Cryptanalysis of Isogeny-based Protocols in Genus 1 and 2

By

Charlotte Weitkämper

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

Cyber Security and Privacy Research Group
School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
June 2023

ABSTRACT

Isogeny-based cryptography is one of the contenders for providing cryptosystems based on mathematical problems which are assumed to be hard for both classical and quantum computers. The most general of these isogeny-related problems, the *pure isogeny problem*, is the task of finding an isogeny between any two given supersingular elliptic curves. Many variants of this problem exist - not all of which are actually hard both classically and quantumly. Some variants use special primes, require the found isogeny to be of a certain degree, provide additional torsion point information, use specific elliptic curves instead of arbitrary ones, or translate the problem to a higher-dimensional setting using genus-2 curves.

This thesis focuses on the cryptanalysis of encryption schemes using variants of the pure isogeny problem for their underlying hardness assumption. We provide several attacks on the Supersingular Isogeny Diffie–Hellman (SIDH) protocol and some variants thereof. Note that these results predate the recent remarkable full break of the SIDH protocol by Castryck and Decru, as well as others.

We first introduce a general attack framework using a malleability oracle to reduce inverting a one-way function with specific characteristics to quantumly solving a hidden shift problem. This framework can be instantiated to provide a quantum subexponential attack on SIDH with overstretched parameters by defining a group acting on subsets of the SIDH keyspace. Furthermore, we present adaptive attacks on two variants of the SIDH protocol which recover static secret keys by repeatedly sending malformed public information. The first protocol produces related key exchange instances from making use of non-trivial automorphisms existing on special elliptic curves. The second protocol is a variant using Jacobians of hyperelliptic genus-2 curves as well as elliptic curve products and isogenies between them. We conclude this thesis with the presentation of several new algorithms for computing isogenies between arbitrary supersingular elliptic curves of prescribed degree which, in most cases, require knowledge of the endomorphism rings. Making use of speed-ups obtained from quantum search and factoring algorithms, these methods result in an acceleration of the computation of certain isogenies.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisors and co-supervisors over the years, Christophe Petit, Tom Chothia and David Galindo. Thank you for your tireless support, continued guidance and inspiration. Christophe, thank you also for introducing me to the world of isogenies and isogenists, and for suggesting so many fascinating topics and research problems which provoked countless insightful discussions. Unfortunately, in the end, there was not time to explore them all.

I would also like to thank all my collaborators, fellow workshop attendees and project group members for teaching me not only how to approach (and often solve) a problem but also how to put results down on paper; thank you for answering all my isogeny-related questions ever since I started my PhD. Furthermore, to all reviewers of abstracts and papers, as well as the examiners of my thesis, thank you for your many questions, suggestions and ideas, and for guiding me in improving my work.

And finally, to my wonderful family and friends without whose encouragement this thesis would not exist, and to all my office and department mates who made the last few years all the more fun: thank you!

The research in this thesis was possible due to financial support in form of a studentship from the University of Birmingham and EPSRC.

Contents

List of Figures

List of Tables

Introduction

With the emergence of cryptographically-sized quantum computers expected in the not too distant future[1], the cryptographic community is faced with the fact that quantum algorithms, most famously those by Shor [Sho94] and Grover [Gro96], will most likely be able to find solutions to classically assumed hard problems such as computing discrete logarithms and integer factorisation much more efficiently than any classical algorithm. Most currently deployed asymmetric cryptographic schemes rely on such problems; for example, the popular encryption and signature schemes based on RSA [RSA78] are only secure as long as it is infeasible for any adversary to factor the composite numbers used in the protocols as moduli into their two very large prime factors. These hard problems will be efficiently solvable once a sufficiently powerful quantum computer, i.e. one with a large enough number of qubits and reliable fault-tolerance, is built.

This calls for public key cryptography algorithms relying on alternative hard problems which, in addition to providing security in the classical setting as was previously sufficient, can also withstand attacks by adversaries with access to powerful quantum computers. Hence, solutions that provide security even in the future presence of such quantum computers need to be found. It is imperative to understand that as some applications require long-term security, protecting today's communication from the future capabilities of quantum comput-

[1]Note that this could mean within the next years, a decade or multiple decades - opinions on a timeline vary between experts (see for example [Mos18; MP22]) - but it is a question of when more than if.

ing is already a necessity.

The search for post-quantum cryptographic protocols has been majorly advanced by the call for submission of such quantum-secure solutions to a 'Post-Quantum Cryptography Standardization' project [NIST16] lead by the US American National Institute of Standards and Technology (NIST). The current and former candidates for standardisation can be classed largely into lattice-based, code-based, multivariate, hash-based as well as isogeny-based protocols for key encapsulation mechanisms and signature schemes.

As isogeny-based schemes we consider cryptographic protocols whose security relies mostly on the hardness of mathematical problems involving the computation of isogenies between principally polarised abelian varieties, where isogenies are surjective morphisms with finite kernel between two such varieties. Traditionally, only isogenies between specific types of elliptic curves which are principally polarised abelian varieties of dimension one were considered. The polarisation is not usually of concern since isogeny-based cryptography is mostly instantiated with isogenies of elliptic curves which naturally preserve principal polarisations. Hence, principal polarisation is often ignored or disregarded in this setting. Technically, it means that there exists an isogeny between the abelian variety and its dual variety, the Picard group of degree-zero divisor classes, which is also an isomorphism. However, this polarisation gains more practical importance when one considers the natural generalisation of isogeny-based cryptography by broadening the scope to isogenies between higher-dimensional varieties. This development has recently received more attention. Hereby, the main focus lies on the cryptographic uses of curves of genus 2 and their Jacobians, which are principally polarised abelian surfaces, and isogenies between them.

The main computational problem underlying this branch of post-quantum cryptography is the *pure isogeny problem*. It consists of finding an isogeny $\varphi : E \to E'$ between two given supersingular isogenous elliptic curves E and E' over some finite field $\mathbb{F}_q$, which

can be translated into a path-finding problem in a type of expander graph. Often, schemes utilising isogenies have additional requirements on φ and for security rely on variants of this most general problem. For example, breaking the most well-known scheme using isogenies, the *Supersingular Isogeny Diffie–Hellman* key exchange [JD11], entails computing a party's secret isogeny φ having a specific prime-power degree and additionally exhibiting a certain action on a torsion subgroup of E. The details of this additional information are provided to an attacker through a party's public message in the active phase of the key exchange. SIDH was a post-quantum protocol based on the presumed hardness of computing an isogeny of known degree between two supersingular elliptic curves given some additional torsion point information. After successful active attacks, the then believed to be IND-CCA secure Supersingular Isogeny Key Encapsulation (SIKE) mechanism [JACCDHJKLLNRSU17] obtained from SIDH by using a transformation initially due to Fujisaki and Okamoto [FO99; Den03; HHK17] was selected by NIST as an alternate candidate in 2022 for standardisation. It was found to be insecure shortly after due to a very effective, passive classical attack. This new class of attacks is due to Castryck and Decru [CD23] and concurrent work by Maino, Martindale, Panny, Pope and Wesolowski [MM22; MMPPW23] as well as Robert [Rob23], and fully recovers a secret key in SIDH. However, this strategy only allows one to find a secret isogeny when enough torsion information is given; no efficient method exists to date which solves the pure isogeny problem of finding any isogeny between two arbitrary given supersingular elliptic curves. Hence, this result only eliminates a small number of isogeny-based protocols from consideration for post-quantum cryptography. Many isogeny-based schemes establishing a variety of different primitives remain unaffected, including the CSIDH (Commutative SIDH) key exchange [CLMPR18] and the SQISign [DKLPW20] signature.

In comparison to post-quantum schemes based on other mathematical problems, isogeny-based protocols often offer more efficient communication and less bandwidth requirements due to relatively small key sizes. However, the computation times of most

schemes are often not competitive, and the comparative youth of the field with respect to other disciplines of quantum-safe cryptography has lowered the community's confidence in these schemes. A further setback to isogeny-based cryptography has been the recent break of SIKE, the only isogeny proposal in the "Post-Quantum Cryptography Standardization" project by NIST [NIST16]. It is important to highlight that the SIDH attacks only impact the security of a small proportion of isogeny-based protocols which should restore some trust in isogeny-based cryptography in the wider cryptographic community. Nonetheless, these attacks provide new tools for constructing further protocols, and there are still many promising schemes which offer a variety of cryptographic primitives and are unaffected by these results. Many of these have been first introduced after the initial NIST call and thus have not enjoyed the same popularity and scrutiny as SIDH and SIKE had as part of the process of the standardisation project. It is imperative to keep working on improving efficiency of such protocols as well as continue to assess and test their security so that they can provide competitive solutions to the post-quantum challenge.

In this thesis we investigate the classical and quantum security of several protocols relying on isogeny-based mathematical problems. In particular, we present a quantum subexponential attack on SIDH for a specific set of parameters, devise classical adaptive attacks on variants of the SIDH protocol, one using elliptic curves and one using Jacobians of genus-2 curves, and also examine the problem of finding an isogeny between arbitrary curves of prescribed degree. We have organised the presentation of mathematical and cryptographic background material as well as our findings as follows.

Outline of this thesis

In **Chapter 1** we begin by introducing the mathematical background necessary to understand and work with isogeny-based cryptography, focusing on abelian varieties in dimensions one and two, and isogenies between a pair of such elliptic curves or abelian surfaces.

Chapter 2 will then dive more deeply into how isogenies between elliptic curves or abelian surfaces can be utilised to design cryptographic schemes such as hash functions and key exchange protocols. Literature relevant to the field and the schemes we focus on in the proceeding chapters are presented; this, in particular, includes descriptions of the SIDH, G2SIDH and Jao–Urbanik key exchange schemes.

We present our first contribution, the results of collaborative work with Péter Kutas, Simon-Philipp Merz and Christophe Petit, published as [KMPW21], in **Chapter 3**. This entails the proposal of a new attack framework which can be utilised to invert one-way functions in quantum subexponential time. By requiring access to a malleability oracle with respect to some commutative group action, we succeed in reducing the computation of inverses to an instance of the hidden shift problem for which (several) subexponential quantum algorithms [Kup05; Kup13] exist. The setting in which we deploy this attack is overstretched SIDH. This means we manage to recover a secret kernel generator of one of the parties in SIDH where the torsion sizes are unbalanced and also larger than in practice. These restrictions allow us to construct a group of fixed-degree endomorphisms of the starting curve acting on certain candidate kernel subgroups which themselves can be mapped to their corresponding codomain curve. An attacker can use the public torsion information to evaluate this action, modulo some heuristic assumptions, and can then reduce finding the correct kernel to solving a hidden shift instance. This application is a new way to exploit the additional torsion point information provided in SIDH and, though SIDH is now classically broken, was one of the first quantum attacks on the protocol at the time. It showed that even though supersingular elliptic curve endomorphism rings are non-commutative, SIDH was not immune to such attacks. Though we instantiate our attack with instances derived from isogeny-based cryptography and in particular SIDH, the framework could not only be of interest when examining the security of newly proposed isogeny-based key exchanges such as FESTA [BMP23] or M(D)-SIDH [FMP23], but also further other cryptanalytic efforts to

achieve more efficient quantum attacks.

Chapter 4 contains cryptanalysis of the Jao–Urbanik protocol [UJ20], collaborative work with Andrea Basso, Péter Kutas, Simon-Philipp Merz and Christophe Petit which was published as [BKMPW20]. The scheme in question is a variant of the static-static k-SIDH key exchange by Azarderaksh, Jao and Leonardi [AJL18] where the existence of non-scalar automorphisms on two special supersingular elliptic curves is exploited to generate multiple shared SIDH instances from a single pair of the two involved parties' keys. In our adaptive attack on the Jao–Urbanik key agreement, we manage to exploit exactly the structure underlying the protocol's evolvement from k-SIDH, thus voiding its efficiency improvement even though larger parameters were suggested to account for the possibility of such attacks. We achieve this by utilising the relationship between kernel subgroups appearing in these related SIDH instances instead of applying similar active attacks on SIDH and k-SIDH [GPST16; DGLTZ20] straightforwardly, or as straightforwardly as possible, and thus significantly reduce the number of oracle queries necessary to recover a secret key generating an isogeny of degree a power of two, or another small prime ℓ.

Chapter 5 focuses on examining the practicality and security of the genus-2 variant of the SIDH protocol (G2SIDH) as proposed by Flynn and Ti [FT19; Ti19]. The results in this chapter are based on collaborative work with Sabrina Kunzweiler and Yan Bo Ti, and have been published as [KTW21]. The contributions of this chapter are twofold: Firstly, we suggest that a special type of basis, a symplectic one, is used for computing G2SIDH key exchanges. This allows us to provide a classification of generators of (ℓ^n, ℓ^n)- and $(\ell^n, \ell^k, \ell^{n-k})$-isogeny kernel subgroups which appear in G2SIDH and results in a new sampling algorithm for secret keys. If a slight restriction of the original key space is acceptable (as it was for standard elliptic curve SIDH), this improves secret generation in comparison with the original proposal by avoiding having to solve several linear congruences and furthermore providing uniform sampling. Secondly, we present a polynomial-time adaptive attack on the protocol

which recovers a static key through an oracle providing one bit of information per query. This attack is not detectable by the standard checks performed to recognise maliciously generated torsion information and is described in the case when the attacked party uses a symplectic basis. However, should the key exchange be instantiated with an arbitrary torsion basis, the attack is shown to be translatable to this alternative setting.

The more general problem of computing an isogeny φ of a specified degree d between two given arbitrary but d-isogenous supersingular elliptic curves without knowledge of any additional information (such as the action of the isogeny on specific points as was the case in the previous chapters) is examined in **Chapter 6**. In particular, we utilise the speed-ups provided by quantum routines for search and factoring [Gro96; Sho94] to improve upon the current best (quantum) algorithm for this problem by developing an algorithm to find φ. This algorithm is heuristic quantum-polynomial time for certain parameters, given that the endomorphism rings of the domain and codomain curves are known. This is achieved by computing a connecting ideal between the endomorphism rings and solving a norm equation therein. We present several different approaches to obtaining an integral solution to this quadratic in four variables, all are based on algorithms by Cornacchia [Cor08], Coppersmith [Cop96a] and subsequent variants, and thus provide an approach for a broader set of parameters than was feasible until now. The results of this chapter are based on collaborative efforts together with Benjamin Benčina, Péter Kutas, Simon-Philipp Merz, Christophe Petit and Miha Stopar which have been submitted for review under the title *Improved quantum algorithms for finding fixed-degree isogenies between supersingular elliptic curves*.

We conclude in **Chapter 7** with some final remarks and address the question of what will become of isogeny-based cryptography, now that the arguably most well-known and popular protocol in the field has been broken. We will further discuss several open problems and future research directions relating to the contents of this thesis.

Contributions

We briefly summarise the contributions of this thesis.

- Building a framework for inverting malleable one-way functions in quantum subexponential time: Quantum algorithms like Kuperberg's [Kup05; Kup13] can solve instances of the hidden shift problem. This means that if we can reduce a problem occurring in isogeny-based cryptography to solving a hidden shift instance, we obtain a quantum subexponential attack. We provide conditions on the one-way function and an abelian group acting on its domain in order to compute inverses. Though SIDH inherently exhibits non-commutative characteristics, we artificially construct a setting from an instance of the protocol using overstretched parameters in which we succeed to compute the kernel of a secret isogeny given the domain and codomain curves as well as the action on a large torsion subgroup.

- Cryptanalysis of SIDH-based non-interactive key exchanges: A NIKE is a cryptographic primitive which requires both participating parties to have published their public key in advance so that they can obtain a shared secret key at any point without further communication between them. We show that two such protocols making use of the SIDH construction, the Jao–Urbanik NIKE [UJ20] using supersingular elliptic starting curves which have non-trivial automorphisms and the static-key version of G2SIDH [FT19] which is the natural generalisation of SIDH to using Jacobians of hyperelliptic genus-2 curves, are vulnerable to an adaptive attack in this setting. In fact, the decryption oracles utilised in these attacks can be viewed as oracles for the respective version of the decisional Diffie–Hellman oracles, and the adaptive attack then proves that the underlying computational isogeny problems can be reduced to the corresponding decisional problems in both cases.

- Analysis of maximal ℓ^n-isotropic subgroups of $J[\ell^n]$: It is known that these subgroups are exactly the kernels of isogenies which are chains of (ℓ, ℓ)-isogenies. We show that, given a set of arbitrary generators of a subgroup in terms of a fixed symplectic basis of the ℓ^n-torsion, we can normalise the generators and thus obtain canonical generators. This strategy results in a better understanding of such torsion subgroups and in particular their classification into different types. Furthermore, our analysis leads to an improved key generation algorithm for G2SIDH which allows for uniform sampling from the keyspace.

- Computing isogenies of specific degree between two given supersingular elliptic curves: The hardness of this computational problem is crucial to the security of many isogeny-based cryptosystems. In particular, secret keys of such schemes often consist of a secret isogeny, or the corresponding kernel subgroup, while the domain and codomain curves as well as the degree of the isogeny are public information. Thus, finding methods to efficiently compute isogenies satisfying the provided constraints could detrimentally reduce the security of such schemes. We provide improved quantum algorithms which utilise different classical strategies for solving quadratic multivariate equations over the integers once the endomorphism rings of the given curves have been computed using the method from [EHLMP20] with quantum speed-ups from Grover's algorithm [Gro96]. Our algorithms provide a more efficient technique for finding fixed-degree isogenies for certain types of parameters.

Part I

Background

CHAPTER 1

Abelian varieties & isogenies in dimensions one and two

Throughout this thesis we will study cryptographic schemes which can be built from properties of elliptic curves and Jacobians of hyperelliptic genus-2 curves, and how these schemes are vulnerable to both active and passive attacks. To obtain successful cryptanalytic results it is indispensable to have a thorough understanding of abelian varieties and their properties. We will give a brief introduction to the topic of abelian varieties of low dimension, their isogenies and endomorphisms mainly following expositions in [Mum70; Mil86] on general abelian varieties and [Sil09; Voi18] on elliptic curves. We refer the reader to these fundamental texts for further detail.

Many of the following statements can be made generally for abelian varieties over algebraically closed fields. With regards to our interest in applications in isogeny-based cryptography, we focus particularly on abelian varieties which are defined over finite fields (or their algebraic closure), i.e. fields of prime characteristic p, such that $p > 3$.

Notation and terminology Throughout this thesis, we will use the following notation. For two real valued functions f, g with $g(x)$ positive for large enough values, we write $f(x) = O(g(x))$ (as $x \to \infty$) if there exists some constants $c > 0$ and $N_c > 0$ such that $|f(x)| \leq cg(x)$ whenever $x \geq N_c$. Hence, $O(\text{poly}(x))$ indicates that a quantity is asymptotically upper bounded by a polynomial in x. Furthermore, we define $f(x) = o(g(x))$ (as $x \to \infty$) to mean that for all $\epsilon > 0$ there exists an integer N_ϵ such that $|f(x)| \leq \epsilon g(x)$ whenever $x \geq N_\epsilon$. We call an algorithm *efficient* if the execution time is bounded by a polynomial in the security parameter of the underlying cryptographic scheme. Sometimes, we may want to omit factors polynomial in $\log p$, where p is the characteristic of the finite field we are working with. In those cases, we abbreviate $O(g \cdot \text{polylog}\, p)$ by $O^*(g)$. We say that an algorithm is *subexponential* when it runs in time $2^{o(x)}$ where x is the length of the input. We call a function $\mu : \mathbb{N} \to \mathbb{R}$ *negligible* if for every positive integer c there exists an integer N_c such that $|\mu(x)| < x^{-c}$ for every $x > N_c$. We call an integer B-*smooth*, if it only has prime factors smaller than B. When $B \ll n$, we sometimes say that the integer is "smooth", meaning that its smoothness bound B is in $O(\text{poly}(\log n))$. If we say that we have *oracle access* to some given function, we understand this as being able to evaluate the function efficiently for any element in its domain of definition. More precisely, we consider the oracle to be a black box such that a query to the oracle with a genuine input element produces the corresponding evaluation of the function.

1.1 Abelian varieties and their isogenies

An abelian variety is a projective algebraic variety which admits a group law defined by regular morphisms of varieties. By convention, we write this group law additively. More precisely, let K be an algebraically closed field with $\text{char}\, K \neq 0$. An abelian variety is then defined as follows.

Definition 1.1 (Abelian variety)**.** *An* abelian variety *over K is a complete group variety A over K, that is, A is a variety with morphisms $+ : A \times A \to A$ and $- : A \to A$ and contains an element $\mathcal{O} \in A(\overline{K})$ such that $A(\overline{K})$ under $+$ and inversion $-$ forms a group with identity element $\mathcal{O}$.*

As the inversion map $-$ is a homomorphism on $A(\overline{K})$, the group law on an abelian variety is commutative. For any integer N, we can define the *multiplication-by-N* map, written as $[N]$, on A which adds N copies of an element in A (and if $N < 0$ then negates the result). This allows us to define the *N-torsion subgroup* $A[N]$ of A as the kernel of the map $[N]$, or alternatively, the set of all elements of the variety having order dividing N. The set $A[N]$ forms a subgroup of A and its group structure depends on the genus g of A, since for N coprime to $\operatorname{char} K$, we have

$$A[N] := \{P \in A(\overline{K}) \mid [N]P = \mathcal{O}\} \cong (\mathbb{Z}/N\mathbb{Z})^{2g}.$$

An *isogeny* is a morphism $\varphi : A \to A'$ of abelian varieties which is surjective and has finite kernel. We call two abelian varieties *isogenous* if there exists an isogeny between them. The *degree* of an isogeny φ is then defined as the degree of the field extension $K(A)/K(A')$ defined via the embedding of the function field $K(A')$ into $K(A)$ induced by the surjectivity of φ. We say that an isogeny is *separable* or *(purely) inseparable* if this field extension is separable or (purely) inseparable. If φ is separable, we can alternatively define $\deg \varphi$ as the order of $\ker \varphi$ when considered as a finite group scheme. Each isogeny φ can be factored into a separable part $\varphi_1 : A \to B$ and a purely inseparable part $\varphi_2 : B \to A'$ such that $\varphi = \varphi_1 \circ \varphi_2$. Note that multiplication-by-N is a separable isogeny as long as N is coprime to $\operatorname{char} K$, while the *Frobenius endomorphism* π on an abelian variety $A/\mathbb{F}_q$ which raises points coordinatewise to the q-th power has degree q^g and is purely inseparable. To each separable isogeny $\varphi : A \to A'$, we can associate a (unique) *dual isogeny* $\hat{\varphi} : A' \to A$ satisfying

$\deg \varphi = \deg \hat{\varphi}$ and that composing φ and $\hat{\varphi}$ results in the multiplication-by-$\deg \varphi$ map on A and A' respectively.

For the following exposition, we usually only consider separable isogenies and can hence use the fact that $|\ker \varphi| = \deg \varphi$. To uniquely identify such isogenies, one can utilise a result presented in [Mum70, Theorem 4]: For an abelian variety A, there exists a one-to-one correspondence between finite subgroups $G \subset A$ and separable isogenies $\varphi : A \to A/G$, up to post-composition with an isomorphism. Hence, instead of writing out an isogeny as a function on the elements of A, we often simply specify its kernel as a finite subgroup of A. Let us now consider $\mathrm{Hom}(A, A')$ for two abelian varieties A and A', i.e. the group of morphisms $A \to A'$ which are also group homomorphisms. Then $\mathrm{Hom}(A, A')$ is a free abelian group with finitely many generators, and we denote $\mathrm{Hom}(A, A)$ by $\mathrm{End}(A)$, which is the endomorphism ring of A.

1.2 Elliptic curves

In this section we will only focus on elliptic curves, which are smooth non-singular projective curves of genus one, and hence one-dimensional abelian varieties. To instantiate cryptographic protocols with elliptic curves and their isogenies, we need to be able to perform computations explicitly. Hence, one is often introduced to elliptic curves as the solution set to a (projective) algebraic plane curve defined by a polynomial of degree 3.

Let K be a finite field of prime characteristic p with q elements, i.e. $K = \mathbb{F}_q$ for q a power of $p > 3$. Then an elliptic curve E over $\mathbb{F}_q$ defined by $Y^2Z = X^3 + AXZ^2 + BZ^3$ for some $A, B \in \mathbb{F}_q$ can be written in so-called *short Weierstraß form* [1] as $y^2 = x^3 + Ax + B$.

[1] Note that there are many other ways to parametrise elliptic curves instead of using the (short) Weierstraß form as introduced here. Depending on the context in which a curves is considered, a certain curve form might be beneficial. In particular, Montgomery curves and Edwards curves provide convenient and efficient algorithms for certain arithmetic operations.

The $\mathbb{F}_q$-points on E are then

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q^2 \mid y^2 = x^3 + Ax + B\} \cup \{\mathcal{O}\},$$

where $\mathcal{O}$ is the *point at infinity* denoting the point $(X : Y : Z) = (0 : 1 : 0)$ on the associated projective curve.

The inherent group law on $E(\mathbb{F}_q)$ with identity element $\mathcal{O}$ can also be geometrically constructed on the set of points on E under the "chord-and-tangent rule", however, we forgo a precise description of this procedure here[2]. To categorise elliptic curves, one associates the *j-invariant* defined as

$$j(E) = 1728\frac{4A^3}{4A^3 + 27B^2}$$

to a curve E. Note that this fraction is well-defined since elliptic curves are non-singular. There exists an isomorphism $f : E \to E'$ between two curves E and E' precisely when $j(E) = j(E')$. We often omit equivalence up to isomorphism for elliptic curves, and thus usually consider isomorphism classes of elliptic curves identified by j-invariants instead of one specific curve in a class.

