2^{13} \cdot P_B), (Q_{C_3}, 2^{13} \cdot Q_B) \rangle$ splits.

*We want to glue the curves $C_3$ and $E_{start}$ into the jacobian of a genus 2 curve, via the (2,2)-subgroup provided in step 6. Going through step 1 and 2 in 4.3.4 our isogeny chain will look like:*

$$C_3 \times E_{start} \to Jac(C_3) \to Jac(C_3) \to \ldots \to\to Jac(C_12)$$

*from the last Jacobian we will check if it splits, by determining the determinant. It does not split and we have to continue guessing*

Guess 2 $k_3 = 1$

Step 3 We can construct our auxiliary isogeny $\gamma = [-83] + [18] \cdot 2i$

Step 4 The $\Lambda \cdot 13$-isogeny $\tau_3 : E_{start} \to C_3$ with kernel $\gamma(\ker \phi_3)$, where $\ker \phi_3 = \Lambda P_B + (0 \cdot \frac{M}{5} \cdot y_1 + (1 + 1 \cdot 3) \cdot \frac{M}{3^3} \cdot y_2 + 1 \cdot \frac{M}{13} \cdot y_3) \cdot \Lambda Q_B$

Step 5 Compute the points $P_{C_3}, Q_{C_3} \in C_3$ where $P_{C_3} = 2^0 \tau_3 \gamma(P_A)$ and $Q_{C_3} = 2^0 \tau_3 \gamma(Q_A)$
*We have created a new curve $C_3$ with $a_3 = (6578 \cdot \omega^3 + 1739 \cdot \omega^2 + 4965 \cdot \omega + 6985)$ and the two new points are: $P_{C_3} = (6702 \cdot \omega^3 + 2235 \cdot \omega^2 + 5213 \cdot \omega + 3183, -)$ and $Q_{C_3} = (4654 \cdot \omega^3 + 2234 \cdot \omega^2 + 1117 \cdot \omega + 326, -)$*

Step 6 Check whether or not the subgroup $\langle (P_{C_3}, 2^{13} \cdot P_B), (Q_{C_3}, 2^{13} \cdot Q_B) \rangle$ splits.
*We want to glue the curves $C_3$ and $E_{start}$ into the jacobian of a genus 2 curve, via the (2,2)-subgroup provided in step 6. Going through step 1 and 2 in 4.3.4 our isogeny chain will look like:*

$$C_3 \times E_{start} \to Jac(C_3) \to Jac(C_3) \to \ldots \to\to Jac(C_12)$$

*from the last Jacobian we will check if it splits, by determining the determinant. For $k_3 = 1$, it splits. We can update our recovered digits $sk_{Bob} = \{3^2 : [1,1], 5 : 0, 7 : -, 13 : 1\}$.*

**Iteration 2** We have recovered all isogenies except one and we will brute force the last by computing the isogeny from $E_{start} \to E_B$

Guess 1 $k_4 = 0$

- *bobskey* $= (0 \cdot \frac{M}{5} \cdot y_1 + (1 + 1 \cdot 3) \cdot \frac{M}{3^3} \cdot y_2 + 1 \cdot \frac{M}{13} \cdot y_3 + 0 \cdot \frac{M}{7} \cdot y_4) \% M = 2380$
- The $M$-isogeny $\phi : E_{start} \to E'$ with $ker(\phi) = P_B + bobskey \cdot Q_B$
- Check if $j(E') = j(E_B)$
  *For $k_4 = 0$, $j(E') = j(E_B)$ returns false which means we have not made the correct guess.*

Guess 2 $k_4 = 0$

- *bobskey* $= (0 \cdot \frac{M}{5} \cdot y_1 + (1 + 1 \cdot 3) \cdot \frac{M}{3^3} \cdot y_2 + 1 \cdot \frac{M}{13} \cdot y_3 + 1 \cdot \frac{M}{7} \cdot y_4) \% M = 3550$
- The $M$-isogeny $\phi : E_{start} \to E'$ with $ker(\phi) = P_B + bobskey \cdot Q_B$
- Check if $j(E') = j(E_B)$
  *For $k_4 = 1$, $j(E') = j(E_B)$ is indeed true which means we have the last correct digit.*

**Finish** We have successfully attacked SIDH by recovering Bob's key digit by digit $sk_{Bob} = \{3^2 : [1,1], 5 : 0, 7 : 1, 13 : 1\}$ and $k_b = 3550$.

### 5.2.2 Timing How Long One Guess Take

During the analysis of the next two examples, we need to estimate how long it takes to make one guess for each isogeny of Bob's order. This estimation was conducted using our modified implementation of the attack. We executed this attack on the prime numbers cited in the examples and recorded the time used for each guess.

Our time measurement began when the guessing number was selected and stopped upon the oracle's response. The tables showcased, namely Table 5.1 and 5.2, in the examples represent the mean values derived from roughly 100 guesses for every isogeny degree. All timing computations were performed on a MacBook Air with an Apple M1 chip.

### 5.2.3 Attacking Example II

For the 127-bit prime, we will use the prime presented in 3.7.2.

$$p = 2^{127} - 1$$

The presented prime is one of the Mersenne primes which enables Alice to only work with two primes. Since $p - 1$ is not as smooth we need to slightly adjust the order of Bob by removing the two biggest factors. The reasoning behind this is that computing isogenies of this degree will take seconds. From an attacking point, this will result in months to years of computing, and from a practical point, this will never be used as the key exchange will be too slow. Alice and Bob's order will be as follow:

$$\begin{aligned} M &= 2^{127} \\ N &= 3^3 \cdot 7^2 \cdot 19 \cdot 43 \cdot 73 \cdot 127 \cdot 337 \cdot 5419 \cdot 92737 \end{aligned}$$

Using the equations in 5.1.7, we can find the worst, average, and best number of queries done to our oracle.

$$\#Q_{best} = 12 \qquad \#Q_{avrg} = 49383 \qquad \#Q_{worst} = 98766$$

This estimate is only viable when the step size $\beta_i = 1$, however, this does not hold for this example. No matter which factors one begins with, one will always end up recovering two factors at once. It is therefore important to find the most optimal succession to limit the number of queries to our oracle. Using some of the techniques we will discuss later in section 5.3, we find that one of the most optimal succession is:

| Time of guessing digit | |
|---|---|
| Isogeny degree | Avrg Time |
| 73 | 1.071s |
| 337 | 1.147s |
| $7 \cdot 43$ | 1.053s |
| $3 \cdot 19$ | 1.013s |
| $3 \cdot 7$ | 0.8460s |
| $3 \cdot 127$ | 0.5289s |
| 92737 | 3.207s |
| 5419 | 1.156s |

**Table 5.1:** Mean time of how long one guess takes for a given isogeny.

$$(73, 2^{122}) \to (337, 2^{121}) \to (7 \cdot 43, 2^{108}) \to (3 \cdot 19, 2^{100}) \to$$
$$(3 \cdot 7, 2^{79}) \to (3 \cdot 127, 2^{29}) \to (92737, 2^{29}) \to 5419$$

where the first element in the tuple is the isogeny degree we are recovering and the second element is $2^{127-\alpha_i}$. With this in mind, we can calculate the new best, average, and worst number of queries.

$$\#Q_{best} = 8 \qquad \#Q_{avrg} = 49653 \qquad \#Q_{worst} = 99306$$

How long it takes to make one guess is dependent on many things. The isogeny degree we are recovering, how many $(2, 2)$-isogenies we need to compute, pushing the basis points through the auxiliary isogeny. In Table 5.1 we present on average how long it will take to make one guess for each part.

Using these values we can find the best, average, and worst running time for our attack.

$$time_{best} = 10s \quad time_{avrg} = 152366s(42hours) \quad time_{worst} = 304732s(85hours)$$

Practically running this example multiple times is not feasible without dedicated equipment that can run for many hours up until days. However, we did run it once and the attacking time clocked in at approximately 31 hours, which is below average according to our analysis.

### 5.2.4 Attacking Example III

For the final example, we will utilize the 255-bit prime

$$p = 0x76042798BBFB78AEBD02490BD2635DEC131AB\\ FFFFFFFFFFFFFFFFFFFFFFFFFFF$$

Given our inability to compute $(\ell, \ell)$-isogenies for $\ell > 2$, it becomes necessary to adjust the orders assigned to Alice and Bob. Alice's order will need to be constrained to only two primes. Concurrently, to ensure the condition $M - N > 0$ is met, we will reduce some of Bob's factors. To prevent this example from leaning too close to SIDH, we will systematically eliminate Bob's three factors until the condition $M - N > 0$ is satisfied. Consequently, the revised orders for Alice and Bob will be as follows:

$$\begin{aligned} M &= 2^{110} \\ N &= 3^{34} \cdot 11 \cdot 17 \cdot 19^2 \cdot 29 \cdot 37 \cdot 53^2 \cdot 97 \cdot 107 \end{aligned}$$

Using the equations in 5.1.7, we can find the worst, average, and best number of queries done to our oracle.

$$\#Q_{best} = 44 \qquad \#Q_{avrg} = 250 \qquad \#Q_{worst} = 500$$

Just like the example in 5.2.3, this estimate is only viable when the step size $\beta_i = 1$. Finding the optimal sequence is not a trivial task, however, we believe one of the best sequences is:

$$(3, 2^{110}) \to (3 \cdot 3, 2^{107}) \to (3 \cdot 3, 2^{100}) \to (3 \cdot 3, 2^{100}) \to (3 \cdot 3, 2^{95}) \to (3 \cdot 3, 2^{88}) \to \\ (3 \cdot 3, 2^{86}) \to (3 \cdot 3, 2^{82}) \to (3 \cdot 3, 2^{78}) \to (3 \cdot 3, 2^{78}) \to (3 \cdot 3, 2^{72}) \to (3 \cdot 3, 2^{69}) \to \\ (3 \cdot 3, 2^{66}) \to (3 \cdot 3, 2^{63}) \to (3 \cdot 3, 2^{59}) \to (3 \cdot 3, 2^{57}) \to (33, 2^{57}) \to (3 \cdot 19, 2^{48}) \to \\ (3 \cdot 19, 2^{42}) \to (17, 2^{35}) \to (37, 2^{35}) \to (53, 2^{35}) \to (29 \cdot 53, 2^{14}) \to (97, 2^{8}) \to 107$$

where the first element in the tuple is the isogeny degree we are recovering and the second element is $2^{127-\alpha_i}$. With this in mind, we can calculate the new best, average, and worst number of queries.

$$\#Q_{best} = 24 \qquad \#Q_{avrg} = 1581 \qquad \#Q_{worst} = 3162$$

Comparing this to the example in 5.2.3, we have on average only 3% of guesses. This is of course due to the large isogeny which carries much more possibilities.

In Table 5.2 we present on average how long it will take to make one guess for each part.

| Time of guessing digit | |
|---|---|
| Isogeny degree | Avrg Time |
| 3 | 1.762 |
| $(3 \cdot 3) \times 15$ | 1.560s (min/1.294, max/1.919)) |
| 33 | 1.336s |
| $3 \cdot 19$ | 1.260s |
| $3 \cdot 19$ | 1.209s |
| 17 | 1.148s |
| 37 | 1.210s |
| 53 | 1.174s |
| $29 \cdot 53$ | 1.125s |
| 97 | 1.162s |
| 107 | 0.5807s |

**Table 5.2:** Mean time of how long one guess takes for a given isogeny. $\times 15$ denotes a $(3 \cdot 3)$-isogeny where compute 15 times and the minimum, mean, and max value is given.

Using these values we can find the best, average, and worst running time for our attack.

$$time_{best} = 13.52s \quad time_{avrg} = 1197s(20minutes) \quad time_{worst} = 2394s(40minutes)$$

To validate our analysis, we conducted a practical test of the attack on the given example, repeating it five times. The average duration across these runs amounted to 24 minutes. However, it's important to keep in mind the execution time is strongly influenced by the nature of Bob's secret key.