Reframing the definition of an isogeny for elliptic curves, we consider non-constant rational maps $\varphi : E \to E'$ between two elliptic curves E and E' defined over $\mathbb{F}_q$ which are also group homomorphisms from $E(\mathbb{F}_q)$ to $E'(\mathbb{F}_q)$. This means in particular, that $\varphi(\mathcal{O}) = \mathcal{O}$. Then two curves are *isogenous* if there exists an isogeny between them, which happens if and only if $|E(\mathbb{F}_q)| = |E'(\mathbb{F}_q)|$ [Tat66]. The degree of the isogeny φ is then its degree as a rational map. When we wish to compute an isogeny or evaluate it on specific curve points, the degree plays an important role. For example, the well-known formulae by Vélu [Vél71] provide an algorithm for computing the isogeny from its kernel points. This algorithm produces an expression of the isogeny as a rational map in polynomial time if its degree is

[2] A detailed description can be found in [Sil09, Section III.2].

smooth. Therefore, finding an isogeny is often equated to finding its corresponding kernel subgroup, as the isogeny can then be efficiently computed and evaluated. The state of the art on finding isogenies between two given elliptic curves is presented later in Section 6.1.2.

Let ℓ be a prime different from p, and $e \in \mathbb{Z}^+$. Then one can define the *Weil pairing* $e_{\ell^e} : E \times E \to \boldsymbol{\mu}_{\ell^e}$, where $\boldsymbol{\mu}_{\ell^e}$ denotes the set of primitive ℓ^e-th roots of unity, and this pairing is bilinear, alternating and nondegenerate. For a constructive definition of this pairing, we refer the reader to [Sil09, Section III.8]. We often make use of the following property of the Weil pairing which follows from [Sil09, Proposition 8.2]: for an isogeny $\varphi : E \to E'$ of elliptic curves and points $P, Q \in E[\ell^e]$, it holds that

$$e_{\ell^e}(\varphi(P), \varphi(Q)) = e_{\ell^e}(P, Q)^{\deg \varphi}, \tag{1.1}$$

where the first Weil pairing is computed on E' and the second one on E.

The following lemma [Sil09, Chapter III, Corollary 4.11] is a useful result describing how the isogenies corresponding to two subgroups can be related if one subgroup contains the other:

Lemma 1.2. *Let E, E_1 and E_2 be elliptic curves and let $\phi_1 : E \to E_1$ and $\phi_2 : E \to E_2$ be two non-constant isogenies such that $\ker \phi_1 \subseteq \ker \phi_2$ and ϕ_1 is separable. Then there is a unique isogeny $\phi : E_1 \to E_2$ such that $\phi_2 = \phi \circ \phi_1$.*

Recall that an isogeny from E to itself is called an *endomorphism*. Together with the zero map, endomorphisms of E form a ring under addition and composition denoted by $\mathrm{End}(E) := \mathrm{Hom}(E, E)$. The structure of the ring $\mathrm{End}(E)$ allows us to partition the set of all elliptic curves into two parts: The first consisting of *ordinary* curves which, when considered over a finite field, are characterised by having an endomorphism ring which is an order in an imaginary quadratic number field, and hence commutative; the second encompassing the

remaining curves which all have non-commutative endomorphism rings isomorphic to orders in a quaternion algebra. The latter elliptic curves are called *supersingular*.

Remark 1.3. *Note that for a supersingular elliptic curve E defined over $\overline{\mathbb{F}}_p$, we always have $j(E) \in \mathbb{F}_{p^2}$. Since there also always exists an $\overline{\mathbb{F}}_p$-isomorphism to a supersingular elliptic curve which is defined over $\mathbb{F}_{p^2}$, we can consider all such elliptic curves to be defined over $\mathbb{F}_{p^2}$ (up to isomorphism).*

1.2.1 Supersingular endomorphism rings

Since the remainder of this thesis focuses strongly on supersingular elliptic curves, we now present some properties of their endomorphism rings. More precisely, the endomorphism rings of supersingular elliptic curves defined over fields of characteristic p are (isomorphic to) maximal orders in the quaternion algebra ramified at p and at infinity. The following definitions of quaternion algebras and their (maximal) orders will give more insight into the resulting implications.

Quaternions

For p the characteristic of our base field, we set

$$(a,b) = \begin{cases} (-1,-1) & \text{if } p = 2, \\ (-1,-p) & \text{if } p \equiv 3 \pmod 4, \\ (-2,-p) & \text{if } p \equiv 5 \pmod 8, \\ (-r,-p) & \text{if } p \equiv 1 \pmod 8, \end{cases}$$

where $r \equiv 3 \pmod 4$ is a prime that is not a square modulo p. Then the quaternion algebra $(a, b \mid \mathbb{Q})$ is defined to be the four-dimensional $\mathbb{Q}$-algebra spanned by $1, i, j, k$ with $i^2 = a$ and $j^2 = b$, and where we denote ij by k. These basis elements follow the non-commutative multiplication rule $k = ij = -ji$. By our choice of a and b, this quaternion algebra is the

quaternion algebra ramified at p and ∞[3] which we denote by $B_{p,\infty}$. Let $a_1, \ldots, a_4 \in \mathbb{Q}$, and consider the quaternion $\alpha = a_1 + a_2 i + a_3 j + a_4 k$. Then we always have an involution

$$\alpha \mapsto \overline{\alpha} = a_1 - a_2 i - a_3 j - a_4 k.$$

The *reduced trace* of an element $\alpha \in B_{p,\infty}$ is defined as $\mathrm{tr}(\alpha) := \alpha + \overline{\alpha}$ and its *reduced norm* is $\mathrm{Norm}(\alpha) := \alpha\overline{\alpha}$. Recall that an order $\mathcal{O} \subset B_{p,\infty}$ is a $\mathbb{Z}$-submodule of $B_{p,\infty}$ which is also a subring, i.e. we have $1 \in \mathcal{O}$ and the order is closed under addition and multiplication. We call an order *maximal* if it is not properly contained in any other proper, i.e. non-trivial, order.

The Deuring correspondence

The so-called *Deuring correspondence* establishes a categorical equivalence between certain objects from algebraic geometry and notions relating to quaternion algebras, providing an important tool for linking computations of elliptic curve isogenies with quaternion computations. More precisely, Deuring [Deu41] showed that there is an equivalence of categories of supersingular elliptic curves over $\overline{\mathbb{F}}_p$, together with the isogenies between them, and the maximal orders in $B_{p,\infty}$, together with the left ideals of the orders. This implies a bijection between conjugacy classes of supersingular j-invariants and maximal orders (up to isomorphism), and this mapping consists of identifying a supersingular elliptic curve E with its endomorphism ring $\mathrm{End}(E)$. Furthermore, if we let $E/\mathbb{F}_q$ and $E'/\mathbb{F}_q$ be supersingular elliptic curves with endomorphism rings $\mathrm{End}(E)$ and $\mathrm{End}(E')$ respectively which are isogenous via $\varphi : E \to E'$, the tuple (E', φ) is mapped to an integral left $\mathrm{End}(E)$-ideal I with right order isomorphic to $\mathrm{End}(E')$. The ideal I is called a *connecting ideal* of $\mathrm{End}(E)$ and $\mathrm{End}(E')$. By [Voi18, Lemma 42.1.11], the set of isogenies from E to E', $\mathrm{Hom}(E, E')$, is a left $\mathrm{End}(E)$-module and a right $\mathrm{End}(E')$-module, and forms a $\mathbb{Z}$-lattice of rank 4. In particular, this

[3] This means that $B_\nu := B_{p,\infty} \otimes_{\mathbb{Q}} \mathbb{Q}_\nu$ is a division algebra only for $\nu = p, \infty$, and a matrix ring over $\mathbb{Q}_\nu$ everywhere else. More information on ramified and unramified places can be found in [Voi18, Section 14.1].

implies that finding an isogeny between two supersingular isogenous elliptic curves E and E' can be solved by finding a corresponding quaternion in the $\mathbb{Z}$-lattice $\mathrm{Hom}(E, E')$. This relationship is explored for example in [KLPT14; Wes22a] and [EPSV23].

1.2.2 Isogeny graphs

Fix some prime ℓ distinct from $p = \mathrm{char}\, K$. Then consider all elliptic curves over K up to isomorphism, i.e. classes of isomorphic curves represented by their j-invariants. Then the ℓ-isogeny graph G_ℓ is the graph with vertices corresponding to the j-invariants, where two vertices are joined by an edge exactly when the j-invariants represent isomorphism classes of ℓ-isogenous curves. Note that this defines an undirected graph[4] as the existence of an ℓ-isogeny between two (isomorphism classes of) curves in one direction implies the existence of the dual isogeny of the same degree in the opposite direction. In particular, non-backtracking walks of length e in the ℓ-isogeny graph correspond to ℓ^e-isogenies between the curves corresponding to the starting and end node. This graph consists of multiple components, where each component either only includes nodes of ordinary elliptic curves or supersingular elliptic curves. Depending on which type of curves are displayed in a (part of the) isogeny graph, the graph takes on different structures. For example, j-invariants of ordinary curves together with the isogenies between them form a "volcano-like" graph [Koh96]. Meanwhile, supersingular elliptic curves yield $(\ell + 1)$-regular ℓ-isogeny graphs exhibiting properties of expander graphs; see e.g. [ACLLNSS23] for more details. This structure has given rise to interesting cryptographic constructions, some of which will be explored in more detail in the next chapter.

[4] More precisely, the graph is undirected nearly everywhere. Exceptions occur at nodes denoting elliptic curves with non-trivial automorphisms.

1.3 Principally polarised abelian surfaces

In the remainder of this chapter, we will focus on abelian surfaces, i.e. abelian varieties of dimension two, and discuss isogenies more explicitly in this setting.

Again, let p and ℓ be different primes, q a power of p, and consider e to be a positive integer. We now focus on abelian surfaces defined over the finite field $\mathbb{F}_q$. Let A be such a surface over a characteristic-p field. Firstly, we can define a notion of supersingularity in the dimension-2 setting. In particular, we consider an abelian surface $A/\mathbb{F}_q$ to be *supersingular* if there exists an isogeny from A to a product of supersingular elliptic curves over $\mathbb{F}_q$. Further, the subset of *superspecial* abelian surfaces is then characterised by surfaces $A/\mathbb{F}_q$ in fact being isomorphic to this product of supersingular elliptic curves. Since we are interested in finding the "correct", most useful higher-dimensional analogue of a supersingular elliptic curve to utilise for cryptographic purposes, we cannot simply work with supersingular or superspecial abelian surfaces on their own but always have to consider a surface A together with its canonical principal polarisation. In other words, we are mainly interested in *principally polarised abelian surfaces* (PPAS), and often restrict our consideration to those which are furthermore superspecial (PPSSAS). Let $A^\vee$ denote the dual variety of A. Then we call an isogeny $\phi_\mathcal{L} : A \to A^\vee$ induced by some ample divisor $\mathcal{L}$ of A a *polarisation* of A, and say it is principal when this isogeny is furthermore an isomorphism between the two varieties. Formally, a PPSSAS is hence given by a tuple $(A, \mathcal{L})$ or $(A, \phi_\mathcal{L})$. However, since we only consider principally polarised A in the following, we usually omit specifying the polarisation or its induced isogeny.

Principally polarised superspecial abelian surfaces can be divided into two different types (cf. [Wei57] on results by Torelli). For $A/\overline{\mathbb{F}}_p$, we either have $A \cong J$ where J denotes the Jacobian of a smooth hyperelliptic genus-2 curve, or that A is isomorphic to an elliptic curve product $E_1 \times E_2$. Therefore, it is possible to explicitly describe the surface A as either

being associated to the solution set of the equation $C : y^2 = f(x)$ where f is a degree-5 or degree-6 polynomial over $\overline{\mathbb{F}}_p$ representing a genus-2 curve, or being expressed via the equations $y^2 = g(x)$ and $y^2 = h(x)$ for some polynomials g and h of degree 3 which each define an elliptic curve. This means that in practice, A is either parametrised componentwise in terms of its elliptic curve factors and tuples of elliptic curve points are taken as elements of A, or otherwise one works with degree-0 divisors of C, which are often written in Mumford representation for convenience.

If the PPSSAS is an elliptic curve product, we represent its $\overline{\mathbb{F}}_q$-isomorphism class by the (unordered) pair of j-invariants of its elliptic curve factors. The $\overline{\mathbb{F}}_q$-isomorphism classes of hyperelliptic curve Jacobians are often represented by their (weighted projective) Igusa invariants. While we often only consider the absolute Igusa invariants consisting of an ordered triple $(j_1, j_2, j_3) \in \overline{\mathbb{F}}_q^3$ for cryptographic purposes and in this thesis, a rigorous definition of the generalisation from the (absolute Weierstraß) j-invariant can be found in [Igu60].

1.3.1 Isogenies of PPAS

In order to define a practical notion of isogenies between principally polarised abelian surfaces, we will look at the kernel subgroups defining such isogenies. A particular condition on these subgroups is dependent on the non-degenerate, alternating *Weil pairing* which exists on any abelian surface $A/\mathbb{F}_q$. This pairing is a map

$$e_{\ell^e} : A[\ell^e](\overline{\mathbb{F}}_q) \times A^\vee[\ell^e](\overline{\mathbb{F}}_q) \to \boldsymbol{\mu}_{\ell^e}$$

where $A[\ell^e](\overline{\mathbb{F}}_q)$ denotes the ℓ^e-torsion group of A and $\boldsymbol{\mu}_{\ell^e}$ denotes the group of ℓ^e-th roots of unity. Should A furthermore be principally polarised, then the isomorphism $A \simeq A^\vee$ induced by the polarisation can be used to redefine the Weil pairing e_{ℓ^e} on pairs of elements

in $A[\ell^e](\overline{\mathbb{F}}_q)$. Moreover the pairing is compatible, i.e.

$$e_{mm'}(S,T)^m = e_{m'}([m]S,[m]T) \quad \text{for all } S,T \in A[mm']$$

for two positive integers m, m'. We can now explicitly state which criteria we need a subgroup of a PPAS to adhere to in order to define an isogeny which preserves the principal polarisation.

Definition 1.4 (Maximal ℓ^e-isotropic subgroup)**.** *Let A be an abelian variety and G be a proper subgroup of the torsion group $A[\ell^e]$. Then we call G a* maximal ℓ^e-isotropic subgroup *if*

(i) the ℓ^e-Weil pairing on $A[\ell^e]$ restricts trivially to G, and

(ii) G is a maximal subgroup with respect to Property (i).

An isogeny emanating from an abelian surface A whose kernel is given by a maximal ℓ^e-isotropic subgroup preserves any principal polarisation of A, so that any subgroup satisfying the above conditions for isotropy defines an isogeny between principally polarised abelian surfaces [Mum70]. From this definition, we can see that cyclic subgroups of A will not satisfy the maximality condition (ii). Hence, the isogenies which we are interested in do not have kernels of rank 1. We will later see that a polarisation preserving isogeny $\varphi : A \to A'$ can have a rank-2 or rank-3 kernel. The former isogeny is an (ℓ^e, ℓ^e)-isogeny with $\ker\varphi \cong (\mathbb{Z}/\ell^e\mathbb{Z})^2$, the latter is a $(\ell^e, \ell^{e-k}, \ell^k)$-isogeny for some $1 \leq k \leq e/2$ with $\ker\varphi \cong \mathbb{Z}/\ell^e\mathbb{Z} \times \mathbb{Z}/\ell^{e-k}\mathbb{Z} \times \mathbb{Z}/\ell^e\mathbb{Z}$.

Remark 1.5. *Isogenies between elliptic curves are often understood as a sequence or chain of isogenies, each having prime degree. Similarly, we understand an isogeny $\phi : A \to A'$ between PPAS corresponding to a kernel subgroup G which is ℓ^e-isotropic to be the composition or chain of e (ℓ,ℓ)-isogenies $\phi_1, \ldots, \phi_e$ between intermediate varieties. Each of these shorter isogenies corresponds to a subgroup of rank two generated by two linearly independent order-*

ℓ elements. Generally, chaining e (ℓ, ℓ)-isogenies together could result in an isogeny whose kernel subgroup is not maximal ℓ^e-isotropic, as this setting does not prevent us from chaining together some intermediary (ℓ, ℓ)-isogeny directly with its dual $\phi_{i+1} = \widehat{\phi}_i$, resulting in only composing with a multiplication map rather than extending the isogeny chain by two steps. In most cryptographic situations we only consider isogenies which are non-backtracking in order to prevent ambiguity arising from varying isogeny degrees or decreasing security when isogeny chains are shorter than expected. To formally characterise maximal ℓ^e-isotropic kernels of non-backtracking isogenies, we can add

(iii) $G \not\subset A[m]$ for any $m < \ell^e$

to the conditions of Definition 1.4. We will use this extended definition in the remainder of this thesis.

1.3.2 Computing isogenies

In general, computing isogenies in the two-dimensional setting is less efficient than computing isogenies between elliptic curves, and for quite small primes ℓ, computing (ℓ, ℓ)-isogenies is already infeasible. However, for certain isogenies and chains of isogenies there exist compact formulae and strategies to compute and evaluate isogenies.

Most prominently, a *Richelot isogeny* is a $(2, 2)$-isogeny from a Jacobian of a hyperelliptic genus-2 curve C to another with a maximal 2-isotropic subgroup of $A[2]$ as its kernel. Then the codomain of the isogeny, in particular the defining equation of the corresponding hyperelliptic curve, can be explicitly computed via quadratic splittings of the defining polynomial of C. A general strategy for computing (ℓ, ℓ)-isogenies for $\ell \neq 2, p$ is presented by Cosset and Robert in [CR15], though the arithmetic is often performed in large extension fields and therefore costly. Formulae for $(3, 3)$-isogenies have been provided by Bruin, Flynn and Testa in [BFT14], and $(5, 5)$-isogenies are covered in [Fly15]. Recently,

Kunzweiler [Kun22] presented a more efficient strategy for computing chains of Richelot isogenies, and this result has been adapted for $(3^e, 3^e)$-isogenies in [DK23].

CHAPTER 2

Isogeny-based cryptography

Elliptic curve cryptography is a well-studied and popular area of public key cryptography. Its schemes are currently regularly deployed in practice and rely on the hardness of pairing-based or discrete logarithm-based problems. Schemes based on computing discrete logarithms in the groups of elliptic curve points over finite fields, which is a foundational problem in elliptic curve cryptography, are especially vulnerable to future quantum attacks [Sho94]. Thus, there has been a considerable effort to use alternative problems originating from elliptic curves and higher-dimensional abelian varieties, as well as the maps between them, to design cryptographic protocols based on problems which are assumed to be hard both classically and quantumly. The field of isogeny-based cryptography is mainly based on the general problem of computing an isogeny between two given, arbitrary elliptic curves (or abelian surfaces), or variants thereof. For supersingular isogeny-based cryptography, the *pure isogeny problem* is as follows.

Problem 2.1 (Pure isogeny problem)**.** *Given two supersingular elliptic curves E and E' defined over the field $\mathbb{F}_{p^2}$ with p^2 elements, find an isogeny $E \to E'$.*

As the first known isogeny-based constructions were suggested in the mid to late 1990s and the most well-known scheme, the Supersingular Isogeny Diffie–Hellman key exchange [JD11], was not introduced until 2011, the field of isogeny-based cryptography is much less mature than elliptic curve cryptography, and also most other fields providing (assumed) quantum-safe primitives.

In 1997, a first key exchange utilising the properties of *ordinary* elliptic curves was constructed by Couveignes and only disclosed to some other researchers in private [Cou97]. More generically, the author introduced the concept of a *hard homogeneous space* (HHS), a framework describing a transitive action of a finite group on a set which allows efficient algorithms for some computational problems in the group while other computational tasks are sufficiently hard to solve. For example, one of the algorithmic requirements for the HHS group action is that it should be easy to evaluate the group action for any given pair of a set element and a group element while it should be hard to find a group element whose action takes a given set element to another. It is possible to construct HHS instances from isogenies between certain types of elliptic curves. Later, Rostovtsev and Stolbunov independently rediscovered and further developed Couveignes' initial proposal [RS06]. The so-called CRS scheme makes use of the group action which can be constructed from elements of the ideal class group acting on classes of isomorphic ordinary elliptic curves as follows.

If we denote the set of all isomorphism classes over $\overline{\mathbb{F}}_q$ of isogenous curves with n points and endomorphism ring $\mathcal{O}$ by $\mathrm{Ell}_{q,n}(\mathcal{O})$, we can represent the isomorphism class of a curve E in $\mathrm{Ell}_{q,n}(\mathcal{O})$ by its j-invariant $j(E)$. Any *horizontal* isogeny $\varphi : E \to E_{\mathfrak{b}}$ between curves in $\mathrm{Ell}_{q,n}(\mathcal{O})$ is determined by E and $\ker \varphi$ up to isomorphism, and the kernel corresponds to an ideal $[\mathfrak{b}]$ in $\mathcal{O}$. Since principal ideals in $\mathcal{O}$ correspond to endomorphisms, ideals that are equivalent in the ideal class group of $\mathcal{O}$, $\mathrm{Cl}(\mathcal{O})$, induce the same isogeny up to isomorphism [Wat69]. Hence, this setting describes a well-defined group action which further satisfies the criteria to instantiate HHS as it is free and transitive and fulfills the

relevant algorithmic requirements.

$$\cdot : \mathrm{Cl}(\mathcal{O}) \times \mathrm{Ell}_{q,n}(\mathcal{O}) \to \mathrm{Ell}_{q,n}(\mathcal{O}),$$
$$([\mathfrak{b}], j(E)) \mapsto j(E_{\mathfrak{b}}).$$

The isogeny computations to evaluate the group action in this setting often have to be performed over extension fields and hence this scheme is quite slow. Adapting the CRS scheme to supersingular elliptic curves, Castryck, Lange, Martindale, Panny and Renes managed to eliminate most of the performance issues allowing for larger practical parameters when introducing CSIDH [CLMPR18].[1] Both the CRS scheme and CSIDH have been shown to be vulnerable to quantum algorithms devised by Childs, Jao and Soukharev which run in subexponential time [CJS14] exploiting their commutative nature to reduce the underlying task of finding the corresponding ideal $[\mathfrak{b}]$ given two j-invariants $j(E)$ and $j(E_{\mathfrak{b}})$ to an instance of the hidden shift problem which can then be solved using one of Kuperberg's subexponential quantum algorithm for solving the hidden subgroup and hidden shift problems [Kup05; Kup13]. We describe this result in more detail in Section 3.4.

Meanwhile, the first construction using *supersingular* curves is the CGL hash function developed by Charles, Lauter and Goren [CLG09] over a decade later in 2009. Given a fixed starting curve, the hash function utilises the expander properties of supersingular isogeny graphs to deterministically construct an isogeny walk from its input. In order to hash a message, one converts the message into an ℓ-ary string. Then, the encoded message describes a random, non-backtracking walk in the ℓ-isogeny graph, providing the ending curve's j-invariant as the hash value of the input message. In particular, each value of the encoded message corresponds to one ℓ-isogeny emanating at first from the starting curve (where one ℓ-isogeny is canonically discarded), and later from the intermediate curves (where one of

[1] Note that the CSIDH construction is strictly speaking not an instance of a hard homogeneous space as certain algorithmic requirements are not satisfied for all elements. These issues are overcome through some precomputation and do not affect CSIDH in practice; see Section 3.1.2 for more details.

the $\ell + 1$ outgoing ℓ-isogenies would be backtracking). SIDH, the *Supersingular Isogeny Diffie–Hellman* key exchange [JD11], was the most prominent isogeny protocol working with supersingular elliptic curves. Though the scheme was fully broken in 2022 (see [CD23], [MM22; MMPPW23] or [Rob23]) its impact on isogeny-based cryptography remains fundamental and the results concerning its security are vital resources for further studies in the field.

Collectively, many isogeny-based schemes for varying primitives have been introduced and analysed, broken, discarded or fixed since the very first protocol by Couveignes. This includes signatures, e.g [DG19] and [DKLPW20], zero-knowledge proofs, e.g. [DDGZ23], and verifiable delay functions like [DMPS19] among many others. In this chapter we will provide an overview of relevant isogeny-based cryptographic schemes and related results which specifically utilise abelian varieties in dimensions one and two and are based on the SIDH protocol. We will conclude the chapter with a section on algorithmic results for computing isogenies between low-dimensional abelian varieties.

2.1 SIDH-based elliptic curve constructions

The Supersingular Isogeny Diffie–Hellman (SIDH) protocol introduced in 2011 by De Feo and Jao [JD11] forms the basis of the former NIST post-quantum candidate *Supersingular Isogeny Key Encapsulation* (SIKE) [JACCDHJKLLNRSU17]. Before recalling the SIDH protocol in detail, we describe how the authors utilised the fact that quotienting out an elliptic curve by two large only trivially intersecting subgroups is commutative to construct a Diffie–Hellman-type key agreement. We further discuss a selection of cryptanalytic results regarding SIDH and SIKE, as well as countermeasures for (some of) the attacks. Finally, we introduce some of the elliptic curve schemes which are based on the fundamental SIDH idea.

SIDH squares Recall that the original Diffie–Hellman protocol [DH76] is a key exchange utilising the hardness of computing discrete logarithms in a multiplicative group G generated by an element g of prime order p. Secret keys are then integers a and b modulo p chosen by each of the two parties at random, and the corresponding public keys are $y_A = g^a$ and $y_B = g^b$. As exponentiation is a commutative operation in the group G, by exchanging public keys, both Alice and Bob can compute a shared secret $(g^a)^b = g^{ab} = (g^b)^a$ by raising the other party's public key to the power of their own secret key.

The idea underlying SIDH is to recreate a similar construction with random walks in the supersingular isogeny graphs for two different small primes. In particular, one can choose a prime p such that one of $p+1$ and $p-1$ is divisible by large powers of ℓ_A and ℓ_B and then fix a public starting curve $E/\mathbb{F}_{p^2}$ for the two involved parties, say Alice and Bob. Now Alice considers the starting curve as a node in the ℓ_A-isogeny graph and takes a random, non-backtracking walk of length e_A to an end curve E_A. Meanwhile Bob considers E as part of the ℓ_B-isogeny graph and takes a random walk corresponding to a $\ell_B^{e_B}$-isogeny with codomain E_B. The parties can then exchange the j-invariants of the end curves of their walks, effectively swapping their location nodes in their respective graphs. To complete a Diffie–Hellman-type construction, Alice and Bob now each need to perform a walk "corresponding" to their own secret, but this time starting from the public curve the other party provided. This procedure should then allow both Alice and Bob to reach a shared node in their respective graphs as the codomain of their "shifted" isogeny.