## 5.3 Ways To Speed Up the Attack

In this section, we will explore various techniques and optimizations aimed at accelerating the attack B-SIDH. As cryptographic attacks are often computationally intensive, improving the efficiency is crucial for reducing the time and resources required to recover Bob's secret key, we will examine a variety of methods that can help optimize the attack process.

These methods encompass parallelization, pre-computation, algorithmic improvements, and strategic ordering of computations. Castryck and Decru present some of these methods in [CD23b], as well as the lessons learned during the development of suiting the attack to B-SIDH.

### 5.3.1 Compute Isogenies Using $\sqrt{\acute{e}lu}$

The primary factor that makes attacking B-SIDH slower is the computation of high-degree isogenies. As discussed in Section 3.7.3, the $\sqrt{\grave{e}lu}$ algorithm requires $\mathcal{O}(\sqrt{\ell})$ field operations, which is more efficient than the naive $\mathcal{O}(\ell)$ approach, where $\ell$ is the degree of the isogeny.

According to the SageMath documentation for $\sqrt{\acute{e}lu}$, the isogeny degree threshold where $\sqrt{\acute{e}lu}$ begins to outperform the regular Vélu's algorithm can range from $\approx 100$ to $\approx 1000$, depending on the specific situation. Our experiments, detailed in Appendix A, indicate that the optimal threshold for our context is $\approx 300$. Thus, we use the regular Vélu's algorithm for isogenies of degree $< 300$ and $\sqrt{\acute{e}lu}$ for isogenies of degree $> 300$.

### 5.3.2 Parallelisation

Parallelizing code execution is an effective way to accelerate the attack. Although Castryck and Decru's attack was designed to run on a single core, the implementation in [OP22b] cleverly parallelizes certain parts of the algorithm. Each guess can be made in parallel; however, it is essential to note that we cannot run more parallel computations than the torsion part we are recovering. For example, when recovering a 5-isogeny, a maximum of four cores can be utilized for guessing the correct digit and asking our oracle. In SIDH, where we only recover 3-isogenies, using two cores is the best possible performance. Since B-SIDH often requires recovering larger isogenies, more cores can be used, making parallelization more effective.

However, running the attack on n-cores does not necessarily lead to an n-times faster attack. While increasing the number of cores does speed up the attack, each guess takes a longer time.

### 5.3.3 Recover Smallest Degree Isogenies First

A small yet effective optimization for speeding up the attack involves recovering isogenies with smaller degrees first. When determining the isogeny $\tau$ in step 3 in 5.1.8, we compute an isogeny chain with a degree equal to what we have already recovered. By computing the highest degree isogenies last, we significantly reduce the computational cost by avoiding the calculation of high-degree isogenies each time we recover a new isogeny.

### 5.3.4 Extending Bob's Secret Isogeny

Consider the scenario where there is no candidate for $\beta_i$ of only one factor such that $N' - M$ is congruent to 1 mod 4. We then have to choose two factors and significantly increase the number of queries required. One solution to this is prolonging Bob's secret isogeny with an arbitrary isogeny $\phi'$ of the smallest degree, in most cases a 3-isogeny, such that the expression $N' - M' \cdot new\_isogeny\_degree$ is 1 mod 4. Then let $P' = \phi'(P)$ and $Q = \phi'(Q)$ ´and treat $\phi \circ \phi'$ as the new secret isogeny.

### 5.3.5 Finding Optimal Course of Action

Castryck and Decru employed a precomputed table to optimize the selection of $\alpha_i$ and $\beta_i$, eliminating the need for real-time factoring. Their optimized selection strategy involved choosing $\alpha_i$ as large as possible. The larger the $\alpha_i$, the shorter the chain of $(2, 2)$-isogenies, as the length is determined by $a - \alpha_i$. However, the precomputed table used in their attack is not suitable for attacking B-SIDH, as it only works when Alice's order is $2^a$ and Bob's degree is $3^b$. Creating a generic precomputed table for B-SIDH is impractical due to the diverse prime factors involved.

Despite the infeasibility of a precomputed table, it is worthwhile to conduct some preliminary analysis to determine the most optimal sequence for recovering parts of Bob's secret integer. Taking into account the considerations from 5.3.3 and the strategy of maximizing $\alpha_i$, we can devise a simple algorithm to find the best sequence.

Let $n \geq 0$ and $m \geq 3$ represent some prime factors in $M$. We want to choose the smallest prime factor $m$ such that $C = n - \frac{M}{m}$ is positive and free of prime factors congruent to 3 mod 4. If our chosen $m$ satisfies this condition, check if there exists a larger $n$ for which $C = \frac{N}{n} - \frac{M}{m}$ holds. If no prime factors pass this test, relax the constraints and increment the number of prime factors one is considering. Choose $m = m_1 \cdot m_2 \cdot \ldots \cdot m_i$, with the smallest possible values for two prime factors $m_1 \cdot m_2 \cdot \ldots \cdot m_i \in M$. Once we have found our $n$ and $m$, we recover the isogeny of degree $m$.

In some cases, choosing two small prime factors might be more optimal than selecting one large prime factor to avoid the issue described in 5.3.3. i.e. guessing $19 \cdot 337$ before 5419 is more optimal since we avoid computing the 5419-isogeny $19 \cdot 337$ times.

Finding optimal sequence practical can be a daunting task, especially as the number of possibilities can be quite large. However, we can do some greedy choices and make an algorithm that will find a good sequence. It is essential to recognize that $C = M - N$ either will be (i) $C \equiv 1 \bmod 4$ or (ii) $C \equiv 3 \bmod 4$. If $C$ is the former, we know that all factors that are congruent to 3 mod 4 will satisfy the condition since N is always congruent to 0 mod 4 (This is because N is the prime power of two), and if $C$ is the latter, all factors congruent to 1 mod 4 mod will satisfy.