However, the Diffie–Hellman construction does not transfer to isogeny graphs as straightforwardly as hoped since it was not known how to push forward a secret isogeny to an arbitrary starting curve without knowing an isogeny between these two curves. Jao and De Feo [JD11] suggest that the parties exchange some specific torsion point information, and this additional knowledge allows both parties to actually compute a pushforward of their secret isogeny. This process adds some additional computations to the public key generation

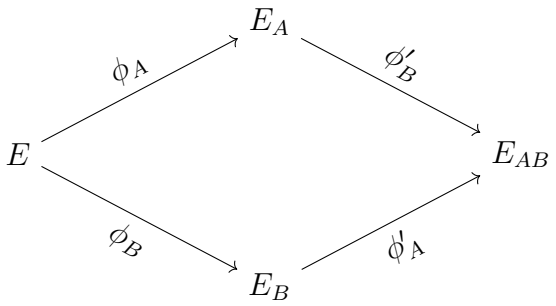

Figure 2.1: SIDH isogeny square.

but means that the parties can practically "commute" function composition in this setting, overcoming the technical difficulty of non-commutative endomorphism rings of supersingular curves. As can be seen in Fig. 2.1, Alice and Bob jointly compute two isogenies from E to E_{AB} during the key exchange, one being the composition of ϕ'_B with ϕ_A via E_A and the other being $\phi'_A \circ \phi_B$ via E_B. The curves and isogenies involved in one specific SIDH instance as in the diagram are often called an *SIDH square* with the isogenies known to one party, i.e. ϕ_X and ϕ'_X for either party $X \in \{A, B\}$, *parallel* as their domain and codomain curves are isogenous via same-degree isogenies respectively.

The main result underlying the construction of this key exchange is that the final elliptic curves computed by Alice and Bob belong to the same isomorphism class of supersingular elliptic curves, allowing both parties to agree on a single j-invariant as a shared secret. In particular, Alice and Bob compute

$$\phi'_A(\phi_B(E)) \cong E_{AB} \cong \phi'_B(\phi_A(E))$$

together. If P_A, Q_A are independent generators of $E[\ell_A^{e_A}]$ such that Alice's secret walk corresponds to an isogeny with kernel generated by $P_A + [\alpha]Q_A$ and she is provided during the exchange of public keys with the image of the $\ell_B^{e_B}$-torsion generators P_B, Q_B under Bob's

secret isogeny ϕ_B, then Alice can compute E_{AB} and its j-invariant from

$$E_{AB} = E_B/\langle \phi_B(P_A) + [\alpha]\phi_B(Q_A)\rangle.$$

Bob can then proceed mutatis mutandis and obtain $j(E_{AB})$.

2.1.1 The SIDH protocol

In the setup phase of the original protocol as in [JD11], a supersingular elliptic curve E defined over a field $\mathbb{F}_{p^2}$ is fixed together with a prime p. It is important that p is of the form $p = \ell_A^{e_A}\ell_B^{e_B}f - 1$, where ℓ_A and ℓ_B are two small primes and f is a small cofactor coprime to both ℓ_A and ℓ_B. In most applications, in order to ensure similar security for both parties, e_A and e_B are large integers chosen in such a way that $\ell_A^{e_A} \approx \ell_B^{e_B}$. Fix some points P_A and Q_A such that they jointly generate the $\ell_A^{e_A}$-torsion, $E[\ell_A^{e_A}]$, and similarly, $\langle P_B, Q_B\rangle = E[\ell_B^{e_B}]$. Then the protocol is as follows. An overview is presented in Fig. 2.2.

1. **Key Generation**

   - Alice selects a random cyclic subgroup G_A of $E[\ell_A^{e_A}]$ of order $\ell_A^{e_A}$. Due to the definition of a basis, she can find integers x_A, y_A not both divisible by ℓ_A such that $A = [x_A]P_A + [y_A]Q_A$ is a generator of G_A. Alice then computes the isogeny $\phi_A : E \to E/G_A$ with codomain $E_A := E/G_A$.

   - Alice's secret key is the pair (x_A, y_A) corresponding to G_A and her public key is the triple $(j(E_A), \phi_A(P_B), \phi_A(Q_B))$.

   - Analogously, Bob selects a random cyclic order-$\ell_B^{e_B}$ subgroup G_B of $E[\ell_B^{e_B}]$ generated by a point $B = [x_B]P_B + [y_B]Q_B$ defined via some positive integers x_B, y_B not both divisible by ℓ_B. Bob computes his secret isogeny $\phi_B : E \to E/G_B$ with codomain E_B.

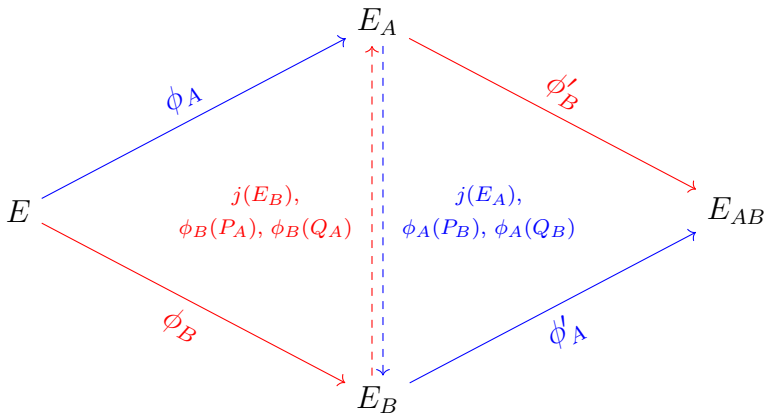

Figure 2.2: The Supersingular Isogeny Diffie–Hellman (SIDH) protocol.

- Bob's secret key is the pair (x_B, y_B) defining G_B and his public key is the triple $(j(E_B), \phi_B(P_A), \phi_B(Q_A))$.

2. **Key Exchange**

- Alice and Bob exchange their public information.

- Alice and Bob can now both compute the shared secret, the j-invariant of the curve $E_{AB} := E/(G_A, G_B)$. This is possible utilising the combination of their secret key and the other party's public key, i.e. the images of the torsion points (e.g. Alice computes $\phi_B(G_A)$ as $\langle [x_A]\phi_B(P_A) + [y_A]\phi_B(Q_A)\rangle$ and can then obtain the desired curve and its j-invariant from $E/\langle A, B\rangle = E_B/\phi_B(G_A)$).

In [JD11], Jao and De Feo propose that in practice, a starting curve E with j-invariant 1728 and primes $\ell_A = 2$ and $\ell_B = 3$ should be used as these parameters offer an efficient run of the protocol.

Keyspace Let $\ell \in \{\ell_A, \ell_B\}$ and n be the corresponding exponent in $\{e_A, e_B\}$. Furthermore, let P and Q be the generators of the ℓ^n-torsion. As described in the first step of the SIDH protocol, choosing a secret SIDH key for the party using the ℓ^n-torsion amounts to selecting a

random cyclic order-ℓ^n subgroup of $E[\ell^n]$, i.e. uniformly sampling an element of the keyspace $\mathcal{K}_\ell$. Then it is possible to find two integers x and y modulo $\mathbb{Z}/\ell^n\mathbb{Z}$ such that $xP + yQ$ is a generator of the selected secret subgroup. In particular, by [GPST16, Lemma 2.1], it is possible to normalise secret keys in a large subset of $\mathcal{K}_\ell$ in such a way that either party can always choose the secret integers x, y so that one of them equals 1 given that the torsion generators P, Q must be independent by definition. Therefore it is possible to choose only a single secret integer instead of two, resulting in the partition of the keyspace into two disjoint subsets as follows.

$$\mathcal{K}_\ell = \{\langle P + [x]Q\rangle \mid x \in \mathbb{Z}/\ell^n\mathbb{Z}\} \cup \{\langle [\ell x]P + Q\rangle \mid x \in \mathbb{Z}/\ell^n\mathbb{Z}\} \tag{2.1}$$

However in practice, one usually restricts the secret generation to the first type of group since this significantly simplifies the uniform secret generation procedure while discarding only a relatively small subset of possible keys. In [JACCDHHJKKLLNPRSU22, Section 1.3.9], it is shown that the keyspace with this restriction in place has cardinality ℓ^n in comparison to the cardinality $\ell^{n-1}(\ell+1)$ of $\mathcal{K}_\ell$ in total generality. In particular, this implies that we can assume Alice's and Bob's secret subgroups G_A and G_B to be generated by points $A = P_A + [\alpha]Q_A$ and $B = P_B + [\beta]Q_B$ respectively, and that we can consider their respective secrets to be $\alpha \in \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ and $\beta \in \mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$.

Remark 2.2. *Throughout this thesis, we will use several different ways to describe secret SIDH subgroups such as via corresponding isogenies and their representations as rational maps or integer scalars defining the kernel generators in terms of a fixed torsion basis. This also applies for secrets in variants of the protocol introduced in the remainder of this chapter.*

2.1.2 Cryptanalysis of SIDH

The security of SIDH is not simply based on the hardness of Problem 2.1 as the protocol reveals additional information which can be used by an attacker to (partially) compute a party's secret key. The additional knowledge which can be gathered initially is the degree of the specific isogeny which is to be recovered. By agreeing on a prime p parametrising the finite field to work in and specifying each party's torsion basis, it is revealed that the secret isogenies in a particular SIDH instance will have degrees $\ell_A^{e_A}$ and $\ell_B^{e_B}$ respectively. Furthermore, once public information has been exchanged, an attacker can also observe the action of the secret isogeny on the other party's torsion subgroup. Hence, an adversary is equipped with additional information, and the security assumption underlying the SIDH protocol is that the *supersingular computational Diffie–Hellman problem* [JD11, Problem 5.3] is computationally infeasible. With the setup and parameters as described in Section 2.1.1, the SSCDH problem is the following.

Problem 2.3 (SSCDH problem)**.** *Let $\phi_A : E \to E_A$ be an isogeny with kernel $\langle P_A + [\alpha]Q_A \rangle$ for some random $\alpha \in \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ (and ϕ_B defined analogously). Then, given E_A, E_B as well as the points $\phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A)$, the* supersingular computational Diffie–Hellman problem *is to find the j-invariant of the curve $E/\langle P_A + [\alpha]Q_A, P_B + [\beta]Q_B \rangle$.*

Suppose we want to recover Alice's secret key α. As discussed previously, the (reduced) keyspace for SIDH in practice has size $\ell_A^{e_A} \approx \sqrt{p}$. This is also the number of different isogenies of degree roughly $\sqrt{p}$ which need to be computed and whose codomain curves need to be compared to the given curve. Should the two curves be isomorphic, one can check their action on the relevant torsion group. Alternatively, one requires $O(p^{1/4})$ storage and time each to perform a meet-in-the-middle attack on SIDH: Firstly, all isogenies of degree $\ell_A^{e_A/2}$ emanating from E are computed. Secondly, one computes all isogenies of degree $\ell_A^{e_A/2}$ emanating from the target curve E_A. Then one finds the elliptic curve in the intersection

of codomains of both sets of isogenies, i.e. an elliptic curve E_m such that there exist isogenies $\phi_1 : E \to E_m$ and $\phi_2 : E_A \to E_m$, both of degree $\ell_A^{e_A/2}$. In particular, this implies that one can find the desired isogeny by concatenating the dual of ϕ_2 with ϕ_1 to obtain $\phi_A = \hat{\phi}_2 \circ \phi_1 : E \to E_A$. Note that primarily this solves the problem of finding an isogeny of given degree between two elliptic curves without taking the additional torsion point information provided in Problem 2.3 into account. Due to the relatively small degree of the desired isogeny, it is however reasonable to assume that a unique isomorphism class of curves, that of E_m, will lie in the intersection of the codomain curve sets: While the total number of isomorphism classes isogenous to E is roughly $p/12$ [Sil09], the isogeny between E and E_A is only of degree roughly $\sqrt{p}$, implying that there are approximately $p^{1/4}$ candidates for ϕ_1 and ϕ_2 each. Hence, the curve E_m found "in the middle" should also lead to an isogeny matching the torsion point information. The computational and particularly storage requirements of these two naive attacks are infeasible, therefore, to break SIDH, more inventive techniques than brute force, meet-in-the-middle or best-known path-finding attacks are required. We present some of the most impactful cryptanalytic results below.

The GPST attack

One branch of cryptanalysis of the SIDH protocol focuses on exploiting the auxiliary torsion point information provided by the honest participant(s) of the scheme to reconstruct one of the secret isogenies. The active attack [2] on standard SIDH presented by Galbraith, Petit, Shani and Ti (GPST) [GPST16] utilises the torsion information provided to extract Alice's secret key α bit by bit through multiple executions of the protocol where Bob sends changing, malformed torsion point images. This strategy is only effective when the same static secret key is used for multiple protocol instances and makes clear that SIDH should not be used with static keys for non-interactive key exchange.

[2] An active or adaptive attack is a standard attack framework on systems where one party (usually Alice for convenience) uses a static key (i.e. reuses their private key) and can hence be considered as an oracle for e.g. signing or encrypting, depending on the type of cryptosystem, with a fixed secret value.

Fix the primes Alice and Bob use as $\ell_A = 2$ and $\ell_B = 3$ in the setup of SIDH as presented in Section 2.1.1. Suppose further that Alice uses a static secret key α which an adversary wishes to recover. The attacker will hence take the role of Bob in the key exchange protocol. The attack oracle with which the GPST attack can be modeled provides one bit of information per interaction. On input of Bob's public information (E_1, R, S) together with a candidate curve E_2 for the shared secret curve, the oracle will confirm whether an execution of the protocol with the provided public key will produce a curve with the same j-invariant as E' or not. More precisely, $\mathcal{O}$ is defined as follows.

$$\mathcal{O}(E_1, (R, S), E_2) = \begin{cases} 1 & \text{if } \; E_2 \simeq E_1/\langle R + [\alpha]S \rangle, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

In an honest run of the protocol, the adversary first computes Bob's public key $(E_B, R := \phi_B(P_A), S := \phi_B(Q_A))$ and finds the elliptic curve E_{AB}.[3] Then Bob sends several varying queries to the oracle where the curves $E_1 = E_B$ and $E_2 = E_{AB}$ are fixed as in an honest protocol execution, while the torsion points are maliciously changed in each step to extract information. The adversary wants the malformed information to remain undetected and thus must ensure that throughout the attack, all queries are valid. In particular, the attacker must create the malformed basis elements (R', S') in such a way that their Weil pairing value is the same as $e(R, S)$ where R and S are the points in the honest first interaction as defined above.

If we write α as $\sum_{i=0}^{e_A - 1} 2^i \alpha_i$ with the α_i bits, the attacker determines the parity of the secret key α in a first step by sending malformed points $(R', S') = (R, S + [2^{e_A - 1}]R)$. By the lemma below, $\mathcal{O}(E_B, (R', S'), E_{AB}) = 0$ then implies that α is odd, while $\alpha_0 = 0$ otherwise.

Lemma 2.4 ([GPST16, Lemma 2])**.** *Let $R, S \in E[2^{e_A}]$ be linearly independent points of*

[3] Note that by construction $\mathcal{O}(E_B, (R, S), E_{AB}) = 1$.

order 2^{e_A} *and let* $\alpha \in \mathbb{Z}$. *Then*

$$\langle R + [\alpha](S + [2^{e_A-1}]R)\rangle = \langle R + [\alpha]S\rangle$$

if and only if α *is even.*

This strategy of extracting the next more significant bit of α can then be iterated for the bits $i = 1, \ldots, e_A - 3$, while the last two bits must be brute forced to avoid detection. Let K_i denote the part of the key which has been recovered in the first i steps of the iteration. For the $i+1$-th step, the attacker sends public information of the form

$$(E_B, R' = \theta(R - [2^{e_A-i-1}K_i]S), S' = \theta([1 + 2^{e_A-i-1}]S)) \tag{2.3}$$

where θ is some scaling factor ensuring that the condition on the Weil pairing, $e_{2^{e_A}}(R, S) = e_{2^{e_A}}(R', S')$, is met. The points are specifically chosen such that they are of the correct order, satisfy the Weil pairing and reveal one bit of α per oracle query since

$$O(E_B, (R', S'), E_{AB}) = \begin{cases} 1 & \text{if } a_i \equiv 0 \pmod 2, \\ 0 & \text{if } a_i \equiv 1 \pmod 2. \end{cases}$$

Computing the kernel subgroup which Alice would utilise to define ϕ'_A when provided with Bob's malformed information gives

$$\langle R' + [\alpha]S'\rangle = \begin{cases} \langle R + [\alpha]S\rangle & \text{if } a_i = 0, \\ \langle R + [\alpha]S + [2^{e_A-1}]S\rangle & \text{if } a_i = 1. \end{cases}$$

By [GPST16, Lemma 3], we know that the two subgroups are distinct, so that a case distinction can be made from the oracle's response. This way K_{e_A-3} can be computed, and the authors suggest the brute force method for the two remaining bits. In total, the full recovery of α takes less than e_A queries to the oracle along with some computational efforts for the

last bits.

Remark 2.5. *We give some more insight via an alternative presentation of the GPST attack in Section 5.4.2. This allows us to devise a setting in which a GPST-inspired adaptive attack can be performed in a higher-dimensional setting; see Chapter 5.*

Countermeasures With the novel SIDH attacks described in Section 2.1.2, public key validation in SIDH is possible by simply running the attack. Until then, it used to be unknown how a party using a static key, say Alice, could reliably confirm that a received public key was honestly generated by Bob. This confirmation vonsisted of ensuring that indeed $R = \phi_B(P_A)$ and $S = \phi_B(Q_A)$ where (E_B, R, S) is Bob's public information. There were multiple small checks Alice could perform, none of which were powerful enough to detect the malformed points an attacker might send as part of a GPST-type attack. She needed to confirm the following.

- $R, S \in E_B$.
- The order of R and S is $\ell_A^{e_A}$.
- R and S are independent and hence generate the $\ell_A^{e_A}$-torsion on E_B.
- The points have the correct Weil pairing value (i.e. $e_{\ell_A^{e_A}}(R, S) = e_{\ell_A^{e_A}}(P_A, Q_A)^{\ell_B^{e_B}}$)

Remark 2.6. *The last check exploits the relationship of the Weil pairing of two points and their images under an isogeny with the isogeny degree as described in Equation (1.1). The Weil pairing can also be used to check for independence of the points, rendering the previous check redundant.*

The only effective way to prevent adaptive attacks on static keys was to choose new ephemeral keys for each instance of the SIDH protocol. In [JACCDHJKLLNRSU17], a version of the Fujisaki–Okamoto transform due to Hofheinz, Hövelmanns and Kiltz [HHK17]

is used to design an IND-CCA secure key encapsulation mechanism based on SIDH which is called SIKE (Supersingular Isogeny Key Encapsulation). SIKE does not support the use of static keys, hence some further adaptations of the original SIDH protocol try to rectify this issue while analogously hindering the above attacks. Two of these non-interactive key exchange protocols are presented in Section 2.1.4.

Petit's torsion point attack

Another class of attacks on the (assumed) hard problem underlying SIDH is to compute endomorphism rings of the public curves involved from the provided torsion information. Knowledge of the structure of a curve's endomorphism ring, or even partial information on it, can make solving the explicit problem at the core of a protocol execution much simpler. The reason for wanting to compute (low-degree) endomorphisms of supersingular elliptic curves is that they allow an attacker to make deductions about the isogeny graph. This is due to the Deuring correspondence (Section 1.2.1) which allows an attacker to translate the task of computing an isogeny to an equivalent task of finding a connecting ideal between maximal orders. An algorithm to compute a connecting ideal of prime power norm in polynomial time has been developed by Kohel, Lauter, Petit and Tignol (KLPT) [KLPT14]. However, the isogenies determined by the resulting ideals do not satisfy the degree requirements specified by the SIDH protocol as they are usually too large and the algorithm can therefore not solve the isogeny problem underlying SIDH directly. Thus, this group of attacks on SIDH consist roughly of first computing an endomorphism θ of desirable degree on the starting curve E. One can then shift θ to the curve E' along the secret isogeny $\varphi : E \to E'$ by computing a so-called shifted "lollipop" endomorphism of E': One composes θ with the secret isogeny and its dual and shifts it by some integer to obtain $\tau = \hat{\varphi} \circ \theta \circ \varphi + [d]$. Then due to clever choices of θ and the unbalancedness of the parameters in the scheme, $\ker \varphi$ can be extracted.

The first passive attack realising this idea was introduced by Petit [Pet17] and it suc-

ceeds when SIDH is performed with non-standard parameters. In contrast to the suggestion of [JD11] to use balanced parameters where $\ell_A^{e_A} \approx \ell_B^{e_B} \approx \sqrt{p}$, the Petit attack requires unbalanced torsion sizes, i.e. the (prime power) order of one torsion must be sufficiently larger than the other. In [QKLMPPS21], the required size and unbalancedness of the parameters is reduced in general and it is highlighted that a torsion point attack is feasible on balanced parameter sets for very specific choices of primes. Further improvements to the performance of these torsion point attacks have been made in [QKLMPPS21] and [FKMT22].

Full key recovery

Recently, several results have emerged relying mainly on a theorem by Kani [Kan97] about gluing and splitting isogenies of abelian surfaces. Informally, this theorem provides a criterion to determine whether an (N, N)-isogeny $\varphi : E_1 \times E_2 \to A$ between abelian surfaces with the domain being an elliptic product also has a product of elliptic curves as a codomain. Due to the small proportion of elliptic curve products among all principally polarised abelian surfaces, it is highly unlikely that a random walk in the isogeny graph ends with an isogeny to a split surface.[4] To make the deduction of whether A is an elliptic product via Kani's theorem, one needs to examine the kernel of the isogeny, $\ker \varphi =: H$, and determine if it corresponds to an order-N isogeny diamond configuration.

Definition 2.7 (Isogeny diamond configuration of order N)**.** *Let N be a positive integer and $\phi : C \to E$ a (separable) isogeny between elliptic curves. Now consider two only trivially intersecting subgroups $G_1, G_2 \subset \ker \phi$ such that $|G_1| \cdot |G_2| = \deg \phi$ and $|G_1| + |G_2| = N$. Then the triple (ϕ, G_1, G_2) is an* isogeny diamond configuration of order N*.*

More precisely, checking whether H defines an isogeny between elliptic products means

[4] More precisely, Castryck, Decru and Smith show the following in [CDS20, Section 4]: When considering principally polarised abelian surfaces defined over $\mathbb{F}_{p^2}$ for large p, there are approximately $p^2/288$ elliptic curve products in comparison to the roughly $p^3/2880$ Jacobians of hyperelliptic genus-2 curves among all such surfaces. Hence, the proportion becomes negligible when cryptographically-sized primes are considered.

testing whether H is a group of the form

$$H = \langle (P, x\phi(P)), (Q, x\phi(Q)) \rangle$$

for P, Q a basis of an E_1-torsion coprime to N, $\phi : E_1 \to E_2$, and some invertible $x \in \mathbb{Z}$, forming an N-isogeny diamond configuration with some subgroups G_1, G_2 of $\ker \phi$. If so, A is a split surface obtained from a product of two elliptic curves.

It is possible to manipulate the setting of a traditional SIDH instance by constructing auxiliary curves and isogenies related to the curves known to an attacker in such a way that one can iteratively guess isogenies of increasing lengths and test whether they yield an isogeny between elliptic curve products as described above.

This strategy was first introduced in [CD23], where the authors manage to recover Alice's secret keys for all security level parameter sets provided for SIKE with very small computational and time resources. Concretely, Castryck and Decru show that if SIDH and related schemes are instantiated with a starting elliptic curve E such that its endomorphism ring is known, as is the case for SIKE for example, an attacker can recover a secret SIDH isogeny in polynomial time under certain heuristics. A similar attack has independently been described by Maino and Martindale [MM22] and others [MMPPW23] where a more direct strategy for starting curves with known endomorphism ring is proposed to improve performance and the attack is generalised to curves for which the endomorphism ring is unknown or cannot be efficiently represented. Robert [Rob23; Rob22] shortly thereafter managed to generalise the strategy further: Firstly, the author shows that key recovery is possible in deterministic polynomial time regardless of which supersingular elliptic curve is chosen as the starting curve E in the SIDH parameters. In particular, it is no longer necessary to know $\operatorname{End} E$ to perform the attack. This result stems from Robert's concept of a more general framework embedding varieties and isogenies in $2g$ dimensions, e.g. abelian

varieties of the form $E^g \times E_B^g$ and isogenies emanating from them, instead of the 2-dimensional setting explored in the previous attacks. In particular, using varieties in 4 or 8 dimensions allows breaking SIDH for parameters which were infeasible due to constraints on computing isogenies of non-smooth degrees. Secondly, Robert shows that polynomial-time attacks can (theoretically) be performed on higher-dimensional variants of elliptic curve SIDH such as G2SIDH which instantiates the protocol with two-dimensional abelian varieties instead of elliptic curves; see Section 2.2.1.

2.1.3 Fixing SIDH

As shown in the previous section, the SIDH protocol is vulnerable to both active and passive attacks. In particular, since the first result introducing the family of Castryck–Decru attacks was published in 2022, SIDH and SIKE are fully broken and no longer recommended for secure key exchange or encapsulation. In order to preserve SIDH-based cryptographic protocols, some candidate variants of SIDH have been proposed to prevent revealing as much torsion point information as is needed for the efficient attacks of Section 2.1.2. We briefly present two approaches for repairing SIDH below.

The first idea is due to Fouotsa, Moriya and Petit [Fou22; Mor22; FMP23] and consists of obscuring the information contained in a public key in order to prevent efficient attacks utilising Kani's theorem. In particular, the authors introduce the M-SIDH and MD-SIDH protocols which are based on scaling and hence masking the torsion point images provided and computing variable-degree isogenies and hence masking isogeny degrees respectively.

Masking torsion points in both parties' public keys reduces to scaling the usual points computed in SIDH by a random secret integer. To prevent the recovery of this secret scaler by an attacker, the authors require the (fixed) isogeny degrees to have a large number of distinct prime factors, leading M-SIDH to work with isogeny degrees which are not prime-

powers as in SIDH originally. For MD-SIDH, instead of always using isogenies of degree $\ell_A^{e_A}$ or $\ell_B^{e_B}$, one selects isogeny degrees uniformly at random from divisors of a fixed maximal composite degree. This means that, again, the degrees of the involved isogenies are no longer prime powers but products of several prime powers. Additionally, torsion points are scaled to obscure the isogeny degrees further. It follows that an adversary cannot launch Castryck–Decry-type attacks unless they have recovered the isogeny degree first. Note however, that computing the Weil pairing of the two points in a public key still reveals some information about the degree used, implying that the number of distinct primes dividing the isogeny degrees and parameters in general need to be large to ensure security. Hence, M-SIDH and MD-SIDH are significantly less efficient than SIDH.

Another recent idea of repairing the SIDH construction is to consider artificial orientations of supersingular elliptic curves and associated isogenies for the protocol. In [BF23], the authors present the binSIDH and terSIDH protocols which utilise such artificial orientations and work with fixed-degree and variable-degree isogenies respectively. In comparison to the other fixes above, the implementation results of [BF23], especially for the latter protocol, seem to indicate the possibility of terSIDH being a more efficient key exchange.

Let N be a positive integer. Then Basso and Fouotsa define an artificial N-orientation on a supersingular curve E to be a pair of cyclic subgroups of order N in $E[N]$ which only intersect trivially. For an artificial N-orientation (G, H), isogenies associated to the orientation can be defined as isogenies ϕ such that $\ker\phi$ is a cyclic subgroup of $G \oplus H$. This implies that the isogeny can be decomposed into isogenies of coprime degrees as $\phi = \phi_H \circ \phi_G$ with $\ker\phi_G \subset G$ and $\ker\phi_H \subset H$. Such artificial orientations reveal less information than "real" orientations (as for example defined in [CK20]) but suffice for constructing parallel isogenies as they are required to complete an SIDH square: Suppose SIDH is instantiated with supersingular elliptic curves equipped with artificial orientations for Bob and Alice, and both parties use artificially oriented isogenies. While a party in the original SIDH

protocol, say Alice, requires specific torsion point images under Bob's isogeny ϕ_B to push forward her isogeny ϕ_A to obtain the isogeny ϕ'_A emanating from E_B for completing the shared secret computation, if isogenies associated to artificial orientations on E are used, only the oriented isogenies need to be pushed forward. This means only the images on two cyclic torsion subgroups need to be communicated. It is expected that this information is not sufficient to launch attacks like those discussed in Section 2.1.2.

Overall, the new attacks on SIDH provide a new understanding of certain isogeny problems with torsion information and thus have a significant cryptanalytic impact on isogeny-based schemes. As described above, there are already some promising ways to repair the SIDH construction, but the discovery of efficient ways to compute isogenies between supersingular curves only given their degree and enough torsion information creates another opportunity for isogeny-based cryptography: These new techniques could be used constructively to build new isogeny-based encryption or other schemes as has been done for example from Petit's torsion attacks [Pet17].

2.1.4 Protocol variants

Many isogeny-based schemes have been constructed which are utilising the commutativity achieved in the SIDH isogeny square by providing additional torsion point information. This has included schemes covering several cryptographic primitives, and specifically ones proposed with the aim of preventing adaptive attacks even in the static-static setting of a non-interactive key exchange (NIKE). Although these variants are also broken by the full key recovery attacks on SIDH, they can a priori be fixed with similar techniques. We will present two such key exchange protocols which are closely related to SIDH, k-SIDH and a variant due to Urbanik and Jao.

k-SIDH

As a reaction to the active GPST attacks on the SIDH protocol, Azarderakhsh, Jao and Leonardi [AJL18] developed a generic transformation which allows one to construct a static-static key agreement from a key exchange scheme which had previously shown vulnerabilities to adaptive attacks on static keys. In particular, this transformation can be applied to SIDH to obtain the so-called k-SIDH protocol. This modification was believed to be secure against active attacks, and its passive security relies on the passive security of the underlying key agreement, SIDH. However, in the case of SIDH and k-SIDH, active security then comes at the price of significant loss of efficiency, since the main idea is combining secret information from multiple protocol instances into a shared secret in such a way that attacking individual instances via the known active attacks is computationally infeasible.

With the set-up being as in the original SIDH protocol, both parties first agree on a supersingular elliptic curve E defined over a field $\mathbb{F}_{p^2}$ where the prime p is chosen to be of the form $p = 2^{e_A}3^{e_B}f \pm 1$, together with bases for the 2^{e_A}-torsion and the 3^{e_B}-torsion of E. Let these bases be denoted by $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$ respectively. For some positive integer k, Alice and Bob then both choose k random secret integers which each generate a secret kernel subgroup and corresponding isogeny. Each party thus obtains a tuple of k separate SIDH public keys as their k-SIDH public information. For each of the k^2 possible combinations of their secret integers, Alice and Bob now perform an SIDH-type key agreement. Let H be a hash function exhibiting preimage-resistance. Then the ley exchange is executed as follows.

1. **Key Generation**

   - Alice selects k secret integers modulo 2^{e_A}, $\alpha^{(1)}, \ldots, \alpha^{(k)}$, and for each secret value computes the corresponding isogeny $\phi_{A,r}$ with codomain curve $E_A^{(r)} = E/\langle P_A + [\alpha^{(r)}]Q_A\rangle$.

- Alice's secret key is then $(\alpha^{(1)}, \ldots, \alpha^{(k)})$ while her public key is

$$(E_A^{(1)}, \phi_{A,1}(P_B), \phi_{A,1}(Q_B)), \ldots, (E_A^{(k)}, \phi_{A,k}(P_B), \phi_{A,k}(Q_B)).$$

- Bob similarly selects k secret integers modulo 3^{e_B}, $\beta^{(1)}, \ldots, \beta^{(k)}$, and computes the the corresponding isogenies $\phi_{B,r}$ along with their codomain curves $E_B^{(r)} = E/\langle P_B + [\beta^{(r)}]Q_B \rangle$.

- Bob's secret key is $(\beta^{(1)}, \ldots, \beta^{(k)})$ and his public key is

$$(E_B^{(1)}, \phi_{B,1}(P_A), \phi_{B,1}(Q_A)), \ldots, (E_B^{(k)}, \phi_{B,k}(P_A), \phi_{B,k}(Q_A)).$$

2. **Key Exchange**

- Alice sends her public key to Bob, and Bob responds with his own public key.

- For every two public curves $E_A^{(r)}, E_B^{(s)}$ for $1 \leq r, s \leq k$, Alice and Bob each compute the j-invariant $j_{r,s}$ of the elliptic curve which would be the resulting shared secret of a regular SIDH key exchange instance involving the pair of curves.

- The hash $h = H(j_{1,1}||j_{1,2}||\ldots||j_{k,k})$ of the concatenation of all the shared j-invariants is then used as the shared secret between Alice and Bob.

For verification of the public information received from another party, it is checked whether the torsion images supplied are independent and of the correct order, and whether their Weil pairing is of the correct value. Furthermore, they exchange hashes of their shared secret which allows both parties to verify if they agree on the final hash value h.

The Jao–Urbanik NIKE

Jao and Urbanik [UJ20] proposed that the non-trivial automorphisms which exist on specific classes of supersingular elliptic curves could be used to design another version of a non-interactive key exchange based on SIDH in the hopes of reducing the computational cost associated to k-SIDH. The inefficiency of k-SIDH stems in part from the fact that key sizes are increased by a factor of k and computation times are increased by a factor of k^2 in comparison to SIDH. Hence, Jao–Urbanik proposed using special starting curves in the protocol which would allow both parties to construct one or two related SIDH instances for every pair of secret keys. This is possible due to the existence of non-trivial automorphisms which facilitate the translation of a single secret kernel to one or more isomorphic curves, effectively thus defining distinct isogenies which can lead to distinct shared j-invariants.

Recall that an automorphism of an elliptic curve E is an invertible map $E \to E$ which is further a group homomorphism. In the general case, an automorphism η of a curve $E/\mathbb{F}_{p^2}$ can either be the identity map $\eta(P) = P$, or the negation map $\eta(P) = -P$ for all $P \in E$. However, when considering an SIDH-type prime p satisfying $p \equiv 2 \pmod 3$ and $p \equiv 3 \pmod 4$, there do exist two isomorphism classes of supersingular curves which have additional automorphisms distinct from the identity and negation maps which are always present: the classes characterised by j-invariants $j = 0$ and $j = 1728$. For curves with j-invariant $j(E) = 0$, there exists an order-6 automorphism η_6 of defined by

$$\begin{aligned} \eta_6 : E &\to E \\ (x, y) &\mapsto (\zeta_3 x, -y) \end{aligned}$$

where ζ_3 denotes a primitive third root of unity. The automorphism satisfies the equations $\eta_6^3 = -1$ and $\eta_6^2 = \eta_6 - 1$. For curves with j-invariant $j(E) = 1728$, we have an automorphism η_4 of order four explicitly given by $\eta_4(x, y) = (-x, iy)$. To maximise efficiency, using the

former as starting curves is advised by the authors. Thus, we centre our discussion in the following around curves with j-invariant 0 and automorphism η_6.

Let E be such a curve of j-invariant $J(E) = 0$. Hence, E is equal or isomorphic to $E_0 : y^2 = x^3 + 1$. Since there exists η_6 as an order-6 automorphism on E, we can observe the following behaviour which will aid in the construction of the Jao–Urbanik protocol.

Lemma 2.8. *Let $G \subseteq E$ be a cyclic subgroup of order 2^{e_A} of a supersingular elliptic curve $E/\mathbb{F}_{p^2}$ with $j(E) = 0$ as before. Then $\eta_6(G)$ and $\eta_6^2(G)$ are also subgroups of E, and the three subgroups are all distinct.*

Proof. It is obvious that the image of a subgroup of E under a group homomorphism will be a subgroup. To prove that the subgroups G and $\eta_6(G)$ are distinct, assume the opposite. If $\eta_6(G) = G$ for G cyclic, then there exists some odd k for which $G \subseteq \ker(\eta_6 + k)$. Since $\deg(\eta_6) = \mathrm{tr}(\eta_6) = 1$ is implied by $\eta_6^2 - \eta_6 + 1 = 0$, we then have

$$\deg(\eta_6 + k) = (\eta_6 + k)(\bar{\eta_6} + k) = \deg(\eta_6) + k\,\mathrm{tr}(\eta_6) + k^2 = 1 + k + k^2.$$

As k is odd, $\deg(\eta_6 + k)$ is also odd. This is impossible as it hence is not divisible by 2^{e_A} as implied by $G \subseteq \ker(\eta_6 + k)$. Analogous arguments show that $\eta_6(G)$ and $\eta_6^2(G)$, as well as $\eta_6^2(G)$ and $G = -G$ are pairwise distinct. □

Lemma 2.8 is an extension and formalisation of the statement of [BKMPW20, Footnote 2]. Note that an analogous result can be stated for subgroups of a different prime power order, therefore we can deduce that the isogenies emanating from E which correspond to the three different kernels G as well as the images of G under different powers of the automorphism η_6 respectively are all distinct, the associated codomain curves are isomorphic however. As an example, observe that an isogeny $\varphi : E \to E/G$ with kernel G while the map $\varphi \circ \eta_6^{-1} : E \to E/G$ is obtained by quotienting out E with $\eta_6(G)$. We can thus deduce

that $E/G \cong E/\eta_6(G)$.

Suppose Alice has generated a secret key for SIDH with the starting curve E and sends her public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$ to Bob. We can then extract two more distinct but related SIDH public keys from this single public key with the help of η_6. All three public codomain curves $E/\langle A\rangle \cong E/\langle\eta_6(A)\rangle \cong E/\langle\eta_6^2(A)\rangle$ are isomorphic, and therefore collectively have one j-invariant, while each of the isogenies associated to the public keys are not. An analogous argument can be made for Bob's public information. We now formally prove [ACLLNSS23, Lemma 6].

Lemma 2.9. *Suppose a base curve E with $j(E) = 0$ together with the parameters as suggested by Jao and Urbanik [UJ20] is used for SIDH. Then a single exchange of Alice's and Bob's SIDH public keys $pk_A = (E_A, \phi_A(P_B), \phi_A(Q_B))$ and $pk_B = (E_B, \phi_B(P_A), \phi_B(Q_A))$, where $\{P_A, Q_A = \eta_6(P_A)\}$ and $\{P_B, Q_B = \eta_6(P_B)\}$ are bases of $E[2^{e_A}]$ and $E[3^{e_B}]$ respectively, yields three shared secret (isomorphism classes of) curves.*

Proof. Let α and β be Alice's and Bob's respective secret keys, such that $E_A = E/\langle A\rangle$ and $E_B = E/\langle B\rangle$ for their respective secret kernel generators $A = [\alpha]P_A + Q_A$ and $B = [\beta]P_B + Q_B$. Then their secret isogenies ϕ_A, ϕ_B map E to E_A and E_B respectively. Define related isogenies

$$\phi'_A := \phi_A \circ \eta_6^{-1}, \phi''_A := \phi_A \circ \eta_6^{-2}, \phi'_B := \phi_B \circ \eta_6^{-2}, \phi''_B := \phi_B \circ \eta_6^{-1},$$

which map E to $E/\langle\eta_6^2(A)\rangle, E/\langle\eta_6(A)\rangle, E/\langle\eta_6(B)\rangle$ and $E/\langle\eta_6^2(B)\rangle$ respectively. The properties of η_6 and her knowledge of pk_B then allow Alice to compute the images of her chosen torsion points under ϕ'_B and ϕ''_B as $\phi'_B(P_A) = -\phi_B(Q_A), \phi'_B(Q_A) = \phi_B(P_A) - \phi_B(Q_A)$ and $\phi''_B(P_A) = \phi_B(P_A) - \phi_B(Q_A), \phi''_B(Q_A) = \phi_B(P_A)$. Similarly, Bob can use pk_A to compute $\phi'_A(P_B) = \phi_A(P_B), \phi'_A(Q_B) = \phi_A(P_B)$ and $\phi''_A(P_B) = -\phi_A(Q_B), \phi''_A(Q_B) = \phi_A(P_B) - \phi_A(Q_B)$.

Alice and Bob then perform three standard SIDH-instances, first using the given public keys pk_A, pk_B, and then using the computed torsion point images under ϕ'_A, ϕ'_B and ϕ''_A, ϕ''_B respectively.

Alice obtains three curves by computing the curves

$$\begin{aligned}
&E_B/\langle [\alpha]\phi_B(P_A) + \phi_B(Q_A)\rangle && = E/\langle A, B\rangle,\\
&E_B/\langle [\alpha]\phi'_B(P_A) + \phi'_B(Q_A)\rangle = E_B/\langle -\phi_B(P_A) + [1+\alpha]\phi_B(Q_A)\rangle && = E/\langle \eta_6^2(A), B\rangle, \text{ and}\\
&E_B/\langle [\alpha]\phi''_B(P_A) + \phi''_B(Q_A)\rangle = E_B/\langle -[1+\alpha]\phi_B(P_A) + [\alpha]\phi_B(Q_A)\rangle && = E/\langle \eta_6(A), B\rangle
\end{aligned}$$

in turn. Similarly, Bob computes three curves as $E_A/\langle [\beta]\phi_A(P_B)+\phi_A(Q_B)\rangle, E_A/\langle [\beta]\phi'_A(P_B)+\phi'_A(Q_B)\rangle = E_A/\langle -[1+\beta]\phi_A(P_B) + [\beta]\phi_A(Q_B)\rangle = E/\langle A, \eta_6(B)\rangle$ and $E_A/\langle [\beta]\phi''_A(P_B) + \phi''_A(Q_B)\rangle = E_A/\langle \eta_6(B)\rangle$. Since the quotients of the latter two curves only differ by an application of η_6 from those computed by Alice, they obtain the three shared isomorphism classes of curves

$$E/\langle A, B\rangle,\ E/\langle A, \eta_6(B)\rangle \cong E/\langle \eta_6^2(A), B\rangle \text{ and } E/\langle A, \eta_6^2(B)\rangle \cong E/\langle \eta_6(A), B\rangle.$$

□

It follows that per public key pair in the Jao–Urbanik scheme, Alice and Bob obtain three different SIDH squares and resulting shared secret curves which are identified by their j-invariants. Hence, using the Jao–Urbanik technique with k public keys per party, the shared secret can be computed from the j-invariants of $3k^2$ shared secret curves instead of k^2 curves as in standard k-SIDH. In particular, the secret information shared by both parties is

$$h = H(j_{1,1}||j'_{1,1}||j''_{1,1}|| \dots ||j_{k,k}||j'_{k,k}||j''_{k,k}),$$

where || denotes the concatenation of the invariants $j_{r,s}, j'_{r,s}$ and $j''_{r,s}$. The resulting protocol can be described as follows (and is further visualised in Fig. 4.1).

The set-up is again similar to that in SIDH, where the prime is $p = \ell_A^{e_A}\ell_B^{e_B} f \pm 1$, and often $\ell_A = 2$ and $\ell_B = 3$ for efficiency. For the Jao–Urbanik protocol one then selects a curve E with $j(E) \in \{0, 1728\}$, and the corresponding non-trivial automorphism $\eta \in \{\eta_6, \eta_4\}$. Bases of the relevant torsion subgroups are then fixed such that $\{P_A, Q_A = \eta_6(P_A)\}$ and $\{P_B, Q_B = \eta_6(P_B)\}$. Suppose we have selected $E = E_0$ with $\eta = \eta_6$.

1. **Key Generation**

   - Alice selects k secret integers modulo 2^{e_A}, $\alpha^{(1)}, \ldots, \alpha^{(k)}$, and computes the corresponding isogenies $\phi_{A,r}$ together with their codomains $E_A^{(r)} = E/\langle [\alpha^{(r)}]P_A + Q_A \rangle$.

   - Alice's secret key is then $(\alpha^{(1)}, \ldots, \alpha^{(k)})$, while

   $$(E_A^{(1)}, \phi_{A,1}(P_B), \phi_{A,1}(Q_B)), \ldots, (E_A^{(k)}, \phi_{A,k}(P_B), \phi_{A,k}(Q_B))$$

     denotes her public key.

   - Bob also selects k different secret integers modulo 3^{e_B}, $\beta^{(1)}, \ldots, \beta^{(k)}$, and computes the corresponding isogenies $\phi_{B,r}$ along with their codomain curves $E_B^{(r)} = E/\langle [\beta^{(r)}]P_B + Q_B \rangle$.

   - Bob's secret key is $(\beta^{(1)}, \ldots, \beta^{(k)})$ and his public key is

   $$(E_B^{(1)}, \phi_{B,1}(P_A), \phi_{B,1}(Q_A)), \ldots, (E_B^{(k)}, \phi_{B,k}(P_A), \phi_{B,k}(Q_A)).$$

2. **Key Exchange**

   - Alice sends her public key to Bob, and Bob sends his public key to Alice.

- For each pair $E_A^{(r)}, E_B^{(s)}$, $1 \leq r, s \leq k$, Alice and Bob perform the computations required to obtain the j-invariants $j_{r,s}$, $j'_{r,s}$ and $j''_{r,s}$ of the three resulting shared secret curves. Alice computes $j_{r,s}$ as the j-invariant of the curve $E_B^{(s)}/\langle[\alpha^{(r)}]\phi_{B,s}(P_A) + \phi_{B,s}(Q_A)\rangle$, while Bob computes it via $E_A^{(r)}/\langle[\beta^{(s)}]\phi_{A,r}(P_B)+\phi_{A,r}(Q_B)\rangle$. The curves corresponding to $j'_{r,s}$ are computed as $E_B^{(s)}/\langle-\phi_{B,s}(P_A) + [\alpha^{(r)}+1]\phi_{B,s}(Q_A)\rangle$ and $E_A^{(r)}/\langle-[\beta^{(s)}+1]\phi_{A,r}(P_B) + [\beta^{(s)}]\phi_{A,r}(Q_B)\rangle$. The remaining j-invariant $j''_{r,s}$ is that of curves $E_B^{(s)}/\langle-[\alpha^{(r)}+1]\phi_{B,s}(P_A)+[\alpha^{(r)}]\phi_{B,s}(Q_A)\rangle$ and $E_A^{(r)}/\langle-\phi_{A,r}(P_B)+[\beta^{(s)}+1]\phi_{A,r}(Q_B)\rangle$.

- Hashing the concatenation of the shared j-invariants then gives the secret information shared by Alice and Bob, $h = H(j_{1,1}||j'_{1,1}||j''_{1,1}||\ldots||j''_{k,k})$.

2.2 Genus-2 SIDH

In an attempt to mirror improvements made in elliptic curve cryptography by generalising schemes to using hyperelliptic curves, some schemes originating from isogeny-based elliptic curve cryptography have been modified to using isogenies between higher-dimensional abelian varieties. The shift to considering hyperelliptic curves instead of solely elliptic curves for finding abelian groups suitable for cryptography since the discrete logarithm problem seemed intractable therein was first suggested by Koblitz [Kob89] with the purpose of improving computational efficiency and increasing the choice of groups for such cryptographic applications. For discrete logarithm uses, for example, Jacobians of genus-2 curves yield groups of order double the bit-length of elliptic curve groups over the same finite field so that in many applications genus-2 curves are competitive since primes of half the size can be used to provide the same security level. Evidently, the higher-dimensional setting is only preferable if more complicated arithmetic is acceptable during the execution of the protocol.[5] The mo-

[5]Note that for hyperelliptic curves of genus $g \geq 3$, there exist index calculus attacks [GTTD07] which would require parameters to be scaled to sizes that no longer make the genus-g settings attractive.

tivation for considering isogenies between hyperelliptic curves rather than isogenies simply between elliptic curves is similar, with the expected potential for at least equally as useful trade-offs.

As a first step, in [CGL09], the authors of the CGL hash function briefly explain how their original construction can be used to describe collision-resistant hash functions using any general graph with certain expansion properties, thus implying that supersingular (or, indeed, superspecial) graphs of higher-genus curves can yield further instantiations of the hash. (This idea was formalised in [Tak18] and its issues surrounding collision resistance were fixed in [FT19].) While the size of the supersingular isogeny graph over characteristic p is $O(p)$, abelian surfaces provide a graph with $O(p^3)$ nodes. More precisely, each (isomorphism class of a) supersingular elliptic curve has exactly three neighbours in the 2-isogeny graph while each node in the Richelot isogeny graph depicting $(2,2)$-isogenies between principally polarised supersingular (or superspecial) abelian surfaces has 15 neighbouring (classes of) curves. This difference scales proportionally for larger primes ℓ. Each step in the graph for computing the hash function chooses (for reasons related to non-backtracking and avoiding collisions via simple cycle-construction; see [FT19] and [CDS20]) between two or eight neighbouring curves, respectively, and therefore, might amount to one bit of entropy in the former case compared to up to three bits in the latter. Hence, if one assumes that all computational procedures can be easily translated to the genus-2 case without significant loss in efficiency[6], one could hope for reducing down the finite field characteristic to a third of the bit-length for similar security as in the elliptic curve isogeny case. Note that theoretically, this reduction is even more significant when higher genus curves are used. Hence, it seems promising to examine existing isogeny-based cryptosystems which currently use elliptic curve isogenies, as well as known attacks, for use with general abelian varieties.

In this section, we present a natural generalisation of the SIDH key exchange scheme

[6]This is not yet possible, especially when using (ℓ,ℓ)-isogeny chains where $\ell \neq 2,3$.

to the two-dimensional setting with principally polarised abelian surfaces, Genus-2 SIDH or G2SIDH, as first proposed by Flynn and Ti [FT19; Ti19]. We also explain the complicated key generation procedure.

2.2.1 The Genus-2 SIDH protocol

As part of the setup for the key exchange scheme, one is required to choose a prime which is of the usual form, i.e. $p = 2^{e_A} \cdot 3^{e_B} \cdot f - 1$ where $2^{e_A} \approx 3^{e_B}$ and f is a small, positive integer coprime to 2 and 3.[7] To select the base abelian variety, a PPSSAS is constructed as follows. Define H to be the hyperelliptic curve given by

$$H : y^2 = x^6 + 1\,,$$

which is a double cover of the genus-1 elliptic curve $E : y^2 = x^3 + 1$ via

$$\begin{aligned} \phi_1 : E &\to H & \text{and} \quad \phi_2 : E &\to H \\ (x, y) &\mapsto (x^{1/2}, y), & (x, y) &\mapsto (x^{-1/2}, yx^{-3/2}). \end{aligned}$$

As shown in [CF96, p. 155], the ϕ_i induce a $(2, 2)$-isogeny from the elliptic product E^2 to the Jacobian of H, J_H.

Due to the choice of prime $p \equiv 2 \pmod 3$ and the criterion of [Sil09, Theorem V.4.1(a)], E is a supersingular curve with $(p + 1)^2$ elements. Therefore, the Jacobian $J_H = \mathrm{Jac}(H)$ is a principally polarised superspecial abelian surface, as desired. By [Tat66, Theorem 1], J_H is of cardinality $(p + 1)^4$, so that $J_H(\mathbb{F}_{p^2}) = J_H[2^{e_A}] \times J_H[3^{e_B}] \times J_H[f]$ as a group.

Finally, a random principally polarised superspecial abelian surface J is found as the

[7] Due to practical restrictions when computing (ℓ, ℓ)-isogenies for arbitrary primes ℓ and the standard prime choices made for SIKE, Flynn and Ti only suggest using primes 2 and 3. Theoretically, other choices of primes are possible.

endpoint of a short random walk in the $(2,2)$-isogeny graph. Torsion bases P_1, P_2, P_3, P_4 and R_1, R_2, R_3, R_4, all defined over $\mathbb{F}_{p^2}$, are fixed for $J[2^{e_A}]$ and $J[3^{e_B}]$. Then the key exchange proceeds as follows. An overview of the protocol is given in Fig. 2.3. More details of the secret key generation are provided in Section 2.2.2.

1. **Key Generation**

   - Alice chooses a random maximal 2^{e_A}-isotropic subgroup G_A of $J[2^{e_A}]$ corresponding to a 12-tuple of secret scalars $\alpha_{i,j}$ defining the generators of G_A in terms of the torsion basis P_i and computes the $(2^{e_A}, 2^{e_A})$-isogeny $\phi_A : J \to J/G_A =: J_A$.