Our approach will start from the last isogeny we wish to recover. Since we also ideally like to compute large isogenies last, we can take the greedy choice of choosing the largest factor that is congruent to 3 mod 4 and find the smallest $i$ that satisfies $C = 2^i - num$. We will continue this until there is no such factor longer, and relax or constraints to the product of two numbers. Starting by trying the largest number with the smallest number. The algorithms pseudo-code is in Algorithm 5.1 and translated into Python code can be viewed in appendix B. Please note that the algorithm is tailored when Alice's order is a power of two.

## 5.4 Can B-SIDH Be Considered Broken?

The question of whether B-SIDH is broken or not merits an interesting discussion. From a conceptual perspective, a cryptographic protocol can be considered *broken* when an adversary can efficiently recover the secret keys or otherwise compromise the security promises of the protocol. Efficiency, in this context, is typically measured in terms of time complexity.

In the case of B-SIDH, we find that the security landscape is quite varied, dependent upon the specific parameters in play. As discussed earlier, the SIDH protocol is completely broken where the secret key can be recovered within seconds or minutes for all parameters. B-SIDH, on the other hand, presents a more complex scenario.

For certain prime numbers, as shown in section 5.2, the secret key of B-SIDH can indeed be recovered easily. Specifically, this is true when p+1, the degree of Alice's isogeny, is the product of only two primes. The reason for this vulnerability lies in the requirement to compute $(n, n)$-isogenies, where $n$ represents the different prime factors of Alice's order. For values of $n$ greater than 2, efficient polynomial expressions for these isogenies are currently unavailable or hard to utilize. This limits

**Algorithm 5.1** Algorithm for finding an optimal sequence of recovering isogenies.

```
 1: $M_{tmp} \leftarrow M$
 2: $recovered\_M \leftarrow 1$
 3: $results \leftarrow [[-1, -1, -1, -1]]$
 4: while $|\{p^e \mid p^e \in factor(M_{tmp})\}| > 1$ do
 5:     for $prime\_factor \in prime\_combinations(M_{tmp})$ do
 6:         $recovered\_M \leftarrow recovered\_M \times prime\_factor$
 7:         if $recovered\_M \mod 4 \neq 3$ then
 8:             $recovered\_M \leftarrow recovered\_M / prime\_factor$
 9:             continue
10:         end if
11:         $result \leftarrow find\_lowest\_alpha(N, recovered\_M)$
12:         if $result$ is $None$ then
13:             continue
14:         end if
15:         $N_{low}, u, v \leftarrow result$
16:         $M_{tmp} \leftarrow M_{tmp} / prime\_factor$
17:         $results[-1][1] \leftarrow prime\_factor$
18:         $results.append([N_{low}, -1, u, v])$
19:         break
20:     end for
21: end while
22: $results[-1][1] \leftarrow M_{tmp}$
23: $results.reverse()$
24: return $results$
```

the range of prime numbers for which the attack can be performed efficiently, and thereby the range of parameters where B-SIDH can be considered broken.

However, the claim that a protocol is *broken* does not necessarily mean that an attack is feasible in all practical scenarios. Cryptographic strength is not a binary quality, but rather a spectrum of resistance against potential attacks. If an attack algorithm exists that runs in polynomial time, the protocol can be considered theoretically broken, even if the required time for the attack is beyond the reach of current computational capabilities.

In this context, the Castryck and Decru attack on B-SIDH can be considered as running in polynomial time, which indicates that B-SIDH is theoretically broken. Nevertheless, for larger prime factors, the required computation time might extend to years, decades, or even longer. Effectively placing a successful attack beyond the realm of practical feasibility with current knowledge.

While B-SIDH does have vulnerabilities that can be exploited under specific conditions, the range of these conditions is limited. The protocol can therefore be considered theoretically broken, but its practical security remains dependent on the choice of parameters.

In light of these findings, the status of B-SIDH's security further underscores the need for ongoing research. The key to extending the practicality of attacks on B-SIDH lies in the development of efficient polynomial expressions for $(N, N)$-isogenies Currently, our ability to exploit the protocol's vulnerabilities is restricted by the lack of such expressions.

The challenge of finding fast and efficient polynomials for $(N, N)$-isogenies, therefore, marks a significant frontier in isogeny-based cryptography research. By meeting this challenge, we may not only practically attack B-SIDH across all primes, but also inform the development of more robust and secure cryptographic systems.

Chapter 6

# Conclusion

In this thesis, we have studied to what extent Castryck and Decru's attack holds in the case of B-SIDH and explored the applicability of the attack. Our findings indicate that the attack adapts well to B-SIDH under certain circumstances - specifically when $p+1$ or $p-1$, factorizes to only prime powers of two. Additionally, we had to confront some inherent complexities associated with B-SIDH. One of these complexities arose from the practical difficulties in computing $(N, N)$-isogenies, corresponding to Alice's prime powers, which presented a major challenge in our study.

To further optimize the attack's applicability to B-SIDH, we made several modifications to the original methodology of Castryck and Decru's attack. These included modifying the attack to take into account Bob's twisted torsion and creating a new representation of Bob's secret key using the Chinese Remainder Theorem (CRT). However, we found that attempting an attack on B-SIDH is innately slower due to the necessity of computing isogenies of higher degrees.

Moving forward, we illustrate the practicalities and challenges of our adapted attack using two concrete examples, each corresponding to a different prime where Alice's order is a power of two. The first example showed a considerably slower running time, while the second one was significantly faster. The discrepancy between the two can be largely attributed to the varying isogeny degrees and the differing number of oracle queries required.

Building upon these findings, we sought to devise a more efficient attack strategy. We developed a simple algorithm that aimed at finding a better sequence for recovering isogenies. The algorithm, under the right circumstances, enhances the efficiency of the attack on B-SIDH, effectively cutting down the running time.

## 6.1 Research Questions

The primary research objective for this thesis is to find out *to what extent does Castryck and Decru's attack hold in the case of B-SIDH.* To answer this research objective, we formulated three key research questions that guided our exploration of the attack's relevance and efficacy in the context of B-SIDH. Throughout this thesis, we have aimed to answer these questions. This section will provide a concise summary of the insights gathered during our research.

*Research question 1: What properties of SIDH are used in the attack?*

The Castryck-Decru attack on Supersingular Isogeny Diffie-Hellman (SIDH) exploits several properties of the protocol and its parameters:

- **Auxiliary Points Exchange:** In the SIDH protocol, the exchanged data include the images of the auxiliary points. These are used to construct auxiliary isogenies, which are key to the attack.
- **Known Degree of Secret Isogeny:** In SIDH, the secret isogeny $\phi_B$ has a fixed and known degree. The attack uses this information to recover Bob's secret key digit by digit, using a sequence of queries to an oracle rather than computing the secret isogeny itself.
- **Known Endomorphism Ring of Initial Curve:** The initial curve $E_0$ in SIKE is always the same, and its endomorphism ring is known. This allows the construction of an auxiliary isogeny of a particular degree whenever it can be represented in the form $x = u^2 + 4v^2$. However, due to work by Robert [Rob23], this is not required.
- **Isogeny Diamond Configuration:** The attack utilizes the isogeny diamond configuration, a concept rooted in Kani's theorem, to construct explicit isogenies between pairs of supersingular elliptic curves.
- **Structure of Secret Generator:** The structure of the secret generator in SIDH, specifically the kernel of Bob's secret isogeny, is exploited to guess Bob's secret integer digit by digit.
- **Effective Computation of (2,2)-Isogenies:** The attack uses the efficient computation of (2,2)-isogenies called Richelot isogenies, especially when the degree $N_A$ is chosen to be $2^n$. This allows the construction of the glue-and-split oracle.

The attack cleverly combines these properties to create a practical attack against SIDH.

*Research question 2: Can Castryck and Decru's attack be used with the same efficiency on B-SIDH as on SIDH*

Castryck and Decru's attack does not operate with the same efficiency on B-SIDH as it does on SIDH. The primary reason is the fact that B-SIDH utilizes higher degree prime factors, and the efficiency of the attack is polynomial in the isogeny degrees. Consequently, the increased isogeny degrees in B-SIDH would result in a slower attack than in the case of SIDH. The increased isogeny prime degrees also require more queries to our Glue-and-split oracle and section 5.1.7 present a way for finding the number of queries required. Another crucial factor is the necessity to compute $(N, N)$-isogenies for $N > 2$. These computations are significantly slower because efficient polynomials for these calculations are currently unavailable. Furthermore, despite our efforts as detailed in 5.1.2, we were unable to practically implement these computations, which results in only practical attacks where Alice's order is a prime power of two.

*Research question 3: What modification needs to be done to Castryck and Decru's attack in order the use the same attack on B-SIDH*

There are two main modifications. Due to B-SIDH's utilization of twisted torsion, we must adjust the attack to accommodate this difference. As discussed in section 5.1, this results that the attack needs to handle varying isogeny degrees and a varying number of guesses, unlike the original attack scheme on SIDH. The ternary representation of Bob's key within SIDH can no longer be utilized as Bob's order is not a prime power of three. To overcome this obstacle, presented in section 5.1.6, we made an implementation using the Chinese Remainder Theorem (CRT) to provide an effective representation of Bob's key. Lastly, the glue-and-split oracle needs to compute $(\ell, \ell)$-isogenies for Alice's prime powers in order to determine if the correct guess has been made.

## 6.2 Limitations and Remarks

Firstly, our research focused solely on applying Castryck and Decru's attack to B-SIDH, with no exploration of other potential attacks or protocols. Since late 2022 and the inception of this thesis, there have been additional inspired attacks introduced in the field. However, these attacks have been acknowledged but not integrated into our work on attacking B-SIDH, where our sole focus is Castryck and

Decru's attack. Second, due to computational limits and time constraints, we weren't able to fully implement $(N, N)$-isogenies and test them practically.

This work was also limited in its scope to explore the possibilities of attacking B-SIDH, while some optimizations of attacking are mentioned, this is not a main focus. Moreover, although we explored some modifications to the attack to fit the specifics of B-SIDH, there might be other optimizations or adjustments that we did not consider.

Despite these limitations, we believe that our work contributes valuable insights into the world of isogeny-based cryptography and the scope of Castryck and Decru's attack. It also builds on the groundwork for future research to further explore attacking B-SIDH. We also believe that open-source code is important, all code used in this thesis can therefore be found on GitHub. [1]

## 6.3 Future Work

Our research leaves room for more work on attacking B-SIDH. We give three open problems that, if solved, will broaden the practical application of attacking B-SIDH. The first two problems relate to the ability to compute $(\ell, \ell)$-isogenies, while the last open problem aims to further optimize the attacking strategy.

**Open problem 1:** *Write code that utilizes other representations of* $(\ell, \ell)$*-isogenies.*

> The efficient Richelot isogenies are exclusive to $(2, 2)$-isogenies. In order to orchestrate a full-scale attack on B-SIDH, we require additional representations. An intriguing candidate would be the polynomial $(\ell, \ell)$-isogenies by Cosset and Robert. Successfully integrating this into the attack could pave the way for an attack on every prime. However, for certain primes, this might be practically unfeasible due to extensive computation times unless a faster representation is identified.

**Open problem 2:** *Translate Decru and Kunzweiler's fast* $(3^n, 3^n)$*-isogeny computation into SageMath*

> Decru and Kunzweiler present a fast $(3^n, 3^n)$-isogeny computation, and translating this from Magma to SageMath would enable an efficient attack on primes where Alice's order contains both prime powers of two and three.

[1] https://github.com/georgsku/BSIDH-attack

**Open problem 3:** *Further optimize the attacking strategy on B-SIDH*

We have proposed an algorithm to efficiently find sequences for recovering isogenies. However, we conjecture that further optimizations are feasible and the algorithm must be extended to accommodate cases where Alice's degree deviates from a prime power of two. Also, Extending Bob's secret isogeny could also be a good way to avoid having a step size of two. Other potential approaches that we haven't mentioned may also exist and are worthy of exploration.