   - Alice's secret key are the scalars $\alpha_{i,j}$ (or a different representation of the same information, such as G_A), and her public key is the tuple

   $$\big(J_A, \phi_A(R_1), \phi_A(R_2), \phi_A(R_3), \phi_A(R_4)\big),$$

   where J_A is characterised through the Igusa variants.

   - Similarly, Bob chooses a random maximal 3^{e_B}-isotropic subgroup G_B of $J[3^{e_B}]$ corresponding to scalars $\beta_{i,j}$ as his secret key and computes the $(3^{e_B}, 3^{e_B})$-isogeny $\phi_B : J \to J/G_B =: J_B$.

   - Bob's secret key are the scalars $\beta_{i,j}$ (or an alternative representation), and his public key is the tuple

   $$\big(J_B, \phi_B(P_1), \phi_B(P_2), \phi_B(P_3), \phi_B(P_4)\big),$$

   where J_B is characterised through the Igusa variants.

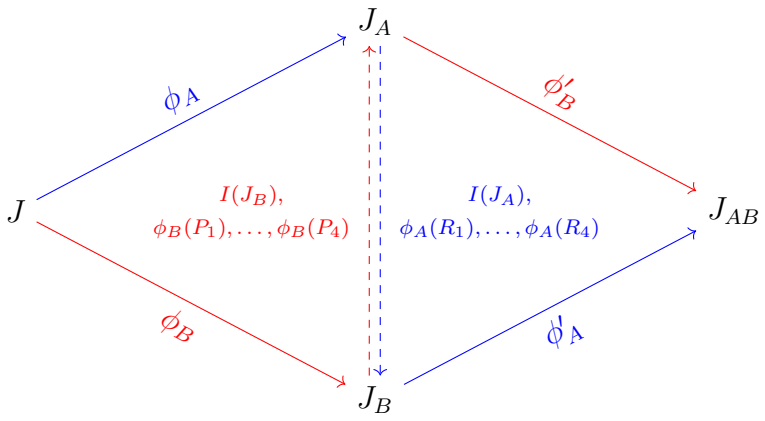

Figure 2.3: The Genus-2 Supersingular Isogeny Diffie–Hellman (G2SIDH) protocol.

2. **Key Exchange**

   - Alice sends her public information to Bob, and he sends her his public key in return.
   - Alice and Bob each use the provided images of the torsion points to compute isogenies parallel to ϕ_A from J_B and ϕ_B from J_A utilising their respective secret scalars. Both these maps share isomorphic codomain varieties $J_{AB} := J/(G_A, G_B)$.
   - Alice and Bob can use (a hash of the Igusa invariants of) the variety J_{AB} as a shared secret.

2.2.2 Key generation for G2SIDH

The key selection process for Bob and Alice is not quite straightforward. Suppose we want to select a secret for Alice to use during the G2SIDH protocol. Ideally, this would mean that Alice uniformly chooses a random maximal 2^{e_A}-isotropic subgroup $G_A \subset J[2^{e_A}]$. This is not immediately feasible.

Flynn and Ti sketch an algorithm for selecting Alice's secret defined by twelve secret

scalars $\alpha_{i,1}, \ldots, \alpha_{i,4} \in \mathbb{Z}/2^{e_A}\mathbb{Z}, 1 \leq i \leq 3$ such that the points

$$A_1 = \sum_{i=1}^{4} [\alpha_{1,i}]P_i, \quad A_2 = \sum_{i=1}^{4} [\alpha_{2,i}]P_i, \text{ and } A_3 = \sum_{i=1}^{4} [\alpha_{3,i}]P_i$$

generate a valid secret kernel $G_A = \langle A_1, A_2, A_3 \rangle$, i.e. a G_A is a maximal 2^{e_A}-Weil isotropic subgroup of $J[2^{e_A}]$. Selecting the $\alpha_{i,j}$ randomly from $\mathbb{Z}/2^{e_A}\mathbb{Z}$ is not possible as this might compromise the isotropy of the subgroup. Firstly, one needs to choose an integer $0 \leq k \leq \lfloor e_A/2 \rfloor$ fixing the orders of the points A_1, A_2 and A_3 to be $2^{e_A}, 2^{e_A-k}$ and 2^k, respectively. To then ensure that the A_i satisfy the triviality condition of the Weil pairing, several congruences need to be satisfied. These congruences are made explicit in [Ti19, Section 2.3] and involve solving several multivariate, underdetermined modular equations. Note that it is also possible for a solution to the congruences not to define three (or, if $k = 0$, two) independent points, resulting in an invalid kernel subgroup. After making the necessary checks however, this strategy will generate a usable secret for Alice.

A procedure for sampling uniformly from the keyspace is not explicitly provided by Flynn–Ti. One could start by calculating a distribution for the selection of k which accurately represents the proportion of rank-2 and different-order of rank-3 subgroups among the valid choices for G_A. Furthermore, the strategy presented here allows the generation of distinct tuples of scalars which eventually define the same subgroups. In order to prevent this issue skewing the uniform generation of secrets, finding a unique representation for each possible subgroup by determining equivalent keys is necessary. An alternative way to sample from the keyspace avoiding these pitfalls and granting uniform sampling is presented in Chapter 5.

2.2.3 Security assumption

The security of G2SIDH relies on the following problem.

Problem 2.10 (Computational G2SIDH problem)**.** *Fix $\ell_A = 2, \ell_B = 3$ and let e_A, e_B be*

integers satisfying $\ell_A^{e_A} \approx \ell_B^{e_B}$. *Further let* J *be a PPSSAS, let*

$$(P_{i,1}, P_{i,2}, Q_{i,1}, Q_{i,2})$$

be a basis for $J[\ell_i^{e_i}]$ *and let* $\phi_i : J \to J_i$ *be an isogeny of degree* $\ell_i^{e_i}$ *for* $i \in \{A, B\}$.

Given J, J_1, J_2 *as well as* $(P_{i,1}, P_{i,2}, Q_{i,1}, Q_{i,2})$ *and* $(\phi_j(P_{i,1}), \phi_j(P_{i,2}), \phi_j(Q_{i,1}), \phi_j(Q_{i,2}))$ *for* $i \neq j \in \{A, B\}$, *determine a PPSSAS* J_3 *such that there exists an isogeny* $\phi : J \to J_3$ *with kernel*

$$\ker(\phi) = \ker(\phi_A) + \ker(\phi_B).$$

Few works are concerned with solving this problem directly. A general algorithm due to Costello and Smith [CS20] examines the pure supersingular isogeny problem for curves of genus 2 and higher of finding an arbitrary isogeny connecting two given varieties. Their strategy is based on a prior algorithm [DG16] which classically computes isogenies between two given elliptic curves. Similarly to Delfs–Galbraith, in [CS20], the authors take random isogeny walks from the two given nodes in the isogeny graph $\Gamma_g(\ell; p)$ consisting of $(\ell, \ldots, \ell)$-isogenies between superspecial principally polarised abelian varieties of dimension g over $\mathbb{F}_{p^2}$ until a distinguished surface is reached. A distinguished surface in this case is one that is (isomorphic to) a product of lower-genus varieties. In the genus-2 setting, this means the algorithm looks for elliptic curve products. Then the problem of connecting the distinguished surfaces can be translated into lower dimensions and solved recursively starting with the Delfs–Galbraith algorithm for elliptic curves and gluing isogenies together for higher genera.

The Costello–Smith algorithm solves the supersingular isogeny problem in genus 2 in $\tilde{O}(p)$ classical computer operations or $\tilde{O}(\sqrt{p})$ quantum calls to the Grover oracle. Note that this algorithm is not well-suited to solve the problem underlying G2SIDH as the secret isogeny between the two given curves is of a fixed degree. The corresponding isogeny walk

is relatively short compared to the isogeny walks usually recovered through the Costello–Smith algorithm, i.e. the walks found through the algorithm are too long to be viable G2SIDH keys. Further, simple adjustments to G2SIDH can prevent an attack using this algorithm by modifying the key generation in such a way that the procedure is aborted and started again as soon as a secret isogeny walk will pass through a vertex which represents an elliptic product.

We provide an adaptive attack on G2SIDH in Chapter 5, but due to the class of Castryck–Decru attacks, and specifically Robert [Rob23; Rob22], it is now possible to attack G2SIDH passively by solving the underlying problem directly. Though some of the countermeasures described in Section 2.1.3 for thwarting the attacks on SIDH seem to also fix G2SIDH at first glance, a thorough examination has not been performed.

Part II

Cryptanalysis

CHAPTER 3

Quantum hidden shift attacks on overstretched SIDH (and other schemes)

Personal contributions: *Chapter 3 is fully based on collaborative work with Péter Kutas, Simon-Philipp Merz and Christophe Petit, published as [KMPW21]. My main contributions were formalising the general framework, working out many parts of the technical details required for instantiating an attack on overstretched SIDH and finally writing up our results and some of the pseudocode algorithms for the publication.*

In this chapter, we introduce a subexponential quantum attack on instances of SIDH with overstretched parameters. This attack can be positioned as a special case within a more general framework, allowing an attacker with access to a certain type of malleability oracle to reduce the inversion of a one-way function to a *hidden shift problem* for which subexponential quantum algorithms exist. Thus we can unify our cryptanalysis of SIDH with previous quantum attacks on isogeny-based protocols, e.g. a method for computing an isogeny between two ordinary elliptic curves, or a similar application of quantum hidden shift

algorithms to CSIDH. The CRS cryptosystem utilising ordinary elliptic curves independently proposed by Couveignes [Cou97] and Rostovtsev–Stolbunov [RS06] was attacked by Childs, Jao and Soukharev [CJS14] in 2010. The Childs–Jao–Soukharev attack showed how to break the CRS scheme in quantum subexponential time using a reduction to an instance of an abelian hidden shift problem. While this attack is tolerable for sufficiently large parameters, the main drawback of the CRS construction is its notable lack of speed. By adapting the CRS scheme to supersingular elliptic curves, the CSIDH protocol does not display the performance issues of CRS and thus allows for larger practical parameters [CLMPR18]. However, there have still been similar cryptanalytic results utilising the idea of reduction to a hidden shift instance for this scheme [BLMP19; BS20; Pei20].

The attack due to Childs, Jao and Soukharev crucially relies on the commutativity of the ideal class groups acting on the endomorphism rings of the relevant elliptic curves over $\mathbb{F}_q$. This motivated the consideration of the full isogeny graph of supersingular elliptic curves for cryptographic use as endomorphism rings of these curves are maximal orders in a quaternion algebra, and hence inherently non-abelian. The result of this idea by Jao and De Feo [JD11] is the SIDH protocol (cf. Section 2.1.1). Before the Castryck–Decru-type attacks [CD23; MMPPW23; Rob23] breaking SIDH, and hence SIKE, with balanced parameters, the best known strategy to finding SIDH secret keys with balanced parameters both classically and quantumly was a claw-finding approach on the isogeny graph [JS19] which did not utilise the additional torsion point information provided by the parties in SIDH. Thus far, the additional torsion point information had only proven useful when finding active GPST-type attacks or in cryptanalysing SIDH with unbalanced parameters; see Section 2.1.2. It was also widely believed that the lack of commutativity in the SIDH protocol would prevent quantum attacks which would effectively reduce the SIDH problem to an abelian hidden shift problem in a variant of the Childs–Jao–Soukharev attack designed for the supersingular case [JD11, Section 5].

This chapter highlights that it is indeed possible to artificially construct a setting in which SIDH is vulnerable to this type of attacks, despite the non-commutativity of the endomorphism rings of supersingular elliptic curves. Though we only show that this is true when non-standard parameters are used, we still disprove the general misconception. We provide the necessary tools and setup for an attack on overstretched parameters, and show that some tricks or improvements to standard subroutines used in our algorithms (such as e.g. the KLPT algorithm [KLPT14]) would advance this method to also succeed for more typical choices of parameters.

In the case of SIDH, the problem we are trying to solve is not the pure isogeny problem but that of recovering a secret isogeny $\varphi : E \to E/K$ where curves E, $E' := E/K$, the degree $d := \deg \varphi$ as well as the images of some torsion points under the secret isogeny are publicly known. The idea underlying our cryptanalysis is to use the provided torsion information in a novel way: We construct an abelian group $G \subseteq \operatorname{End} E$ of E-endomorphisms acting freely and transitively on certain cyclic subgroups of E. These subgroups are kernels of d-isogenies, and therefore they can be mapped to supersingular elliptic curves d-isogenous to E. If we consider the group action of G as an action on the curves and force the endomorphisms in G to be of a certain degree, the public torsion point information allows an adversary to compute the action on E' efficiently under some heuristics. It then only remains to solve an abelian hidden shift problem of two functions mapping G to a set of curves d-isogenous to E containing E' in order to find the kernel K of φ. Thus an attacker recovers the secret isogeny φ.

After providing some necessary background on one-way functions, hard homogeneous spaces and relevant quantum algorithms in Section 3.1, we present our general framework in Section 3.2 by giving sufficient conditions for computing preimages of one-way functions via reduction to a hidden shift problem. The resulting attack on overstretched SIDH in discussed in Section 3.3, while an instantiation of our general framework with the attack of

Childs, Jao and Soukharev and its generalisation to CSIDH is explained in Section 3.4. We conclude this chapter in Section 3.5 with potential improvements and adjustments to our strategy.

3.1 Preliminaries

Throughout this chapter, we say a supersingular elliptic curve E' is *at distance* $d \in \mathbb{Z}^+$ from E if there exists a separable isogeny φ with cyclic kernel of degree d from E to E'.

3.1.1 One-way functions

Informally, a one-way function is a function that is easy to compute on every input but hard to invert given the image of a random input, where the difficulty is to be understood with respect to computational complexity. One-way functions play an integral part in cryptography. For example, one can use a one-way function which has a trapdoor, meaning there exists additional information which makes it easy to invert the function, to construct public key encryption schemes. More formally, we define a one-way function as follows.

Definition 3.1 (One-way function)**.** *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is* one-way *if* f *can be computed by a polynomial-time algorithm, and for all polynomial-time randomised algorithms* F *all positive integers* c *and all sufficiently large* $n = length(x)$*,*

$$\Pr[f(F(f(x))) = f(x)] < n^{-c}.$$

The probability is taken over the choice of x *from the discrete uniform distribution on* $\{0,1\}^n$*, and the randomness of* F*.*

3.1.2 Hard homogeneous spaces and CSIDH

Recall the notion of Couveignes' *hard homogeneous spaces* (HHS) [Cou97], a finite commutative group action for which some operations are easy to compute and others are hard.

Definition 3.2 (Hard homogeneous space)**.** *A* hard homogeneous space *consists of a finite commutative group G acting freely and transitively on some set X. The following tasks are required to be easy (e.g. polynomial-time):*

(i) Compute group operations in G and decide whether elements are equal.

(ii) Sample randomly from G with (close to) uniform distribution.

(iii) Decide validity and equality of a representation of elements of X.

(iv) Compute the action of a group element $g \in G$ on some $x \in X$.

The following problems are required to be hard (e.g. not polynomial-time):

(i) Given $x, x' \in X$, find $g \in G$ such that $g \cdot x = x'$.

(ii) Given $x, x', y \in X$ such that $x' = g \cdot x$, find $y' = g \cdot y$.

Instances of Couveignes' hard homogeneous spaces can be constructed using elliptic curve isogenies and have been the basis of one branch of isogeny-based cryptography which uses the group action we will describe in the following.

Denote the set of all isomorphism classes over $\overline{\mathbb{F}}_q$ of isogenous curves with n points and endomorphism ring $\mathcal{O}$ by $\mathrm{Ell}_{q,n}(\mathcal{O})$, and represent the isomorphism class of a curve E in $\mathrm{Ell}_{q,n}(\mathcal{O})$ by the j-invariant $j(E)$. Any isogeny $\varphi : E \to E_{\mathfrak{b}}$ between curves having the same endomorphism ring in $\mathrm{Ell}_{q,n}(\mathcal{O})$ is determined by E and $\ker \varphi$ up to isomorphism. This kernel corresponds to an ideal $[\mathfrak{b}]$ in $\mathcal{O}$. Recall that the ideal class group of $\mathcal{O}$, $\mathrm{Cl}(\mathcal{O})$, is the

quotient group of the abelian group of fractional $\mathcal{O}$-ideals under ideal multiplication and all principal fractional $\mathcal{O}$-ideals. Since principal ideals in $\mathcal{O}$ correspond to isomorphisms, ideals that are equivalent in $\mathrm{Cl}(\mathcal{O})$ induce the same isogeny up to isomorphism. Hence, we have a well-defined group action

$$\begin{aligned} \cdot : \mathrm{Cl}(\mathcal{O}) \times \mathrm{Ell}_{q,n}(\mathcal{O}) &\to \mathrm{Ell}_{q,n}(\mathcal{O}), \\ ([\mathfrak{b}], j(E)) &\mapsto j(E_{\mathfrak{b}}), \end{aligned}$$

which is free and transitive ([Wat69, Theorem 4.5] and [Sch87, Erratum Theorem 4.5]).

Given two elliptic curves E, E' in $\mathrm{Ell}_{q,n}(\mathcal{O})$ up to isomorphism, it is in general assumed to be hard to find an isogeny $\varphi : E \to E'$.

A similar construction can be performed with endomorphism rings of supersingular curves. This occurrence of hard homogenous spaces is used for the CSIDH protocol [CLMPR18] proposed for post-quantum non-interactive key exchange. Since the endomorphism rings of such curves are orders in a quaternion algebra, they are non-commutative and hence yield a group action with less desirable properties than in the construction for ordinary curves. Therefore, the authors suggest a restriction of the endomorphism ring to the subring of $\mathbb{F}_p$-rational endomorphisms which is an order in an imaginary quadratic field, and as such commutative. Again, the ideal class group of this order $\mathcal{O}$ acts on $\mathrm{Ell}_p(\mathcal{O})$, the set of all isomorphism classes of supersingular isogenous curves over $\mathbb{F}_p$ with $\mathbb{F}_p$-rational endomorphism ring (isomorphic to) $\mathcal{O}$.

Given that the set $\mathrm{Ell}_p(\mathcal{O})$ is non-empty, the group action is free and transitive (see [CLMPR18, Theorem 7] summarising results from [Wat69; Sch87]), and can be used to perform a Diffie–Hellman-type key exchange. Note that CSIDH is strictly speaking not an instance of a HHS as it is not always possible to check whether two arbitrary elements

are equal in polynomial time or compute the group action efficiently for *all* group elements. Indeed, De Feo and Meyer [DM20] give an example of an element which does not adhere to some of the algorithmic requirements of the definition of HHS. In CSIDH, these issues are avoided by computing the group structure of the class group in question as well as the class group relations; for the only suggested set of CSIDH-parameters (using a prime of roughly 512 bits), this has been done for example by Beullens, Kleinjung and Vercauteren [BKV19] so that CSIDH-512 is possible with current computational resources. However, these differences have little impact in practice on the CSIDH protocol itself.

For CSIDH, fix a large prime $p = 4 \cdot \ell_1 \dots \ell_n - 1$ for some small, distinct primes ℓ_i, and consider the supersingular elliptic curve $E : y^2 = x^3 + x$ defined over $\mathbb{F}_p$ with $\mathbb{F}_p$-rational endomorphism ring $\mathcal{O} := \mathrm{End}_p(E) = \mathbb{Z}[\pi]$, where π denotes the Frobenius endomorphism of E.

Note that all supersingular Montgomery curves $E_A : y^2 = x^3 + Ax^2 + x$ for $A \in \mathbb{F}_p$ defined over $\mathbb{F}_p$ are elements of $\mathrm{Cl}(\mathcal{O}) \cdot E$ and furthermore uniquely represented (up to isomorphism over $\mathbb{F}_p$) by the coefficient A. This observation yields the following key exchange.

1. **Key Generation**

   - Alice chooses an n-tuple $(e_1, \dots, e_n)$ with randomly sampled integers $-m \leq e_i \leq m$ for some bound m to generate her private key, an ideal class $[\mathfrak{a}] \in \mathrm{Cl}(\mathcal{O})$ such that $[\mathfrak{a}] = [\mathfrak{l}_1^{e_1} \dots \mathfrak{l}_n^{e_n}]$ for the ideals $\mathfrak{l}_i = (\ell_i, \pi - 1)$.

   - Alice computes the Montgomery curve $E_A := [\mathfrak{a}] \cdot E : y^2 = x^3 + Ax^2 + x$ and takes $A \in \mathbb{F}_p$ to be her public key.

   - Bob analogously produces his private key $[\mathfrak{b}]$ and the corresponding curve E_B, taking B as his public key.

2. **Key Exchange**

   - Alice and Bob exchange their public keys.
   - After verifying that the curve E_B is indeed in $\mathrm{Ell}_p(\mathcal{O})$, Alice computes the secret curve $[\mathfrak{a}]\cdot E_B = [\mathfrak{a}][\mathfrak{b}]\cdot E$. Bob proceeds mutatis mutandis and computes the secret curve as $[\mathfrak{b}]\cdot E_A$.
   - By commutativity of $\mathrm{Cl}(\mathcal{O})$, Alice and Bob have computed the same curve $[\mathfrak{a}][\mathfrak{b}]\cdot E = [\mathfrak{b}][\mathfrak{a}]\cdot E$, and can use the Montgomery coefficient S such that $[\mathfrak{a}][\mathfrak{b}]\cdot E : y^2 = x^3 + Sx^2 + x$ as the shared secret.

There have been multiple proposals to attack concrete parameter suggestions for CSIDH with quantum algorithms. Peikert [Pei20], for example, uses Kuperberg's collimation sieve algorithm to solve the hidden shift instance with quantum accessible classical memory and subexponential quantum time, a strategy independently also explored by Bonnetain–Schrottenloher [BS20].

3.1.3 Solving the hidden shift problems with quantum algorithms

First, we recall what is meant when two functions are said to be *shifts of each other*, or equivalently that these two functions *hide a shift*.

Problem 3.3 (Hidden shift problem)**.** *Let $F_0, F_1 : G \to X$ be two functions defined on some group G, such that there exists some $s \in G$ satisfying $F_0(g) = F_1(g \cdot s)$ for all $g \in G$. The* hidden shift problem *is to find s given oracle access to the functions F_0 and F_1.*

Assume we are given two functions $F_0, F_1 : G \to X$ mapping a finite abelian group G to some finite set X. Multiple approaches utilising quantum computations have been proposed to solve the hidden shift problem.[1] Some of these works have considered different

[1] Before trying to immediately solve a hidden shift instance on F_0 and F_1, in some situations it might be

group structures as well as variations on the promise. We summarise some relevant quantum algorithms solving the injective abelian hidden shift problem, i.e. where the functions F_i are injective functions and G is abelian.

The first quantum subexponential algorithm is due to Kuperberg [Kup05] and reduces the hidden shift problem to the hidden subgroup problem in the dihedral group $D_G \simeq \mathcal{C}_2 \ltimes G$, i.e. to finding a subgroup of D_G such that a function obtained from combining the input functions of the hidden shift problem is constant exactly on its cosets. It requires quantum subexponential time, namely $2^{\mathcal{O}(\sqrt{\log|G|})}$ quantum queries, for a finite abelian group G.[2] A modification of this method proposed by Regev [Reg04] reduces the memory required by Kuperberg's approach (from super-polynomial to polynomial) while keeping the running time quantum subexponential. Furthermore, Childs, Jao, and Soukharev [CJS14] use Kuperberg's algorithm to construct elliptic curve isogenies in subexponential time; we give a brief account of their resulting attack on hard homogenous spaces using hidden shift algorithms in Section 3.4. In [CJS14, Appendix A], a general subexponential quantum algorithm using time $L_{|G|}(1/2, \sqrt{2})$ is presented for finding hidden shifts in finite abelian groups G. Another, slightly faster algorithm, the collimation sieve, using polynomial quantum space was proposed later by Kuperberg [Kup13]. In this variant, parameter trade-offs between classical and quantum running time and quantumly accessible memory are possible.

These algorithms for solving the hidden shift problem when G is abelian generally begin by producing some random quantum states, each with an associated classical label

advisable to decide whether these functions actually satisfy the hidden shift promise fully or at least for a large proportion of the elements in the domain first. One can use a property testing algorithm to determine whether this relationship exists between the two given functions. For example, Friedl, Santha, Magniez and Sen [FSMS09] devise a testing algorithm which has perfect completeness. In our setting, this implies that two functions which are shifts of each other will always be accepted by the tester, preventing the occurrence of false negatives.

[2] Note that this complexity satisfies our definition of subexponential time: Let $N := |G|$, and suppose Kuperberg's algorithm runs in time $2^{f(N)} = 2^{O(\sqrt{\log N})}$. Then we also have $f(N) = o(\log N)$ since for any $\epsilon > 0$ we can choose $N_\epsilon = \max\{N_c, 2^{(c/\epsilon)^2}\}$, where c and N_c are the constants from the definition of what it means that $f(N) = O(\sqrt{\log N})$, so that $|f(N)| \leq c\sqrt{\log N} \leq \epsilon\sqrt{\log N}\sqrt{\log N} = \epsilon \log N$ for $N \geq N_\epsilon$.

or tag, by evaluating the group action on a uniform superposition over the group G. For this generation of states, oracle access to the two functions F_0 and F_1 is needed. Then, the hidden shift s is extracted bitwise through performing measurements on specific quantum states (i.e. ones with desirable labels) which are generated from the random states via some sieving algorithm.

3.2 Malleability oracles and hidden shift attacks

We now introduce the notion of a *malleability oracle* for a one-way function. Under some conditions, such an oracle (which might be readily available for some known one-way functions) allows the computation of preimages of given elements in quantum subexponential time by reduction to the hidden shift problem.

3.2.1 Malleability oracles

Recall the definition of a free and transitive group action.

Definition 3.4 (Group action)**.** *Let G be a group with neutral element e, and let $\mathcal{I}$ be a set. A* (left) group action $\star$ *of G on $\mathcal{I}$ is a function*

$$\star\colon G \times \mathcal{I} \to \mathcal{I}, \quad (g, x) \mapsto g \star x,$$

that satisfies $e \star x = x$, and $gh \star x = g \star (h \star x)$ for all $x \in \mathcal{I}$ and $g, h \in G$.