# References

[BFLS20] D. J. Bernstein, L. D. Feo, *et al.*, «Faster computation of isogenies of large prime degree», *CoRR*, vol. abs/2003.10118, 2020. [Online]. Available: https://arxiv.org/abs/2003.10118.

[BFT14] N. Bruin, E. Flynn, and D. Testa, «Descent via (3,3)-isogeny on jacobians of genus 2 curves», *Acta Arithmetica*, vol. 165, Jan. 2014.

[CD23a] W. Castryck and T. Decru, «An efficient key recovery attack on SIDH», 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*, C. Hazay and M. Stam, Eds., ser. Lecture Notes in Computer Science, vol. 14008, Springer, 2023, pp. 423–447. [Online]. Available: https://doi.org/10.1007/978-3-031-30589-4%5C_15.

[CD23b] W. Castryck and T. Decru, «An efficient key recovery attack on SIDH», 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*, C. Hazay and M. Stam, Eds., ser. Lecture Notes in Computer Science, vol. 14008, Springer, 2023, pp. 423–447. [Online]. Available: https://doi.org/10.1007/978-3-031-30589-4%5C_15.

[CH17] C. Costello and H. Hisil, «A simple and compact algorithm for SIDH with arbitrary degree isogenies», in *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, T. Takagi and T. Peyrin, Eds., ser. Lecture Notes in Computer Science, vol. 10625, Springer, 2017, pp. 303–329. [Online]. Available: https://doi.org/10.1007/978-3-319-70697-9%5C_11.

[CJS14] A. M. Childs, D. Jao, and V. Soukharev, «Constructing elliptic curve isogenies in quantum subexponential time», *J. Math. Cryptol.*, vol. 8, no. 1, pp. 1–29, 2014. [Online]. Available: https://doi.org/10.1515/jmc-2012-0016.

[CLM+18] W. Castryck, T. Lange, *et al.*, «CSIDH: an efficient post-quantum commutative group action», in *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology* *and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, T. Peyrin and S. D. Galbraith, Eds., ser. Lecture Notes in Computer Science, vol. 11274, Springer, 2018, pp. 395–427. [Online]. Available: https://doi.org/10.1007/978-3-030-03332-3%5C_15.

[Cos19] C. Costello, «Supersingular isogeny key exchange for beginners», in *Selected Areas in Cryptography - SAC 2019 - 26th International Conference, Waterloo, ON, Canada, August 12-16, 2019, Revised Selected Papers*, K. G. Paterson and D. Stebila, Eds., ser. Lecture Notes in Computer Science, vol. 11959, Springer, 2019, pp. 21–50. [Online]. Available: https://doi.org/10.1007/978-3-030-38471-5%5C_2.

[Cos20] C. Costello, «B-SIDH: supersingular isogeny diffie-hellman using twisted torsion», in *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, S. Moriai and H. Wang, Eds., ser. Lecture Notes in Computer Science, vol. 12492, Springer, 2020, pp. 440–463. [Online]. Available: https://doi.org/10.1007/978-3-030-64834-3%5C_15.

[CR11] R. Cosset and D. Robert, «Computing (l, l)-isogenies in polynomial time on jacobians of genus 2 curves», *IACR Cryptol. ePrint Arch.*, p. 143, 2011. [Online]. Available: http://eprint.iacr.org/2011/143.

[DH76] W. Diffie and M. E. Hellman, «New directions in cryptography», *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976. [Online]. Available: https://doi.org/10.1109/TIT.1976.1055638.

[DK23] T. Decru and S. Kunzweiler, «Efficient computation of $(3^n, 3^n)$-isogenies», *IACR Cryptol. ePrint Arch.*, p. 376, 2023. [Online]. Available: https://eprint.iacr.org/2023/376.

[DPB17] J. Doliskani, G. C. C. F. Pereira, and P. S. L. M. Barreto, «Faster cryptographic hash function from supersingular isogeny graphs», *IACR Cryptol. ePrint Arch.*, p. 1202, 2017. [Online]. Available: http://eprint.iacr.org/2017/1202.

[FKL+20] L. D. Feo, D. Kohel, *et al.*, «Sqisign: Compact post-quantum signatures from quaternions and isogenies», in *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, S. Moriai and H. Wang, Eds., ser. Lecture Notes in Computer Science, vol. 12491, Springer, 2020, pp. 64–93. [Online]. Available: https://doi.org/10.1007/978-3-030-64837-4%5C_3.

[Gal12] S. D. Galbraith, *Mathematics of Public Key Cryptography.* Cambridge University Press, 2012. [Online]. Available: https://www.math.auckland.ac.nz/%5C%7Esgal018/crypto-book/crypto-book.html.

[Gam85] T. E. Gamal, «A public key cryptosystem and a signature scheme based on discrete logarithms», *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985. [Online]. Available: https://doi.org/10.1109/TIT.1985.1057074.

[GPST16] S. D. Galbraith, C. Petit, *et al.*, «On the security of supersingular isogeny cryptosystems», in *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, J. H. Cheon and T. Takagi, Eds., ser. Lecture Notes in Computer Science, vol. 10031, 2016, pp. 63–91. [Online]. Available: https://doi.org/10.1007/978-3-662-53887-6%5C_3.

[HHK17] D. Hofheinz, K. Hövelmanns, and E. Kiltz, «A modular analysis of the fujisaki-okamoto transformation», in *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, Y. Kalai and L. Reyzin, Eds., ser. Lecture Notes in Computer Science, vol. 10677, Springer, 2017, pp. 341–371. [Online]. Available: https://doi.org/10.1007/978-3-319-70500-2%5C_12.

[JBN14] S. Jain, P. Bhattacharya, and S. Nagpaul, *Basic Abstract Algebra.* Cambridge University Press, 2014. [Online]. Available: https://books.google.no/books?id=x7jXoQEACAAJ.

[JF11] D. Jao and L. D. Feo, «Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies», in *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, B. Yang, Ed., ser. Lecture Notes in Computer Science, vol. 7071, Springer, 2011, pp. 19–34. [Online]. Available: https://doi.org/10.1007/978-3-642-25405-5%5C_2.

[Kan97] E. Kani, «The number of curves of genus two with elliptic differentials.», vol. 1997, no. 485, pp. 93–122, 1997. [Online]. Available: https://doi.org/10.1515/crll.1997.485.93.

[Kuh88] R. M. Kuhn, «Curves of genus 2 with split jacobian», *Transactions of the American Mathematical Society*, vol. 307, no. 1, pp. 41–49, 1988. [Online]. Available: http://www.jstor.org/stable/2000749 (last visited: Jun. 1, 2023).

[Kun22] S. Kunzweiler, «Efficient computation of $(2^n, 2^n)$-isogenies», *IACR Cryptol. ePrint Arch.*, p. 990, 2022. [Online]. Available: https://eprint.iacr.org/2022/990.

[LL93] A. Lenstra and H. Lenstra, *The Development of the Number Field Sieve* (Lecture Notes in Mathematics). Springer, 1993. [Online]. Available: https://books.google.no/books?id=%5C_QBvFRP69xEC.

[LM95] R. Lercier and F. Morain, «Counting the number of points on elliptic curves over finite fields: Strategies and performance», in *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding*, L. C. Guillou and J. Quisquater, Eds., ser. Lecture Notes in Computer Science, vol. 921, Springer, 1995, pp. 79–94. [Online]. Available: https://doi.org/10.1007/3-540-49264-X%5C_7.

[Mil06] J. Milne, *Elliptic Curves.* BookSurge Publishers, 2006, pp. 238+viii.

[MM22] L. Maino and C. Martindale, «An attack on SIDH with arbitrary starting curve», *IACR Cryptol. ePrint Arch.*, p. 1026, 2022. [Online]. Available: https://eprint.iacr.org/2022/1026.

[Mon87] P. Montgomery, «Speeding the pollard and elliptic curve methods of factorization. mathematics of computation», vol. 48, no. 177, pp. 234–264, 1987.

[MZoW+96] A. Menezes, R. Zuccherato, *et al.*, *An Elementary Introduction to Hyperelliptic Curves* (CORR Report). Faculty of Mathematics, University of Waterloo, 1996, (Accessed on 06/14/2023). [Online]. Available: https://books.google.no/books?id=yxZYNAEACAAJ.

[NCG02] M. A. Nielsen, I. Chuang, and L. K. Grover, «Quantum Computation and Quantum Information», *American Journal of Physics*, vol. 70, no. 5, pp. 558–559, May 2002. [Online]. Available: https://doi.org/10.1119/1.1463744.

[NIST] PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates. [Online]. Available: https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4 (last visited: Aug. 11, 2022).

[OP22a] R. Oudompheng and G. Pope, «A note on reimplementing the castryck-decru attack and lessons learned for sagemath», *IACR Cryptol. ePrint Arch.*, p. 1283, 2022. [Online]. Available: https://eprint.iacr.org/2022/1283.

[OP22b] R. Oudompheng and G. Pope, «A note on reimplementing the castryck-decru attack and lessons learned for sagemath», *IACR Cryptol. ePrint Arch.*, p. 1283, 2022. [Online]. Available: https://eprint.iacr.org/2022/1283.

[Rob23] D. 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*, C. Hazay and M. Stam, Eds., ser. Lecture Notes in Computer Science, vol. 14008, Springer, 2023, pp. 472–503. [Online]. Available: https://doi.org/10.1007/978-3-031-30589-4%5C_17.

[SAJA21] H. Seo, M. Anastasova, *et al.*, «Supersingular isogeny key encapsulation (SIKE) round 2 on ARM cortex-m4», *IEEE Trans. Computers*, vol. 70, no. 10, pp. 1705–1718, 2021. [Online]. Available: https://doi.org/10.1109/TC.2020.3023045.

[Sho97] P. W. Shor, «Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer», *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997. [Online]. Available: https://doi.org/10.1137/S0097539795293172.

[SIKE] *Sike statement to nist*, https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/round-4/submissions/sike-team-note-insecure.pdf, (Accessed on 05/24/2023).

[Sil09] J. H. Silverman, *The Arithmetic of Elliptic Curves* (Graduate texts in mathematics). Dordrecht: Springer, 2009. [Online]. Available: https://cds.cern.ch/record/1338326.

[Smi06] B. Smith, «Explicit endomorphisms and correspondences», *Bulletin of The Australian Mathematical Society - BULL AUST MATH SOC*, vol. 74, Dec. 2006.

[SS14] T. J. Shimeall and J. M. Spring, «Chapter 8 - resistance strategies: Symmetric encryption», in *Introduction to Information Security*, T. J. Shimeall and J. M. Spring, Eds., Boston: Syngress, 2014, pp. 155–186. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781597499699000080 .

[Sto10] A. Stolbunov, *Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves*, 2010. [Online]. Available: /article/id/e8001706-6615-4b24-b499-8ea9d348dabb.

[Vél71] J. Vélu, *Comptes rendus de l'académie des sciences de paris, série a, t. 273*, https://aghitza.github.io/pdf/other/velu.pdf, (Accessed on 04/19/2023), Jul. 1971.

[Wes] B. Wesolowski, *Understanding and improving the castryck-decru attack on sidh*, https://www.dropbox.com/s/pmv3lrsg1gayl13/attacksidh.pdf?dl=0, (Accessed on 05/09/2023).

# Appendix A
# Finding Optimal $\sqrt{\text{é}lu}$ Degree

For finding an optimal threshold for when to use vélu's formula and $\sqrt{\text{é}lu}$ we conducted several tests where we computed all the different isogenies in $\mathbb{F}_{p^2}$. For each isogeny, we computed 100 and took the average computing time. All tests were conducted using a MacBook Air with an Apple M1 chip.