The group action is called transitive *if and only if $\mathcal{I}$ is non-empty and for every pair of elements $x, y \in \mathcal{I}$ there exists $g \in G$ such that $g \star x = y$. The group action is called* free *if and only if $g \star x = x$ implies $g = e$.*

Next, we define an oracle capturing the main premise required for our strategy to compute preimages of one-way functions.

Definition 3.5 (Malleability)**.** *Let $f : \mathcal{I} \to \mathfrak{O}$ be an injective (one-way) function and let $\star$ be the action of a group G on $\mathcal{I}$. A* malleability oracle *for G at $o := f(i)$ provides the value of $f(g \star i)$ for any input $g \in G$, i.e. the malleability oracle evaluates the map*

$$g \mapsto f(g \star i).$$

We call the function f malleable*, if a malleability oracle is available at every $o \in f(\mathcal{I})$.*

To abstract away from the notion of group actions, it is possible to define malleability in terms of more general knowledge relating inputs and outputs of the one-way function f respectively. In the following we will concentrate on the group action-based model as defined above since it facilitates the construction of a polynomial-time malleability oracle in the context of SIDH with overstretched parameters; see Section 3.3. In Section 3.4 we describe other contexts where such an oracle arises naturally.

For the remainder of this chapter, we will denote the action of a group element $g \in G$ on a set element $i \in \mathcal{I}$ by $g \cdot i$.

3.2.2 Reduction to hidden shift problem

Given a malleability oracle at $o = f(i)$, computing a preimage of o reduces to a hidden shift problem in the following case.

Theorem 3.6. *Let $f : \mathcal{I} \to \mathfrak{O}$ be an injective (one-way) function and let G be a group acting transitively on $\mathcal{I}$. Given a malleability oracle for G at $o := f(i)$, the preimage of o can be computed by solving a hidden shift problem.*

Proof. Given $o \in f(\mathcal{I})$, the goal is to compute i such that $f(i) = o$. Let k be an arbitrary

but fixed element in $\mathcal{I}$ and define

$$F_k \ : \ G \to \mathfrak{O} \ , \ \theta \mapsto f(\theta \cdot k).$$

Since f is an injective function, $i = f^{-1}(o)$ is unique and thus F_i is well-defined. Moreover, the malleability oracle allows us to evaluate the function F_i on any $\theta \in G$, as $F_i(\theta) = f(\theta \cdot i)$.

Fix some arbitrary $j \in \mathcal{I}$. Since we know j, we can evaluate F_j on any group element θ by evaluating $f(\theta \cdot j)$ via simply computing the group action. Due to the transitivity of the group action of G, there exists $\sigma \in G$ such that $i = \sigma \cdot j$. Since for all $\theta \in G$

$$F_i(\theta) = f(\theta \cdot i) = f(\theta\sigma \cdot j) = F_j(\theta\sigma),$$

the functions F_j and F_i are shifts of each other. Hence, solving the hidden shift problem for F_i and F_j allows us to recover σ, and thus to compute $i = \sigma \cdot j$. □

The following corollary will be used in our attack on overstretched SIDH. Note that the precise complexity of our strategy for inverting a one-way function depends on which one of the quantum algorithms (cf. Section 3.1.3) is selected for solving the hidden shift as well as the cost of accessing the malleability oracle.

Corollary 3.7. *Let $f : \mathcal{I} \to \mathfrak{O}$ be an injective (one-way) function and let G be a finitely generated abelian group acting freely and transitively on $\mathcal{I}$. Given a malleability oracle for G at $o := f(i)$, the preimage of o can be computed in quantum subexponential time.*

Proof. To obtain a hidden shift instance solvable by a subexponential quantum algorithm such as Kuperberg's, we only have to show that for every $k \in \mathcal{I}$ the function $F_k(\theta) = f(\theta \cdot k)$ is injective. Then the claim follows from Theorem 3.6 and the discussion in Section 3.1.3.

Suppose that $F_k(g) = f(g \cdot k) = f(h \cdot k) = F_k(h)$ for some $g, h \in G$. Since f is injective and the group action is free, this implies $g = h$. □

3.3 Attack on overstretched SIDH instances in quantum subexponential time

Despite the non-commutative nature of SIDH, we show in this section that one can find an abelian group action on its private key space. Moreover for sufficiently *overstretched* SIDH parameters, the torsion point information revealed in the protocol allows us to build a malleability oracle for this group action. More precisely this means that for our quantum attack to work in the SIDH setting, we will need to relax the balancedness condition, i.e. the recommendation that $\ell_A^{e_A} \approx \ell_B^{e_B}$ for parameters which was discussed in Section 2.1.1, and require one torsion to be larger than the other by a certain factor. If we let $\{N_1, N_2\} = \{\ell_A^{e_A}, \ell_B^{e_B}\}$ denote the two torsion orders of the parties involved, we further require $N_1 N_2 \gg p$ which prohibits choosing p as originally suggested by Jao–De Feo (and fixed in SIKE via the recommended parameter sets). We call this variant of SIDH *overstretched*. Note that this variant of SIDH is still polynomial-time as long as N_1 and N_2 are smooth numbers, albeit much slower in practice than with the suggested parameters. This gives rise to an attack using quantum subexponential hidden shift algorithms as outlined in Section 3.2.2.

We present our results regarding SIDH as follows: First we sketch our approach to exploiting the torsion point information in Section 3.3.1. We then overcome some technical issues in Sections 3.3.2 to 3.3.4. These issues require small tweaks to our general approach, and we summarise the resulting algorithm in Section 3.3.5. Finally in Section 3.3.6, we present a hybrid approach to combine guessing part of the secret and computing the remaining part using our new attack; this allows us to slightly extend the attack to further parameter sets.

Throughout this section, we let p be a prime with $p \equiv 3 \pmod 4$, E the supersingular elliptic curve with j-invariant 1728 defined over $\mathbb{F}_{p^2}$ given by the equation $y^2 = x^3 + x$, and let $\mathcal{O} = \mathrm{End}(E)$ be its endomorphism ring. Note that $\mathcal{O}$ is well-known; it is the $\mathbb{Z}$-module generated by $1, \iota, \frac{1+\pi}{2}$ and $\frac{\iota+\iota\pi}{2}$ where ι denotes the non-trivial automorphism[3] $(x, y) \mapsto (-x, iy)$ of E, and π is the Frobenius endomorphism $(x, y) \mapsto (x^p, y^p)$. Recall further that we use N_1 and N_2 to denote the torsion orders in SIDH so that $N_1, N_2 \mid p - 1$.

Remark 3.8. *The attack we describe can be expanded to other curves that are close to E by computing the isogeny to E and translating the problem back to E with $j(E) = 1728$.*

3.3.1 Overview of the attack

Let $\mathcal{I}$ be the set of cyclic N_1-order subgroups of E, and let $\mathfrak{O}$ be the set of j-invariants of all supersingular curves that are N_1-isogenous to E. Let f be the function sending any element of $\mathcal{I}$ to the j-invariant of the codomain of its corresponding isogeny, i.e.

$$f : \mathcal{I} \to \mathfrak{O}, \quad K \mapsto j(E/K). \tag{3.1}$$

The function f can be efficiently computed on any input using Vélu's formulae [Vél71], provided N_1 is sufficiently smooth and that the N_1-torsion is defined over a sufficiently small extension field of $\mathbb{F}_p$. In SIDH, the latter is achieved by choosing $N_1 \mid p - 1$ but this is true more generally for sufficiently powersmooth N_1.

On the other hand, inverting f amounts to finding an isogeny of degree N_1 from E to a curve in a given isomorphism class, or equivalently to finding the subgroup of E defining this isogeny. This problem is closely related to the pure isogeny problem whose conjectured hardness is at the heart of isogeny-based cryptography.

[3] We have previously introduced this order-four automorphism as η_4 in Section 2.1.4 where it was utilised in a version of the Jao–Urbanik scheme [UJ20].

In the SIDH protocol, additional torsion point information is transmitted publicly as part of the exchange, and thus also given to adversaries. For the security proof it is assumed that the variant Problem 2.3 of the following problem with $N_1 \approx N_2$ is hard [JD11]. As discussed in Section 2.1.2, this assumption has been disproved.

Problem 3.9. *Let p be a large prime, let N_1 and N_2 be two smooth coprime integers such that $E[N_1]$ and $E[N_2]$ can be represented efficiently, let $K \in \mathcal{I}$ be a cyclic subgroup of order N_1 of E chosen uniformly at random, and let $\varphi : E \to E/K$. Given the supersingular elliptic curves E and E/K together with the restriction of φ to $E[N_2]$, compute K.*

Our attack exploits the information provided by the restriction of the secret isogeny to $E[N_2]$ to construct a malleability oracle for f at the (unknown) secret. Following the framework outlined in Section 3.2, this gives rise to an attack on *overstretched* SIDH.

Let G be a subgroup of $(\mathcal{O}/N_1\mathcal{O})^*$. Then G induces a well-defined group action on $\mathcal{I}$ given by

$$G \times \mathcal{I} \to \mathcal{I} \ , \ (\theta, K) \mapsto \theta(K).$$

Indeed, the degree of any non-trivial representative θ in G is coprime to N_1 and thus preserves the order of any generator of K.

Note that the full group $(\mathcal{O}/N_1\mathcal{O})^*$ contains the invertible elements in $\mathrm{End}(E[N_1])$ (see proof of [Voi18, Theorem 42.1.9]) and is isomorphic to $GL_2(\mathbb{Z}/N_1\mathbb{Z})$. Therefore, it is not abelian. For our attack to succeed, we do however require an abelian subgroup G acting on $\mathcal{I}$ such that G acts freely and transitively on the orbit of a kernel of an isogeny $E \to E/K$ under this group action, as well as one element in this orbit. This leads to the following task.

Task 3.10. *Let $K \in \mathcal{I}$ be any cyclic subgroup of E of order N_1 chosen uniformly at random and let $\varphi : E \to E_A := E/K$. Compute an element $L \in \mathcal{I}$ and an abelian subgroup G of $(\mathcal{O}/N_1\mathcal{O})^*$ such that G acts freely and transitively on the orbit $G \cdot L$, f is injective on $G \cdot L$*

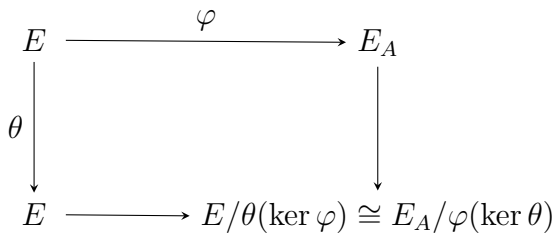

Figure 3.1: An SIDH square with the endomorphism θ where the isogeny φ and the endomorphism θ are of coprime degrees.

and $j(E_A)$ *is contained in* $f(G \cdot L) \subset \mathfrak{O}$.

We solve this task in Section 3.3.2. More precisely, we find three subsets of $\mathcal{I}$ restricted to which f is injective, and we give abelian groups that induce the required action on these subsets. Furthermore, the image of f restricted to one of these three subsets of $\mathcal{I}$ will always contain $j(E/K)$.

In order to apply our general framework from Section 3.2, it remains to construct a malleability oracle for f at $j(E/K)$ for any secret $K \in \mathcal{I}$. To do so, we use both the torsion point information provided in the SIDH protocol and a solution to the following task.

Task 3.11. *Given an endomorphism* $\theta \in G$ *of degree coprime to* N_1 *and an integer* N_2 *coprime to* N_1*, compute an endomorphism* θ' *of degree* N_2 *such that* θ *and* θ' *induce the same action on the set* $\mathcal{I}$ *of cyclic subgroups of* $E[N_1]$ *of order* N_1.

In Section 3.3.4, we first give a direct solution to a variation of this task when using sufficiently overstretched and unbalanced parameters, i.e. $N_2 > p^2 N_1^4$. However, we then show that it suffices to lift elements of πG where π is the Frobenius map. A solution to Task 3.11 for these elements requiring only $N_2 > pN_1^4$ is described in Section 3.3.4.

The following lemma results from the coprimality of $\deg\theta$ and N_1 and is depicted in Figure 3.1.

Lemma 3.12. *Let $\varphi : E \to E_A$ be an isogeny of degree N_1 and let $\theta \in End(E)$ be of degree coprime to N_1. Then $E_A/\varphi(\ker\theta)$ is isomorphic to $E/\theta(\ker\varphi)$.*

Let N_3 be the degree of θ. We cannot compute the curve $E/\theta(\ker\varphi)$ in general without the knowledge of the isogeny φ or its action on the N_3-torsion. However, we can compute the curve if we find an endomorphism θ' of degree N_3' such that θ and θ' have the same action on the N_1-torsion and $\varphi|_{E[N_3']}$ is known. This is the motivation behind Task 3.11, as we know the action of φ on the N_2-torsion in Problem 3.9. A solution to this task yields a malleability oracle for f with respect to the previously described group action of G on $\mathcal{I}$ in the SIDH setting.

Algorithm 1: Computation of $f(\theta(K))$, given $f(K)$ and $\theta \in G$

Let $\varphi : E \to E_A := E/K$ be an isogeny of degree N_1, let N_2 be coprime to N_1 and $G \subset (\mathcal{O}/N_1\mathcal{O})^*$ one of the abelian groups as in Task 3.10 that acts freely and transitively on K.
Input: E, $f(K) = j(E_A)$, $\varphi|_{E[N_2]}$ and $\theta \in G$.
Output: $f(\theta(K)) = j(E/\theta(K))$.

1 Compute endomorphism θ' of degree N_2 having the same action as θ on cyclic N_1-order subgroups of $E[N_1]$ as provided by a solution to Task 3.11;
2 Determine $\varphi(\ker\theta')$, using the knowledge of φ on $E[N_2]$;
3 Compute $j(E_A/\varphi(\ker\theta')) = j(E/\theta(K))$;
4 **return** $f(\theta(K)) = j(E/\theta(K))$

We outline the construction of the malleability oracle in Algorithm 1. Correctness will follow from the proof of Proposition 3.37 given a suitable choice of the acting group G which we will discuss in Section 3.3.2.

For parameters that allow us to construct a malleability oracle, we can then solve Problem 3.9 underlying SIDH-like protocols via a reduction to an injective abelian hidden shift problem using the framework introduced in Section 3.2.2.

Informal result 3.13. *Suppose the parameters allow the efficient solution of Task 3.11, then Problem 3.9 can be solved in quantum subexponential time.*

We use the remainder of this section to prove this result formally under certain assumptions. To this end, we first give solutions to Task 3.10 and, for some parameters, to a variant of Task 3.11. More precisely, we show in Section 3.3.3 that it is sufficient to lift elements from πG instead of G. For this case, we then give a more efficient lifting procedure requiring unbalanced and overstretched parameters. We construct a malleability oracle using the torsion point information provided in SIDH and a subroutine solving our variant of Task 3.11. Apart from some technical details that we will address in the following, the informal result follows from Corollary 3.7. An overview of the attack is depicted in Algorithm 2.

Algorithm 2: Solving SIDH's underlying hardness assumption via an abelian hidden shift problem

Let $\varphi : E \to E/K$ be an N_1-isogeny and $N_2 \in \mathbb{Z}$ such that $\gcd(N_1, N_2) = 1$.
Input: E, E/K, $\varphi(E[N_2])$.
Output: Isogeny $E \to E'$, where $j(E') = j(E/K)$.

1 Compute an abelian group $G \subset (\mathcal{O}/N_1\mathcal{O})^*$ acting freely and transitively on the orbit $G(K)$ and some $J \in G(K) \subset \mathcal{I}$;
2 Define $F_K : G \to \mathfrak{O}, g \mapsto f(g(K))$ and $F_J : G \to \mathfrak{O}, g \mapsto f(g(J))$;
3 Compute injective abelian hidden shift $\theta \in G$ of F_K and F_J, i.e. $\theta \in G$ such that $F_K(g) = F_J(\theta g)$ for all $g \in G$, using a quantum algorithm such as Kuperberg's. To this end, one evaluates F_K using Algorithm 1 and F_J using the knowledge of J;
4 **return** Isogeny $E \to E/\theta(J)$

3.3.2 A free and transitive group action

Recall that E is the supersingular curve with j-invariant 1728, given by the equation $y^2 = x^3 + x$. In this section we provide a solution to Task 3.10. For simplicity, we treat N_1 as a power of 2, but the results generalise to any power of a small prime. A generalisation to powers of 3 is sketched at the end of this section.

We provide the solution by identifying three subsets of $\mathcal{I}$ that are orbits under a free and transitive action of abelian subgroups of $(\mathcal{O}/N_1\mathcal{O})^*$. More precisely, let $P \in E$ such that

$\langle P, \iota(P)\rangle = E[N_1]$ where ι denotes the non-trivial automorphism of E. Let $Q := P + \iota(P)$ and define the following three subsets of $\mathcal{I}$.

$$\begin{aligned}
\mathcal{I}_1 &:= \{\langle P + [\alpha]\iota(P)\rangle \mid \alpha \text{ even } \} \\
\mathcal{I}_2 &:= \left\{ \langle Q + [\alpha]\iota(Q)\rangle \mid \alpha \text{ even and } \alpha \in \left[0, \frac{N_1}{2} - 1\right] \right\} \\
\mathcal{I}_3 &:= \left\{ \langle Q + [\alpha]\iota(Q)\rangle \mid \alpha \text{ even and } \alpha \in \left[\frac{N_1}{2}, N_1 - 1\right] \right\}
\end{aligned}$$

Recall the function f defined in Equation (3.1), mapping cyclic subgroups of $E[N_1]$ of order N_1 to j-invariants of curves at distance N_1 from E,

$$f : \mathcal{I} \to \mathfrak{O}, \quad K \mapsto j(E/K).$$

We will show that restricting the function f to any of the subsets $\mathcal{I}_1$, $\mathcal{I}_2$, or $\mathcal{I}_3$ yields an injective function and we will prove that $f\left(\bigcup_i \mathcal{I}_i\right)$ contains all j-invariants of curves which can arise as SIDH public keys. Furthermore, we will see that

$$G_0 := \{a + b\iota \mid a \text{ odd}, b \text{ even}\} / N_1\mathcal{O}^*$$

acts transitively on $\mathcal{I}_1$. In order to ensure that the action is free, we identify two endomorphisms $a + b\iota$ and $a' + b'\iota$ in G_0 if there exists an odd $\lambda \in \mathbb{Z}/N_1\mathbb{Z}$ such that $a \equiv \lambda a' \pmod{N_1}$ and $b \equiv \lambda b' \pmod{N_1}$. We denote the resulting group by G.

In order to define free and transitive group actions on $\mathcal{I}_2$ and $\mathcal{I}_3$ we define

$$H_0 := \{a + b\iota \mid a \text{ odd}, b \text{ even } \} / (N_1/2)\mathcal{O}^*$$

similarly to G_0. Again, we identify two endomorphisms $a + b\iota$ and $a' + b'\iota$ in H_0 if there exists an odd $\lambda \in \mathbb{Z}/(N_1/2)\mathbb{Z}$ such that $a \equiv \lambda a' \pmod{N_1/2}$ and $b \equiv \lambda b' \pmod{N_1/2}$ and

obtain the resulting group H. The group H will act freely and transitively on $\mathcal{I}_2$ and $\mathcal{I}_3$. Hence, one of these three options will always be a solution to Task 3.10.

The map f is based on the well-known correspondence between $\mathcal{I}$ and curves at distance N_1 from E. However, this correspondence is not necessarily one-to-one. In particular, if E has a non-scalar endomorphism of degree N_1^2, this endomorphism can be decomposed as $\hat{\tau_1} \circ \tau_2$, where τ_1 and τ_2 are non-isomorphic isogenies of degree N_1 from E to E itself. For small enough N_1, the following lemma shows that two kernels correspond to the same curve if and only if they are linked by the automorphism ι.

Lemma 3.14. *Suppose that $N_1^2 < \frac{p+1}{4}$. Then the only endomorphisms of degree N_1^2 of E are $[N_1]$ and $[N_1] \cdot \iota$, where $\iota : E \to E, (x, y) \mapsto (-x, iy)$ is the non-trivial automorphism.*

Proof. Due to the condition $N_1^2 < \frac{p+1}{4}$, an endomorphism θ of degree N_1^2 lies in $\mathbb{Z}[\iota]$. Let $\theta = a + b\iota$ for some $a, b \in \mathbb{Z}$. Then the degree of θ is $a^2 + b^2$. Now we have to prove that the only ways to decompose N_1^2 as a sum of two squares are trivial, i.e. $N_1^2 = N_1^2 + 0^2 = 0^2 + N_1^2$.

Let $N_1 = 2^k$, and we prove the statement by induction on k. For $k = 1$ the statement is trivial. Suppose that $k > 1$ and that $N_1^2 = a^2 + b^2$. Then a and b cannot both be odd as N_1^2 is divisible by four. If they were both even, then dividing by four yields a decomposition of $(N_1/2)^2 = (a/2)^2 + (b/2)^2$. By the induction hypothesis, this decomposition is trivial implying that N_1^2 can also only be decomposed in a trivial way. □

Corollary 3.15. *Suppose that $N_1^2 < \frac{p+1}{4}$. Let φ and φ' be two isogenies of degree N_1 from E to a curve E'. Then either $\ker \varphi = \ker \varphi'$ or $\ker \varphi = \iota(\ker \varphi')$.*

Proof. Consider the endomorphism $\tau = \hat{\varphi'} \circ \varphi$ of E. The degree of τ is N_1^2, so $\tau = [N_1]$ or $\tau = [N_1] \cdot \iota$ by Lemma 3.14. In the former case, the isogenies φ and φ' are identical by the uniqueness of the dual. In the latter case, we have $\ker \varphi = \iota(\ker \varphi')$. □

Thus, an element in the image of f has precisely one preimage if the kernel of the corresponding isogeny is fixed by the automorphism ι.

Finding an abelian group with $\mathcal{I}_1$:

Now, we will give the free and transitive group action on $\mathcal{I}_1$ and show that f restricted to $\mathcal{I}_1$ is injective.

Let P be a point such that $\{P, \iota(P)\}$ is a basis of $E[N_1]$ and recall

$$\mathcal{I}_1 := \{\langle P + [\alpha]\iota(P)\rangle \mid \alpha \text{ even } \}.$$

We show that the restriction of f to $\mathcal{I}_1$ is injective.

Proposition 3.16. *Let $j(E) = 1728$ and suppose that $N_1^2 < \frac{p+1}{4}$. The restriction of f to $\mathcal{I}_1$ is injective.*

Proof. We apply Corollary 3.15 to show that the codomains of isogenies with kernel in $\mathcal{I}_1$ are pairwise non-isomorphic curves. It is clear that $P + [\alpha]\iota(P)$ and $P + [\alpha']\iota(P)$ are not scalar multiples of each other if $\alpha \neq \alpha'$ as $P, \iota(P)$ generate $E[N_1]$. It remains to show that for any even α, α', the points $P + [\alpha]\iota(P)$ and $\iota(P + [\alpha']\iota(P)) = [-\alpha']P + \iota(P)$ are not scalar multiples of each other. Suppose there exists an odd λ such that

$$P + [\alpha]\iota(P) = \lambda([-\alpha']P + \iota(P)).$$

Note that we can restrict to odd λ as the order of both points is N_1. Since $\{P, \iota(P)\}$ is a basis of the N_1-torsion, this implies that $1 \equiv -\lambda\alpha' \pmod{N_1}$. Since α' is even this is a contradiction concluding the proof. □

Clearly, $f(\mathcal{I}_1)$ does not include all elliptic curves at distance N_1 from E, i.e. all curves

in $f(\mathcal{I})$. Every curve at distance N_1 from E which could be a SIDH public key[4] is of the form $E/\langle P + [\alpha]\iota(P)\rangle$ for some $\alpha \in \mathbb{Z}/N_1\mathbb{Z}$, which follows from the observation that the curves $E/\langle[\beta_1]P+[\beta_2]\iota(P)\rangle$ and $E/\langle[-\beta_2]P+[\beta_1]\iota(P)\rangle$ are isomorphic since their kernels are linked by ι. We first restrict ourselves to defining a free and transitive group action on $\mathcal{I}_1$ and define the free and transitive group action on the kernels corresponding to the remaining curves later.

Recall that E is a curve with well-known endomorphism ring, and we are interested in the endomorphisms that are of degree coprime to N_1. While there are infinitely many such endomorphisms, we are only concerned with their action on $E[N_1]$, i.e. we are looking at the group $(\mathcal{O}/N_1\mathcal{O})^*$ which is isomorphic to $GL_2(\mathbb{Z}/N_1\mathbb{Z})$. Furthermore, we are only concerned with the action of the endomorphisms on $\mathcal{I}$, i.e. on cyclic subgroups of $E[N_1]$ of order N_1, and we can therefore identify even more endomorphisms with each other by the following lemma.

Lemma 3.17. *Let (a, b, c, d) and (a', b', c', d') be the coefficients of θ and θ' in $(\mathcal{O}/N_1\mathcal{O})^*$ with respect to some $\mathbb{Z}$-basis of the endomorphism ring $\mathcal{O}$ of E, and let $\mathcal{I}$ be the set of cyclic N_1-order subgroups of $E[N_1]$. Then $\theta(K) = \theta'(K)$ for every $K \in \mathcal{I}$ if and only if there exists some $\lambda \in (\mathbb{Z}/N_1\mathbb{Z})^*$ such that*

$$(a, b, c, d) \equiv \lambda(a', b', c', d') \pmod{N_1}.$$

Proof. Considering the respective restrictions to $E[N_1]$, two endomorphisms are equal if they lie in the same class in $(\mathcal{O}/N_1\mathcal{O})^*$. Moreover, let θ_1, θ_2 be two endomorphisms of E such that $\theta_1 = [\lambda]\theta_2$ for some integer λ, and let P be an element of order N_1. Since scalar multiplication commutes with any endomorphism, it is easy to see that $\theta_1(P)$ and $\theta_2(P)$ generate the same subgroup in $E[N_1]$ if and only if λ is coprime to N_1. □

[4]Recall that the keyspace considered for SIDH is restricted slightly from the set of *all* order-N_1 subgroups of $E[N_1]$ in favour of a more straightforward secret generation algorithm; see Section 5.4.