We did three different tests with different primes. The three different primes were:

$$
\begin{aligned}
&p = 2^{127} - 1\\
&p = 0x76042798BBFB78AEBD02490BD2635DEC131AB\\
&\quad FFFFFFFFFFFFFFFFFFFFFFFFFFF\\
&p = 18809883576148993975748257029181114827 34\\
&\quad 99283258225940944664269318258687
\end{aligned}
$$

For the first prime, Table A.1, we see that computing isogenies with degree $\geq 337$ are faster with $\sqrt{\text{é}lu}$. For the second prime, Table A.2, the only isogeny it paid off using $\sqrt{\text{é}lu}$ was 5419. We can set a temporary lower bound, we can observe that computing isogeny of degree 251 is not beneficial. For the last test, Table A.3, the 269-degree isogeny is borderline not faster using $\sqrt{\text{é}lu}$.

While computing isogenies take varying amounts of time, depending on the field we are working over, we can set a soft threshold of around 300. Using Table A.1 and A.3, we see the threshold is between $269 < \textit{threshold} < 337$.

| Calculating isogeny of degree $\ell$ in a field p = 255-bit | | | |
|---|---|---|---|
| Isogeny degree | Avrg time (Vélu) | Avrg time ($\sqrt{élu}$) | times faster |
| 19 | 0.002171 | 0.01894 | 0.1146 |
| 43 | 0.004890 | 0.02063 | 0.2370 |
| 73 | 0.008827 | 0.02181 | 0.4047 |
| 127 | 0.01479 | 0.02337 | 0.6329 |
| 337 | 0.03981 | 0.02931 | 1.358 |
| 5419 | 0.6603 | 0.09900 | 6.670 |
| 92737 | 12.08 | 0.4430 | 27.27 |

**Table A.1:** Table of how long computing an isogeny of degree $\ell$ takes with vélu's formula and $\sqrt{élu}$ over $p = 2^{127} - 1$.

| Calculating isogeny of degree $\ell$ in a field p = 253-bit | | | |
|---|---|---|---|
| Isogeny degree | Avrg time (Vélu) | Avrg time ($\sqrt{élu}$) | times faster |
| 107 | 0.01333 | 0.04469 | 0.2983 |
| 109 | 0.01424 | 0.04482 | 0.3177 |
| 131 | 0.01665 | 0.04489 | 0.3709 |
| 137 | 0.01742 | 0.04591 | 0.3794 |
| 197 | 0.02566 | 0.04785 | 0.5363 |
| 199 | 0.02531 | 0.04925 | 0.5139 |
| 227 | 0.02921 | 0.04809 | 0.6074 |
| 251 | 0.03265 | 0.05133 | 0.6361 |
| 5519 | 0.7344 | 0.1451 | 5.061 |

**Table A.2:** Table of how long computing an isogeny of degree $\ell$ takes with vélu's formula and $\sqrt{élu}$.

| Calculating isogeny of degree $\ell$ in a field p = 237-bit | | | |
|---|---|---|---|
| Isogeny degree | Avrg time (Vélu) | Avrg time ($\sqrt{élu}$) | times faster |
| 103 | 0.01297 | 0.03206 | 0.4046 |
| 269 | 0.034633 | 0.03550 | 0.9756 |
| 439 | 0.053026 | 0.04330 | 1.224 |

**Table A.3:** Table of how long computing an isogeny of degree $\ell$ takes with vélu's formula and $\sqrt{élu}$.

Appendix B

# Code for Algorithm

```python
# Function to find an optimal sequence
# Returns a list list of form [alpha_i, factor to remove, u, v]
def optimal_sequence(N, M):
    # Initialize variables: temporary M, recovered M, and results
    M_tmp = M
    recovered_M = 1
    results = [[-1,-1,-1,-1]]

    # Iterate while there is at least one prime factor of M_tmp
    while len([p for p, e in list(factor(M_tmp)) for _ in range(e)]) > 1:
        # Iterate over each prime factor combination of M_tmp
        for prime_factor in prime_combinations(M_tmp):
            # Multiply the current prime factor by the recovered_M
            recovered_M *= prime_factor

            # If recovered_M modulo 4 is not equal to 3
            # Then divide the recovered_M by the prime factor and continue the loop
            if recovered_M % 4 != 3:
                recovered_M /= prime_factor
                continue

            # Call the function find_lowest_alpha with parameters N and recovered_M
            # If the result is None, then continue the loop
            result = find_lowest_alpha(N, recovered_M)
            if result is None: continue

            # If there is a result, unpack the result into N_low, u, and v
            N_low, u, v = result
            # Divide the M_tmp by the prime factor
            M_tmp /= prime_factor
            # Append the result to the results list
            results[-1][1] = prime_factor
            results.append([N_low, -1, u, v])
            # Need to update factor of previous list such that u_v list is correct
            # Break the loop
            break
    # Return the results list
    results[-1][1] = M_tmp
    results.reverse()
    return results
```

**Listing B.1:** Code for finding the best sequence for attacking B-SIDH. M is Bob's order.

```
def find_lowest_alpha(N, M):
    for i in range(factor(N)[0][1] + 1):
        C = 2^i - M
        if C % 4 and C > 1:
            u_v = find_u_v(C)
            if u_v is not None:
                return i, u_v[0], u_v[1]
    return None

def prime_combinations(int):
    # Get the prime factors
    prime_factors = [p for p, e in list(factor(int)) for _ in range(e)]

    # Sort in descending order
    prime_factors.sort(reverse=True)

    # Copy the prime factors
    prime_factors_combinations = list(prime_factors)

    result = []
    while prime_factors_combinations:
        # Pop the largest prime factor
        largest_prime_factor = prime_factors_combinations.pop(0)
        # Generate all combinations of the largest prime factor and each of the other prime factors,
        # starting with the smallest, and append them to the result
        result.extend(largest_prime_factor * p for p in reversed(prime_factors_combinations))

    # Concatenate the prime factors and the combinations and return
    return prime_factors + result
```

**Listing B.2:** Two helper functions called find_lowest_alpha and prime_combinations.