Now, we are ready to give a solution to Task 3.10 if $K \in \mathcal{I}_1$.

Proposition 3.18. *Let G be the group of equivalence classes of elements*

$$\{a + b\iota \mid a \text{ odd}, b \text{ even}\} \subset (\mathbb{Z}[\iota]/N_1\mathbb{Z}[\iota])^* \subset (\mathcal{O}/N_1\mathcal{O})^*,$$

where we identify two elements if and only if they differ by multiplication by an odd scalar modulo N_1. Then G is an abelian group and it acts freely and transitively on $\mathcal{I}_1$.

Proof. It is easy to see that the endomorphisms in $\mathbb{Z}[\iota]$ of degree coprime to N_1 form an abelian subgroup of $\mathcal{O}$. Using any basis for $E[N_1]$ of the form $\{P, \iota(P)\}$, we can write the elements of this subgroup as matrices of the form $\left(\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right)$, where a is odd and b is even. By identifying two endomorphisms $a_1 + b_1\iota$ and $a_2 + b_2\iota$ if there exists an integer λ coprime to N_1 and an endomorphism δ such that $a_1 - \lambda a_2 + (b_1 - \lambda b_2) = N_1\delta$, which is possible by Lemma 3.17, we obtain G. As G is closed under multiplication and reduction modulo N_1, it is a subgroup of an abelian group and therefore abelian itself. Note that G contains all equivalence classes under Lemma 3.17 of endomorphisms of the form $a + b\iota$ for even b, independently of the chosen basis.

To examine the orbit of an element in $\mathcal{I}$, which is a cyclic N_1-order subgroup of $E[N_1]$, under the action of G, it is sufficient to look at the orbit of a generator of this cyclic group in $\mathcal{I}$. We consider the orbit of P which has coordinates $(1\ 0)$ with respect to our basis under the group action of G. The image of $(1\ 0)$ under an element $\left(\begin{smallmatrix} 1 & b \\ -b & 1 \end{smallmatrix}\right)$ is $(1\ b)$. Inspecting the cyclic subgroups of E these points generate, we get $G \cdot \langle P \rangle = \mathcal{I}_1$. □

Free and transitive group action on $\mathcal{I}_2$ and $\mathcal{I}_3$:

So far we have defined a free and transitive group action on $\mathcal{I}_1$ and thus for the curves in $f(\mathcal{I}_1)$. However, when the secret kernel is generated by $P + [\alpha]\iota(P)$ with α odd, the curve

$E/\langle P + [\alpha]\iota(P)\rangle$ is not contained in $f(\mathcal{I}_1)$. This is the case we handle next.

One can show that the action of the previously defined group G acting on curves at distance N_1 from E considered via f has three orbits resulting from secret SIDH keys (see the next section for details). We have already seen that $f(\mathcal{I}_1)$ is one orbit, but the odd-α cases will split into two orbits. Clearly, G cannot be free and transitive on both orbits, since the size of the orbits is smaller than the cardinality of the group. We avoid this issue by choosing a different but related group of cardinality $N_1/4$ acting on the curves corresponding to an odd α.

Lemma 3.19. *Again, let $Q := P + \iota(P)$ and define*

$$\mathcal{I}_2 = \left\{ \langle Q + [\alpha]\iota(Q)\rangle \mid \alpha \text{ even and } \alpha \in \left[0, \frac{N_1}{2} - 1\right] \right\}$$
$$\mathcal{I}_3 = \left\{ \langle Q + [\alpha]\iota(Q)\rangle \mid \alpha \text{ even and } \alpha \in \left[\frac{N_1}{2}, N_1 - 1\right] \right\}$$

as before. The restrictions $f_{|\mathcal{I}_2}$ and $f_{|\mathcal{I}_3}$ of f to $\mathcal{I}_2$ and $\mathcal{I}_3$ are injective.

Proof. We show that two distinct isogenies with kernel both in $\mathcal{I}_2$ (or both in $\mathcal{I}_3$) map to two non-isomorphic curves. Let α, α' be such that $\langle Q + [\alpha]\iota(Q)\rangle$ and $\langle Q + [\alpha']\iota(Q)\rangle$ are both in $\mathcal{I}_2$ (or $\mathcal{I}_3$). Suppose there exists an odd λ such that

$$Q + [\alpha]\iota(Q) = \lambda(Q + [\alpha']\iota(Q)).$$

This means $1 - \lambda \equiv 0 \pmod{N_1/2}$ and $\alpha - \lambda\alpha' \equiv 0 \pmod{N_1/2}$ which implies $\alpha \equiv \alpha' \pmod{N_1/2}$. It remains to show that $Q + [\alpha]\iota(Q)$ is never an odd multiple of $[-\alpha]Q + \iota(Q)$. Suppose there exists an odd λ such that

$$Q + [\alpha]\iota(Q) = \lambda([-\alpha']Q + \iota(Q)).$$

This implies $1 + \alpha'\lambda \equiv \alpha - \lambda \equiv 0 \pmod{N_1/2}$, which is a contradiction since $\alpha - \lambda \equiv 0 \pmod{N_1/2}$ implies that λ is even while $1 + \alpha'\lambda \equiv 0 \pmod{N_1/2}$ implies that λ is odd. Therefore, the curves $E/\langle Q + [\alpha]\iota(Q)\rangle$ and $E/\langle Q + [\alpha']\iota(Q)\rangle$ are pairwise non-isomorphic. □

Finally, we give a free and transitive group action on $\mathcal{I}_2$ and $\mathcal{I}_3$. We start by defining the acting group. Recall our definition of H_0 as

$$H_0 = \{a + b\iota \mid a \text{ odd}, b \text{ even } \}/(N_1/2)\mathcal{O}^*.$$

Again, we identify two endomorphisms $a+b\iota$ and $a'+b'\iota$ if there exists an odd $\lambda \in \mathbb{Z}/(N_1/2)\mathbb{Z}$ such that $a \equiv \lambda a' \pmod{N_1/2}$ and $b \equiv \lambda b' \pmod{N_1/2}$ to obtain the subgroup $H \subset H_0$.

Proposition 3.20. *H acts freely and transitively on $\mathcal{I}_2$ and $\mathcal{I}_3$.*

Proof. It is enough to show that H acts transitively on $\mathcal{I}_2$ and $\mathcal{I}_3$ because H, $\mathcal{I}_2$ and $\mathcal{I}_3$ have the same cardinality. We show that the orbit $H \cdot \langle Q\rangle$ contains every element in $\mathcal{I}_2$. This follows immediately from $(1 + \alpha\iota)Q = Q + [\alpha]\iota(Q)$. Similarly, H acts transitively on $\mathcal{I}_3$ as

$$(1 + \alpha\iota)(Q + N_1\iota(Q)/2) = (1 - \alpha N_1/2)Q + (\alpha + N_1/2)\iota(Q) = Q + (\alpha + N_1/2)\iota(Q),$$

where $(\alpha N_1/2)Q = 0$ as α is even. □

What remains to be shown is that every curve $E/\langle P + [\alpha]\iota(P)\rangle$ with odd α has a j-invariant contained in $f(\mathcal{I}_2)$ or $f(\mathcal{I}_3)$.

Proposition 3.21. *Let α be an odd integer. Then $f(\langle P + [\alpha]\iota(P)\rangle)$ is contained in $f(\mathcal{I}_2)$ or $f(\mathcal{I}_3)$.*

Proof. Observe that

$$P + [\alpha]\iota(P) = \frac{1+\alpha}{2}(P + \iota(P)) + \frac{\alpha-1}{2}(-P + \iota(P)) = \frac{1+\alpha}{2}Q + \frac{\alpha-1}{2}\iota(Q).$$

The sum of $\frac{1+\alpha}{2}$ and $\frac{\alpha-1}{2}$ is odd and therefore one of the fractions is even while the other one is odd. If $\frac{\alpha-1}{2}$ is even, then it is clear that the curve is contained in $f(\mathcal{I}_2)$ or $f(\mathcal{I}_3)$. In the case where $\frac{1+\alpha}{2}$ is even, $E/\langle\frac{1+\alpha}{2}Q + \frac{\alpha-1}{2}\iota(Q)\rangle$ is isomorphic to $E/\langle\frac{1-\alpha}{2}Q + \frac{\alpha+1}{2}\iota(Q)\rangle$ (because their kernels are related by ι) and thus the curve is contained in $f(\mathcal{I}_2)$ or $f(\mathcal{I}_3)$. □

In this subsection, we have identified three subsets of $\mathcal{I}$, restricted to which f is injective. Moreover, we have seen that the union $\cup_{i=1}^{3} f(\mathcal{I}_i)$ contains the j-invariants of all curves at distance N_1 from E which can be public curves in an SIDH instance. Finally, we gave an abelian subgroup of $(\mathcal{O}/N_1\mathcal{O})^*$ for each of these subsets of $\mathcal{I}$ that acts freely and transitively on it. Thus, we solve Task 3.10 as long as one determines or guesses which of the three $f(\mathcal{I}_i)$ contains $j(E/K)$.

The orbits of the group action

Recall that we previously defined a group acting on the set $\mathcal{I}_1$ which differs from the group acting on the sets $\mathcal{I}_2$ and $\mathcal{I}_3$. The reason for having multiple group actions is that we require them to be free and transitive. Let G be the group defined at the beginning of Section 3.3.2 and recall that N_1 is a power of 2. Clearly, G acts on all the kernels generated by points of the form $P + [\alpha]\iota(P)$. Let us study the orbits of this group action in more detail. As we have already seen in Proposition 3.18, the kernels where α is even form a single orbit. Now we show that there are two more orbits occurring when α is odd. For simplicity we will refer to a kernel generated by $P + [\alpha]\iota(P)$ by $(1\ \alpha)$.

Lemma 3.22. *Let α be odd. Then $(1\ \alpha)$ is either in the orbit of $(1\ 1)$ or $(1\ 3)$.*

Proof. First we suppose that $\alpha \equiv 1 \pmod 4$ and show that $(1\ \alpha)$ is in the orbit of $(1\ 1)$ in this case. For this, we must prove the existence of an odd λ and an even b such that the following system is satisfied: $\lambda(1+b) = 1$ and $\lambda(1-b) = \alpha$. Solving the system, we find that $\lambda = \frac{1+\alpha}{2}$ and $b = \frac{1-\lambda}{\lambda}$. These satisfy the required criteria since $1 + \alpha \equiv 2 \pmod 4$, hence λ is odd and $1 - \lambda$ is even.

Now suppose that $\alpha \equiv 3 \pmod 4$. In this case, we show that $(1\ \alpha)$ is in the orbit of $(1\ 3)$. Again there must exist an odd λ and an even b such that both $\lambda(1 + 3b) = 1$ and $\lambda(3 - b) = \alpha$. This implies that $\lambda = \frac{1+3\alpha}{10}$, which is an odd integer because α is congruent to 3 $\pmod 4$ and so $1 + 3\alpha$ is congruent to 2 $\pmod 4$. Now one can calculate that b equals $\frac{1-\lambda}{3\lambda}$ which is even since λ is odd, proving the second case. □

By Lemma 3.22 the group action defined above has three orbits on the subset of SIDH public curves of $\mathcal{I}$. However, the action of all of G is no longer free on the (smaller) orbits corresponding to an odd α, hence we consider H.

Generalising the group action to $N_1 = 3^k$

In this section we sketch a generalisation of the previous results in this section to the case where N_1 is a power of 3. This amounts to attacking Bob instead of Alice in an overstretched SIDH instance.

Lemma 3.14 carries over to this case as 9^k can only be written as a sum of two squares in a trivial fashion. Let P be a point such that $\{P, \iota(P)\}$ is a basis of $E[N_1]$. We show that every curve at distance N_1 from E corresponding to an SIDH secret key can be reached by an isogeny with a kernel of the form $\langle P + [\alpha]\iota(P)\rangle$. Let $Q = [\beta_1]P + [\beta_2]\iota(P)$ be a point of order N_1. If β_1 is coprime to 3, then we may multiply Q by an appropriate scalar such that the coordinate of P becomes 1. Suppose that β_1 is divisible by 3. Since Q has order N_1, β_2 is not divisible by 3. Observe that the points Q and $\iota(Q)$ generate kernels leading

to isomorphic curves which implies that $Q = [\beta_1]P + [\beta_2]\iota(P)$ and $\iota(Q) = -[\beta_2]P + [\beta_1]\iota(P)$ correspond to isogenies leading to isomorphic curves. Multiplying $\iota(Q)$ with an appropriate scalar, we obtain a kernel generator of the form $P + [\alpha]\iota(P)$.

However, some curves of the form $E/\langle P + [\alpha]\iota(P)\rangle$ may be pairwise isomorphic. Namely let α be coprime to 3. Then the kernels generated by $P+[\alpha]\iota(P)$ and $P+[-\alpha^{-1}]\iota(P)$ correspond to isomorphic curves. On the other hand, it is easy to see that α and $-\alpha^{-1}$ are not congruent modulo 3. In particular, all curves at distance N_1 from E occurring in SIDH can be reached by isogenies with kernels of the form $P + [\alpha]\iota(P)$ where α is congruent to 0 or 1 (mod 3). With a calculation similar to the one for N_1 being a power of 2, it can be shown that these curves are pairwise non-isomorphic.

The acting group can also be defined in a similar fashion, namely as the endomorphisms of the form $a + b\iota$ where b is divisible by 3 and two endomorphisms are identified whenever they are the same modulo N_1 up to multiplication by a scalar coprime to N_1. Again, for simplicity we denote a point $P + [\alpha]\iota(P)$ as $(1\ a)$. Similarly to our previous discussion of the orbits of the group action, one can check that this action has two orbits:

1. The orbit of $(1\ 0)$ consisting of points of the form $(1\ x)$, where 3 divides x.

2. The orbit of $(1\ 1)$ consisting of points of the form $(1\ x)$, where $x \equiv 1 \pmod 3$.

The orbit of $(1\ 2)$ contains points of the form $(1\ x)$ where x is congruent to 2 (mod 3), but in terms of j-invariants it consists of exactly the same curves as the second orbit. Since all these orbits have the same cardinality as the acting group, the group action is free and transitive, as required.

3.3.3 Using the Frobenius map

In Section 3.3.2, we described how to choose suitable abelian subgroups of $(\mathcal{O}/N_1\mathcal{O})^*$ in order to solve Task 3.10 after guessing whether $j(E/K)$ is a j-invariant in $f(\mathcal{I}_1)$, $f(\mathcal{I}_2)$, or $f(\mathcal{I}_3)$.

The elements of the acting groups chosen as described in the previous section can be trivially lifted to $\mathbb{Z}[\iota] := \mathbb{Q}[\iota] \cap \mathcal{O}$. We will later show as an alternative in Section 3.3.4 how these representatives can be lifted directly to elements of norm N_2 or eN_2, where e is a small positive integer, whenever the SIDH parameters N_1 and N_2 are sufficiently overstretched and unbalanced, satisfying $N_2 > p^2 N_1^4$. For these parameters, this solves a variation of Task 3.11.

First however, we reduce the required unbalancedness partially by proving that we can lift elements from $\pi\mathbb{Z}[\iota]$ instead. Assuming that $N_2 > pN_1^4$, we will then show in Section 3.3.4 how an endomorphisms from $\pi\mathbb{Z}[\iota]$ can be lifted efficiently to another endomorphism of norm N_2 or eN_2, for some small integer e, inducing the same action on $\mathcal{I}$. Note that it is not possible to choose a group generated by an element in $\pi\mathbb{Z}[\iota]$ to solve Task 3.10 directly, acting freely and transitively on a large number of N_1-isogeny kernels, as such an element has multiplicative order at most 4.

As before, let $\varphi : E \to E/K$ denote the secret N_1-isogeny we want to compute. Recall that to run our attack we need to be able to compute $E/\theta(K)$ for every θ in the groups G acting on $\mathcal{I}_1$, and H acting on $\mathcal{I}_2$ and $\mathcal{I}_2$. We have seen that we can represent θ as an element in $\mathbb{Z}[\iota]$.

Let π denote the Frobenius map. Assuming that we can lift $\pi\theta$ to an endomorphism of degree N_2 inducing the same action on $\mathcal{I}$, we can compute $E/\pi\theta(K)$ using knowledge of $\varphi(E[N_2])$ as described in Section 3.3.1. Now let $B := \theta(K)$. Given $E/\pi(B)$, we can compute E/B using the Frobenius map as follows.

Lemma 3.23. *Let E be an elliptic curve defined over $\mathbb{F}_p$, π the Frobenius map and let $B \subset E$ be a cyclic subgroup. $E/\pi(B)$ is isomorphic to the image of the Frobenius map of E/B.*

Proof. Let φ_1 be the isogeny with kernel B and φ_2 the isogeny with kernel $\pi(B)$. The isogeny φ_1 is separable and its kernel is contained in the kernel of $\varphi_2 \circ \pi$. Then, there exists a unique isogeny $\psi : E/B \to E/\pi(B)$ satisfying $\varphi_2 \circ \pi = \psi \circ \varphi_1$ (see [Sil09, Corollary III.4.11.]), i.e. the following diagram commutes.

$$\begin{array}{ccc} E & \xrightarrow{\varphi_1} & E/B \\ {\scriptstyle\pi}\downarrow & & \downarrow{\scriptstyle\psi} \\ E & \xrightarrow[\varphi_2]{} & E/\pi(B) \end{array}$$

The degree of a composition of isogenies is the product of its factors which implies $\deg \psi = p$. Furthermore, ψ is not separable as the Frobenius map is not. As ψ can be decomposed as a composition of the Frobenius map and a separable isogeny (see [Sil09, Corollary II.2.12.]), $\deg \psi = p$ implies that ψ must be a composition of Frobenius and an automorphism. Hence, E/B and $E/\pi(B)$ are linked by the Frobenius map. □

Lemma 3.23 implies that we can compute $E/\theta(K)$ by first computing $E/\pi\theta(K)$ and then applying the Frobenius map. This gives rise to the following strategy when constructing the malleability oracle: Assume we want to compute $E/\theta(K)$ for some $\theta \in \mathbb{Z}[\iota]$ and unknown K, given the image of the N_2-torsion of the isogeny $\varphi : E \to E/K$. Using the lifting algorithm of the next section, we compute an endomorphism θ' of degree N_2 or eN_2 for a small e that induces the same action on $\mathcal{I}$ as $\pi\theta$. As described previously, the torsion point information allows us to compute $E/\theta'(K) = E/\pi\theta(K)$. By Lemma 3.23, applying the Frobenius map yields $E/\pi\theta'(K) = E/\theta(K)$.

3.3.4 Lifting $\theta \in \pi\mathbb{Z}[\iota]$ to an element of norm eN_2

In this section we give an efficient algorithm to lift endomorphisms from $\pi\mathbb{Z}[\iota] = \pi(\mathbb{Q}[\iota] \cap \mathrm{End}(E))$ to another endomorphism of $E/\mathbb{F}_p$ of degree N_2 or eN_2 that induces the same action on $\mathcal{I}$, whenever $N_2 > pN_1^4$. Here, e is the smallest positive integer such that $eN_2/p(c_0^2 + d_0^2)$ is a quadratic residue modulo $2N_1$, where $\pi(c_0 + d_0\iota) \in \pi\mathbb{Z}[\iota]$ is the endomorphism we want to lift.

This will solve the following task, which is a variant of Task 3.11, efficiently.

Task 3.24. *Let N_1, N_2 be coprime integers such that $N_2 > pN_1^4$, let $\theta := \pi(c_0 + d_0\iota) \in \pi\mathbb{Z}[\iota]$ be an E-endomorphism of degree coprime to N_1 and let e denote the smallest positive integer such that $eN_2/p(c_0^2 + d_0^2) \pmod{2N_1}$ is a quadratic residue. Compute an endomorphism θ' of degree N_2 or eN_2 such that $\theta(K) = \theta'(K)$ for all $K \in \mathcal{I}$.*

We have discussed in Section 3.3.3 that we can lift $\pi(c_0 + d_0\iota)$ instead of $c_0 + d_0\iota$. Therefore, this task solves Task 3.11 up to the following two relaxations. First, we require N_2 to be sufficiently large and unbalanced compared to N_1. Second, we allow θ' to be either of degree N_2 or eN_2 for some small positive integer e.

Remark 3.25. *If N_1 were a prime, e could be chosen as the smallest quadratic non-residue modulo N_1. However, in our case N_1 is a composite number. Thus, the product of two quadratic non-residues might not be a quadratic residue if there are multiple cosets of the subgroup of quadratic residues in the group of units modulo $2N_1$.*

We are primarily interested in the case where N_1 is a prime power $\ell^n \in \{\ell_A^{e_A}, \ell_B^{e_B}\}$. By Hensel's lemma, being a quadratic residue modulo ℓ^n is equivalent to being a quadratic residue modulo ℓ, if ℓ is odd, and equivalent to being a quadratic residue modulo 8, if $\ell = 2$. Consequently, there is one coset of the quadratic residues in the group of units of $2N_1$ if ℓ is an odd prime. Therefore, e can be chosen to be the smallest quadratic non-residue modulo ℓ.

For example, if N_1 is a power of 3 one can choose $e = 2$.

If $\ell = 2$, then there are three cosets of the quadratic residues in the group of units, i.e. the ones that contain $3, 5$ and 7 respectively. Consequently, e can always be chosen to be one of $3, 5$ or 7 in this case.

In case N_1 has distinct prime factors, for $eN_2/p(c_0^2 + d_0^2)$ to be a quadratic residue it has to be a quadratic residue modulo the largest prime power dividing $2N_1$ for each distinct prime factor. If the number of cosets grows, so do the possibilities for e und thus the size of the smallest e that is guaranteed to work.

We now describe an algorithm to solve Task 3.24. By Lemma 3.17, it suffices to solve the following task which is similar to the problem solved at the core of the KLPT algorithm [KLPT14].

Task 3.26. *Given $\theta = a_0+b_0\iota+(c_0+d_0\iota)\pi$, find $\theta' = a_1+b_1\iota+(c_1+d_1\iota)\pi$ of degree N_2 or eN_2 with coefficients $(a_1, b_1, c_1, d_1) \equiv \lambda(a_0, b_0, c_0, d_0) \pmod{N_1}$ for some scalar $\lambda \in (\mathbb{Z}/N_1\mathbb{Z})^*$.*

In the following, we provide a solution to this task. Let

$$\theta' = \lambda a_0 + N_1 a_1 + \iota(\lambda b_0 + N_1 b_1) + (\lambda c_0 + N_1 c_1 + \iota(\lambda d_0 + N_1 d_1))\pi.$$

From $\mathrm{Norm}(x + y\iota) = x^2 + y^2$, we can compute

$$\mathrm{Norm}(\theta') = (\lambda a_0 + N_1 a_1)^2 + (\lambda b_0 + N_1 b_1)^2 + p\big((\lambda c_0 + N_1 c_1)^2 + (\lambda d_0 + N_1 d_1)^2\big). \tag{3.2}$$

Since $\theta \in \pi\mathbb{Z}[\iota]$ implies $a_0 = b_0 = 0$, Equation (3.2) simplifies to

$$\mathrm{Norm}(\theta') = N_1^2(a_1^2 + b_1^2) + p\big((\lambda c_0 + N_1 c_1)^2 + (\lambda d_0 + N_1 d_1)^2\big). \tag{3.3}$$

Set e to be the smallest positive integer such that $eN_2/(p(c_0^2+d_0^2))$ is a quadratic residue modulo $2N_1$.

The goal is to compute θ' such that $\mathrm{Norm}(\theta') = eN_2$. Considering Equation (3.3) modulo N_1, we obtain

$$eN_2 \equiv \lambda^2 p(c_0^2+d_0^2) \pmod{N_1}. \tag{3.4}$$

Since $eN_2/p(c_0^2+d_0^2)$ is a quadratic residue modulo $2N_1$ by the choice of e, there exists a solution for λ in Equation (3.4) modulo $2N_1$. Compute any such solution, and lift it to the integers in $[1, 2N_1-1]$. Note that we do not lose generality by the lift as any other lift of λ corresponds to a change in c_1 and d_1 instead.

For fixed c_0, d_0 and λ, this gives an affine relation between c_1 and d_1 modulo N_1, i.e.

$$c_0c_1 + d_0d_1 \equiv \frac{\mathrm{Norm}(\theta') - \lambda^2 p(c_0^2+d_0^2)}{2\lambda p N_1} \pmod{N_1}. \tag{3.5}$$

Finally, one is left with the problem of representing an integer r as the sum of two squares, namely to find a solution (a_1, b_1) for

$$a_1^2 + b_1^2 = r := \frac{\mathrm{Norm}(\theta') - p\big((\lambda c_0 + N_1 c_1)^2 + (\lambda d_0 + N_1 d_1)^2\big)}{N_1^2} \tag{3.6}$$

where λ, c_0 and d_0 are fixed, and c_1, d_1 satisfy an affine equation modulo N_1.

As Petit and Smith pointed out in [PS18], the solution space to Equation (3.5) is a translated lattice modulo N_1. More precisely, we know that c_0 or d_0 is coprime to N_1. Without loss of generality, let d_0 be coprime to N_1. Furthermore, let C denote the right hand side of Equation (3.5). Then, (c_1, d_1) lies in the lattice

$$\langle (c_0/d_0, -1), (N_1, 0) \rangle + (C/d_0, 0). \tag{3.7}$$

Clearly, r from Equation (3.6) can only be represented as a sum of two squares if it is positive. This happens when the parameters N_1 and N_2 are sufficiently overstretched and unbalanced. To find a solution, one computes close vectors (c_1, d_1) to the target vector $(-\lambda c_0/N_1, -\lambda d_0/N_1)$ in the translated lattice.

Given the factorisation of r as defined in Equation (3.6), Cornacchia's algorithm [Cor08] can then efficiently solve for a_1, b_1 or determine that no such solution exists. If no solution exists, a different vector (c_1, d_1) is chosen.

Remark 3.27. *Cornacchia's algorithm requires the factorisation of r. This can be done in classical subexponential time or in quantum polynomial time. To avoid such computations, we apply Cornacchia's algorithm only when r is a prime and otherwise sample another close vector from the lattice.*

Assuming the values of r behave like random values around pN_1^3 for the close vectors, one expects to choose $\log(pN_1^3)$ different vectors (c_1, d_1) before finding a solution for a_1, b_1 with Cornacchia's algorithm. If we do not apply Cornacchia's algorithm unless r is prime, we expect furthermore to sample roughly $\log(pN_1^3)$ values for (c_1, d_1) until r is prime.

The volume of the translated lattice is N_1. Thus, for a generic lattice for which the Gaussian heuristic holds we expect to find a lattice point at distance N_1 from $(\lambda c_0/N_1, \lambda d_0/N_1)$. Furthermore, we can use the Hermite constant for 2-dimensional lattices to trivially bound the distance between this lattice point and the next $2\log(pN_1^3)$ closest lattice points by $\frac{8}{3}\log(pN_1^3)\sqrt{N_1}$. Thus, heuristically r is positive for the expected number of vectors (c_1, d_1) that we need to sample, whenever $eN_2 > pN_1^3 + 8/3\log(pN_1^3)\sqrt{N_1^3}$.

Remark 3.28. *Note that for specific lattices, the Gaussian heuristic might be violated. In the worst case, we can only expect to find a lattice point at distance N_1^2 from $(\lambda c_0/N_1, \lambda d_0/N_1)$ and overall solutions require roughly $eN_2 > pN_1^4$.*

It is easy to see that a solution for (a_1, b_1, c_1, d_1) as computed with the routine described above satisfies Equation (3.9). The full algorithm is summarised in Algorithm 3.

Algorithm 3: Lift element from $\pi\mathbb{Z}[\iota]$ to quaternion of norm N_2 or eN_2

Input: $\theta = \pi(c_0 + d_0\iota) \in \mathrm{End}(E)$, and parameters p, ε, N_1, N_2
Output: $\theta' = N_1a_1 + N_1b_1\iota + (\lambda c_0 + N_1c_1)\pi + (\lambda d_0 + N_1d_1)\iota\pi$ satisfying $\mathrm{Norm}(\theta') = N_2$ or eN_2 with probability $1 - \varepsilon$, and $\perp$ otherwise

1 $e \leftarrow$ least positive integer such that $eN_2/p(c_0^2 + d_0^2) \pmod{2N_1}$ is a quadratic residue;
2 Compute λ in $eN_2 \equiv \lambda^2 p(c_0^2 + d_0^2) \pmod{2N_1}$;
3 Compute affine relation $c_0c_1 + d_0d_1 \equiv C \pmod{N_1}$;
4 Define translated lattice L containing all (c_1, d_1) satisfying the affine relation;
5 $B \leftarrow \log(\varepsilon)\log(pN_1^3)/\log(1 - \log^{-1}(pN_1^3))$;
6 **for** $m = 1, \ldots, B$ **do**
7 Compute next closest vector (c_1, d_1) to $(-\lambda c_0/N_1, -\lambda d_0/N_1)$ in L;
8 $r \leftarrow \frac{\mathrm{Norm}(\theta') - p((\lambda c_0 + N_1c_1)^2 + (\lambda d_0 + N_1d_1)^2)}{N_1^2}$;
9 **if** r prime **then**
10 Use Cornacchia's algorithm to find a_1, b_1 such that $a_1^2 + b_1^2 = r$ or determine that no solution exists;
11 **if** solution found **then**
12 **return** $\theta' = N_1a_1 + N_1b_1\iota + (\lambda c_0 + N_1c_1)\pi + (\lambda d_0 + N_1d_1)\iota\pi$;
13 **return** $\perp$

An examination of Algorithm 3 shows that it aborts after a fixed number of trials for pairs (c_1, d_1), which leads to the following result.

Lemma 3.29. *Algorithm 3 always terminates and is correct if it returns a solution.*

We conclude this section by investigating the heuristic probability of the lifting algorithm returning a solution or aborting unsuccessfully, as well as its complexity.

Lemma 3.30. *Let $0 < \varepsilon < 1$. Assume r in Line 8 of Algorithm 3 behaves like a random value around pN_1^3. Then we expect Algorithm 3 heuristically to return a correct lift with probability $1 - \varepsilon$ and an error $\perp$ otherwise.*

Proof. If r in Line 8 of Algorithm 3 behaves like a random value around pN_1^3, we expect it

to be prime with probability roughly $1/\log(pN_1^3)$ and Cornacchia's algorithm to provide a solution with probability approximately $1/(\log(pN_1^3))$ due to Landau [Lan08] and Ramanujan [Ram13]. Iterating over B short vectors (c_1, d_1) of the lattice as defined in Step 6 of Algorithm 3, we therefore expect our algorithm to return $\perp$ with probability

$$\left(1 - \frac{1}{\log(pN_1^3)}\right)^{B/\log(pN_1^3)}.$$

Hence, iterating over $B \geq \log(\varepsilon)\log(pN_1^3)/\log(1-\log^{-1}(pN_1^3))$ as in Algorithm 3, we fail to find a solution with probability less than ε heuristically. □

Remark 3.31. *In Algorithm 2 the lifting of endomorphisms is used for every element of the acting group G or H with cardinality $N_1/2$ and $N_1/4$, respectively. Since we expect the lifting algorithm to fail heuristically with probability ε for every single group element and the functions in Algorithm 2 are only exact shifts of each other when it does not fail a single time, we need to choose ε sufficiently small. Assuming independence between the different executions of the lifting algorithm, we expect to find two functions satisfying the promise of a hidden shift with probability $(1-\varepsilon)^{N_1/2} \approx 1 - \varepsilon N_1/2$ by first order Taylor approximation. Thus, choosing $\varepsilon < \frac{1}{N_1}$ we expect our lifting to work with probability roughly $\frac{1}{2}$ on all endomorphisms of G and similarly $\varepsilon < \frac{2}{N_1}$ for the elements in H. By the previous lemma, the lifting remains polynomial in $\log(N_1)$ and $\log(p)$ for any such ε. Choosing ε smaller allows us to heuristically achieve a larger success probability of the algorithm. The worst-case complexity of the lifting increases linearly in $|\log(\varepsilon)|$.*

Lemma 3.32. *Let $0 < \varepsilon < 1$. Algorithm 3 runs in time polynomial in $\log p$, $\log N_1$, and $|\log(\varepsilon)|$.*

Proof. The worst-case runtime of the algorithm stems from sampling B (as defined in Algorithm 3, Line 5) potential values of (c_1, d_1) from a lattice of dimension 2. In each iteration one

needs to run a primality test, and apply Cornacchia's algorithm to a prime of size polynomial in p and N_1. □

The main drawback of our lifting algorithm is the requirement of approximately $N_2 > pN_1^3$ in case the Gaussian heuristic is satisfied for the lattice defined in Equation (3.7), and roughly $N_2 > pN_1^4$ otherwise (see Remark 3.28). This bound might be partially caused by inefficiencies in the lifting algorithm. However, the following remark discusses why we can a priori not expect to find a lifting algorithm for balanced parameters.

Remark 3.33. *A randomly chosen non-homogeneous quadratic equation in two variables has in general no solution. Similarly, for arbitrary endomorphisms and any N_1, N_2, we would not expect to find an endomorphism $a_1 + b_1\iota \in \mathbb{Z}[\iota]$ (in the variables a_1, b_1) inducing the same action on $\mathcal{I}$ of degree N_2. Yet, as soon as we lift an endomorphism θ to an endomorphism $\theta' = N_1(a_1 + b_1\iota + c_1\pi) + \lambda\theta$, the degree of the lift will be of degree larger than pN_1^2.*

Lifting $\theta \in \mathbb{Z}[\iota]$ to an element of norm N_2 or eN_2

Similar to our approach above, let e denote the smallest positive integer such that $eN_2/p(a_0^2 + b_0^2)$ is a quadratic residue modulo $2N_1$, where $a_0 + b_0\iota \in \mathbb{Z}[\iota]$ is the endomorphism we want to lift. Remark 3.25 regarding the size of e still applies in this case.

In this section we describe how to lift endomorphisms in $\mathbb{Z}[\iota] = \mathbb{Q}[\iota] \cap \mathrm{End}(E)$ directly to another endomorphism of E of degree N_2 or eN_2 which has the same action on $\mathcal{I}$. This gives an efficient solution to the following variant of Task 3.11.

Task 3.34. *Let N_1, N_2 be integers such that $N_2 > p^2N_1^4$. Given an endomorphism $\theta \in G$ of degree coprime to N_1 and an integer N_2 coprime to N_1, compute an endomorphism θ' of degree N_2 or eN_2 such that $\theta(K) = \theta'(K)$ for all $K \in I$.*

As before, this is a relaxation of Task 3.11 in two ways. First, we require N_2 to be

sufficiently large and unbalanced compared to N_1. Second, we allow θ' to be either of degree N_2 or eN_2 for some small integer e.

We now give the algorithm to solve Task 3.34. By Lemma 3.17 it suffices to solve Task 3.26 for endomorphisms in $\mathbb{Z}[\iota]$.

Let $\theta' = \lambda a_0 + N_1 a_1 + \iota(\lambda b_0 + N_1 b_1) + (\lambda c_0 + N_1 c_1 + \iota(\lambda d_0 + N_1 d_1))\pi$. Then its norm equals

$$\mathrm{Norm}(\theta') = (\lambda a_0 + N_1 a_1)^2 + (\lambda b_0 + N_1 b_1)^2 + p((\lambda c_0 + N_1 c_1)^2 + (\lambda d_0 + N_1 d_1)^2), \tag{3.8}$$

as $\mathrm{Norm}(x + y\iota) = x^2 + y^2$. Since $\theta \in \mathbb{Z}[\iota]$ implies $c_0 = d_0 = 0$, Equation (3.8) simplifies to

$$\mathrm{Norm}(\theta') = (\lambda a_0 + N_1 a_1)^2 + (\lambda b_0 + N_1 b_1)^2 + pN_1^2(c_1^2 + d_1^2). \tag{3.9}$$

We want to compute θ' such that $\mathrm{Norm}(\theta') = eN_2$. Considering Equation (3.9) modulo $2N_1$, we obtain

$$eN_2 \equiv \lambda^2(a_0^2 + b_0^2) \pmod{2N_1}. \tag{3.10}$$

The choice of e implies that there exists a solution for λ. Compute any such solution, and lift it to the integers in $[1, 2N_1 - 1]$. This is without loss of generality as any other lift of λ corresponds to a change in a_1, b_1 instead.

Considering the equation modulo N_1^2 yields an affine relation between a_1 and b_1 modulo N_1, i.e.

$$\lambda(a_0 a_1 + b_0 b_1) \equiv \frac{\mathrm{Norm}(\theta') - \lambda^2(a_0^2 + b_0^2)}{2N_1} \pmod{N_1}.$$

Take the affine relation between a_1 and b_1 modulo N_1, say $e_b b_1 = e_a a_1 + e_c + mN_1$ for some fixed integers e_a, e_b, e_c and a variable integer m. Assume $e_b \not\equiv 0 \pmod{p}$ as lifting

would be trivial otherwise, and substitute b_1 in Equation (3.9) modulo the prime p, i.e.

$$eN_2 \equiv (\lambda a_0 + N_1 a_1)^2 + (\lambda b_0 + N_1 e_b^{-1}(e_a a_1 + e_c + mN_1))^2 \pmod{p}.$$

Note that fixing any value for m leaves a quadratic equation in a_1 modulo p. Fix $m = 0$ and complete the square in the equation to solve it if there exists a solution. Otherwise, increase m by one and repeat. Heuristically, one expects this degree-2 polynomial modulo p to be split with probability $1/2$ and hence we expect to iterate twice before finding a solution.

Once a solution for a_1 is obtained modulo p, lift it to the integers. One is left with the problem of representing an integer as the norm of an element in $\mathbb{Z}[\iota]$, i.e. finding c_1 and d_1 such that

$$c_1^2 + d_1^2 = r := \frac{\mathrm{Norm}(\theta') - (\lambda a_0 + N_1 a_1)^2 + (\lambda b_0 + N_1 b_1)^2}{pN_1^2}$$

if they exist. Clearly, r can only be a norm if it is positive. This happens when the parameters N_1 and N_2 are overstretched, and more precisely if $\mathrm{Norm}(\theta') > p^2 N_1^4$.

Again, if the prime decomposition of r is known, Cornacchia's algorithm can efficiently answer the question whether r can be decomposed that way and compute a solution if one exists. Assuming the value of $(\lambda a_0 + N_1 a_1)^2 + (\lambda b_0 + N_1 b_1)^2$ behaves like a random value around $p^2 N_1^4$, one expects to choose $\log(p^2 N_1^4)$ different values for m with a solution to the quadratic equation modulo p before finding a solution with Cornacchia's algorithm. Like before, we avoid having to factor r by sampling values for m until r is prime, and then apply Cornacchia's strategy.

It is easy to see that a solution for (a_1, b_1, c_1, d_1) as computed with the routine described above satisfies Equation (3.3). The full algorithm is summarised in Algorithm 4.

We conclude this section by investigating the heuristic probability of the lifting algorithm returning a solution or aborting unsuccessfully, as well as its complexity.

Algorithm 4: Lift element from $\mathbb{Z}[\iota]$ to quaternion of norm N_2 or eN_2

Input: $\theta = a_0 + b_0\iota \in \mathrm{End}(E)$, and parameters p, ε, N_1, $N_2 > p^2N_1^4$
Output: $\theta' = \lambda a_0 + N_1a_1 + (\lambda b_0 + N_1b_1)\iota + N_1c_1\pi + N_1d_1\iota\pi$ satisfying $\mathrm{Norm}(\theta') = N_2$ or eN_2 with probability $1-\varepsilon$ and $\perp$ otherwise

1 $e \leftarrow$ least positive integer such that $eN_2/p(a_0^2+b_0^2) \pmod{2N_1}$ is quadratic residue;
2 Compute λ in $eN_2 \equiv \lambda^2(a_0^2+b_0^2) \pmod{2N_1}$;
3 Compute linear relation between a_1 and b_1 modulo N_1, say $e_bb_1 \equiv e_aa_1 + e_c \pmod{N_1}$ for some integers e_a, e_b, e_c, using

$$\lambda(a_0a_1 + b_0b_1) \equiv \frac{eN_2 - ((\lambda a_0)^2 + (\lambda b_0)^2)}{2N_1} \pmod{N_1};$$

4 $B \leftarrow 2\log(\varepsilon)\log(p^2N_1^4)/\log(1-\log^{-1}(p^2N_1^4))$;
5 **for** $m = 0, 1, \ldots, B$ **do**
6 Substitute b_1 using expression $e_bb_1 = e_aa_1 + e_c + mN_1$ in

$$eN_2 \equiv (\ \lambda a_0 + N_1a_1)^2 + (\lambda b_0 + N_1b_1\ )^2 \pmod{p};$$

7 **if** *solution for* $a_1 \pmod{p}$ *exists* **then**
8 Compute a_1 and b_1 modulo p and lift them to integers in $[-p/2, p/2]$;
9 $r \leftarrow \frac{eN_2-((\lambda a_0+N_1a_1)^2+(\lambda b_0+N_1b_1)^2)}{pN_1^2}$;
10 **if** r *is prime* **then**
11 Use Cornacchia's algorithm to decompose r as sum of two squares $c_1^2 + d_1^2$ or determine that no solution exists;
12 **if** *solution is found* **then**
13 **return** $\theta' = \lambda a_0 + N_1a_1 + (\lambda b_0 + N_1b_1)\iota + N_1c_1\pi + N_1d_1\iota\pi$;
14 **return** $\perp$

The success probability is based on the following heuristic assumptions:

1. The discriminant of $(\lambda a_0 + N_1a_1)^2 + (\lambda b_0 + N_1b_1)^2$ in Line 6 of Algorithm 4 is uniformly distributed modulo p.

2. r in Line 9 of Algorithm 4 behaves like a random value around $p^2N_1^4$.

Lemma 3.35. *Let $\varepsilon > 0$. Under the heuristic assumptions mentioned in the preceding paragraph, the algorithm returns a lift with probability $1-\varepsilon$ and an error $\perp$ otherwise.*

Proof. Based on the heuristic that the discriminant of $(\lambda a_0 + N_1 a_1)^2 + (\lambda b_0 + N_1 b_1)^2$ in Line 6 of Algorithm 4 is uniformly distributed modulo p, we expect to find a solution for $a_1 \pmod p$ for half of the chosen m. Moreover, if r (Line 9, Algorithm 4) behaves like a random value around $p^2 N_1^4$, we expect it to be prime with probability roughly $1/\log(p^2 N_1^4)$ and Cornacchia's algorithm to provide a solution with probability roughly $1/(\log(p^2 N_1^4))$ due to Landau [Lan08] and Ramanujan [Ram13]. Iterating over B values of m, we therefore expect our algorithm to return $\perp$ with probability

$$\left(1 - \frac{1}{\log(p^2 N_1^4)}\right)^{B/2(\log(p^2 N_1^4)}.$$

In particular, iterating over $B \geq 2\log(\varepsilon)\log(p^2 N_1^4)/\log(1 - \log^{-1}(p^2 N_1^4))$ as in Algorithm 4, we expect to fail to find a solution with probability less than ε heuristically. □

Finally, it is easy to observe the following result.

Lemma 3.36. *Algorithm 4 always terminates and it runs in time polynomial in* $\log p$, $\log N_1$ *and* $|\log(\varepsilon)|$ *for every* $\varepsilon > 0$.

Proof. For any $\varepsilon > 0$, the worst-case runtime of the algorithm stems from the iteration over up to $2\log(\varepsilon)\log(p^2 N_1^4)/\log(1 - \log^{-1}(p^2 N_1^4))$ values of m. In each iteration one needs to solve at most one quadratic equation modulo p, and apply Cornacchia's algorithm to a prime of size polynomial in p and N_1. □

The main drawback of this lifting algorithm is the requirement for the unbalancedness $N_2 > p^2 N_1^4$. In Section 3.3.3, we argued why we can lift endomorphisms from $\pi\mathbb{Z}[\iota]$ instead, which is possible with an unbalancedness of $N_2 > pN_1^4$ as described at the beginning of Section 3.3.4.

3.3.5 Algorithm summary

We begin the summary of our attack by proving that a solution to Task 3.11 allows us to construct a malleability oracle for f.

Proposition 3.37. *Let $f_{|\mathcal{I}'} : \mathcal{I}' \to \mathfrak{O}$ be the function defined in (3.1) restricted to a domain $\mathcal{I}'$ so it is injective, let G be an abelian subgroup of $(\mathcal{O}/N_1\mathcal{O})^*$ acting freely and transitively on $\mathcal{I}'$ and let $\varphi : E \to E/K$, where $K \in \mathcal{I}'$ is chosen uniformly at random and unknown. Suppose the public parameters allow us to solve Task 3.11 for endomorphisms in G efficiently. Given $\varphi|_{E[N_2]}$, we then have a polynomial-time malleability oracle for G at $f_{|\mathcal{I}'}(K)$.*

Proof. We need to show that there exists an efficient algorithm that, on input $f(K) = f_{|\mathcal{I}'}(K) = j(E/K)$ and $\theta \in G$, computes $f(\theta(K))$. Let φ be the isogeny corresponding to the cyclic subgroup $K \subset E$ of order N_1.

The endomorphism θ has degree N_2 coprime to N_1 and using the efficient solution to Task 3.11, we can compute some θ' of degree N_2 such that it has the same action on the N_1-torsion as θ. Therefore, $f(\theta(K)) = E/\theta(K) = E/\theta'(K)$ up to isomorphism. By Lemma 3.12, this equals $(E/K)/\varphi(\ker\theta')$. Since $\ker\theta'$ lies in $E[N_2]$, we can compute its image under φ and therefore we can calculate $f(\theta(K)) = (E/K)/\varphi(\ker\theta')$ efficiently. □

Proposition 3.37 calls for solutions to the Tasks 3.10 and 3.11. In Sections 3.3.2 and 3.3.4 we presented solutions to *variants* of these tasks. We use the remainder of this section to summarise the impact of these variations on the success of our approach.

Restricting the function $f : \mathcal{I} \to \mathfrak{O}$ to a subset $\mathcal{I}'$ such that $f_{|\mathcal{I}'}$ is injective and its image contains $j(E/K)$ for the K one desires to recover requires information on the secret we do not posses. However, we gave three subsets $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$ of $\mathcal{I}$ in Section 3.3.2 such that f restricted to any of these subsets is injective. The images of these sets under f partition

all curves at distance N_1 from E which can arise from SIDH instances up to isomorphism, i.e. one of the three subsets will yield the desired result. Moreover, we provided abelian subgroups of $\mathbb{Q}[\iota] \cap (\mathcal{O}/N_1\mathcal{O})^*$ acting freely and transitively on $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$.

We then supply an algorithm to solve Task 3.24, a variant of Task 3.11 when N_1 and N_2 are sufficiently unbalanced, lifting endomorphisms from $\pi\mathbb{Z}[\iota]$ to ones with the same action on $\mathcal{I}$ of degree N_2 or eN_2. Here, e is a small integer depending on the parameters p, N_1, N_2 and the endomorphism. As a consequence, to use the torsion point information of $E[eN_2]$ under the secret isogeny given the image of $E[N_2]$, we need to guess the action on $E[e]$. Furthermore, we lift all endomorphisms in the acting group and thus we need to guess the action on $E[T]$, where T is the least common multiple of all e appearing for the different lifts. In Remark 3.25 we discuss which e might appear depending on the factorisation of N_1. For example, T is 2 if N_1 is a power of 3, or $\mathrm{lcm}(3, 5, 7)$ if N_1 is a power of 2. Guessing the action of the secret isogeny on $E[T]$ takes $O(T^3)$ trials. Finally, for efficiency reasons we lift endomorphisms from $\pi\mathbb{Z}[\iota]$, whereas the elements in the abelian groups acting on $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$ have representatives in $\mathbb{Z}[\iota]$. In Section 3.3.3 we showed that this is no restriction via the computation of an action of the Frobenius map.

For each combination of guesses of $E[e]$ under the secret isogeny and whether f maps the secret K into $f(\mathcal{I}_1)$, $f(\mathcal{I}_2)$ or $f(\mathcal{I}_3)$, we can use a subexponential quantum algorithm such as Kuperberg's [Kup13] to compute the hidden shift for the functions F_K and F_J as defined in Algorithm 2 and verify the output of the algorithm. Both functions are injective and therefore the verification can be achieved by computing both functions on a single element and its shift respectively. Once the premise of a hidden shift is satisfied, Kuperberg's algorithm [Kup13] recovers the (correct) solution to the injective abelian hidden shift problem. Thus, we recover the secret isogeny as described in Section 3.3 in general. We can summarise our result as follows.

Theorem 3.38. *Let $N_2 > pN_1^4$. Under the heuristics used for the lifting of endomorphisms in Section 3.3.4, the SIDH problem can be solved in quantum subexponential time via a reduction to the injective abelian hidden shift problem.*

During this section, we have made some restrictions to simplify the presentation of our cryptanalysis. We assumed the starting curve E to be a supersingular curve with j-invariant 1728. However, the attack also applies to other curves with known endomorphism rings that are close to E. In Section 3.3.2, we described the required group action on $\mathcal{I}$ under the further assumption that N_1 is a power of 2, which can be generalised to powers of small primes. A sketch for powers of 3 can be found in Section 3.3.2. Finally, we assumed that $N_1^2 < \frac{p+1}{4}$ in Lemma 3.14. However, to run our attack we can slightly ease this restriction. Namely, if $N_1^2 > \frac{p+1}{4}$, then we choose a divisor N_1' of N_1 such that $N_1'^2 < \frac{p+1}{4}$ and run the attack with N_1' instead. This will reveal the N_1'-part of the isogeny and then we can guess the remaining part. For sufficiently small $\frac{N_1}{N_1'}$, this is only a minor inefficiency.

3.3.6 Hybrid attacks on overstretched SIDH

In this section, we examine to what extent partial knowledge of the secret, i.e. knowledge of the most or least significant bits, renders the attack more efficient. Moreover, we describe how the attack can be adapted to some further parameters that are not quite sufficiently unbalanced. The idea is to apply exhaustive search to recover parts of the secret isogeny until the remaining part of the isogeny is of such small degree that the attack outlined in this paper can be used to recover the remaining part.

We start with the case where the most significant bits of the secret are leaked or correctly guessed. These bits correspond to the last steps of the secret isogeny in the isogeny graph. Assume N_1 is a power of a prime ℓ. If the most significant k digits of the secret with respect to their representation in base ℓ are leaked or guessed correctly, the partial isogeny

which remains to be recovered is of degree N_1/ℓ^k and we can run our attack as soon as N_1/ℓ^k fulfills the unbalancedness criterion $N_2 > p(N_1/\ell^k)^4$.

The case where the least significant digits are known or guessed requires a little more work. For simplicity of our exposition we assume again that N_1 is a power of 2, but the results generalise to powers of small primes.

Lemma 3.39. *Let G be the group of Proposition 3.18, and let $G' \subset G$ be the subset of the form $\{a+b\iota \mid a$ odd, b divisible by $2^k\}$ where we identify two endomorphisms with each other if they differ by multiplication by an odd scalar modulo N_1. Then G' is an abelian subgroup of G.*

Proof. Since G is abelian, it suffices to show that G' is a subgroup. Consider $(a+b\iota)(a'+b'\iota) = (aa'-bb') + (ab'+a'b)\iota$. It is easy to see that $aa'-bb'$ is odd and $ab'+a'b$ is divisible by 2^k if $a+b\iota$ and $a'+b'\iota$ are in G'. □

Assume the least significant k bits of the secret, or equivalently the first k steps of the secret isogeny, are known. Kernels of isogenies of degree $N_1 > 2^k$ that share the same first k steps lie in the same 2^k-torsion subgroup and are therefore congruent modulo 2^k.

Recall the subsets of $\mathcal{I}$ introduced in Section 3.3.2.

Proposition 3.40. *Let $\mathcal{I}'$ be any subsets of $\mathcal{I}_1 := \{\langle P + [\alpha]Q\rangle$ with $2|\alpha\}$ containing all those cyclic subgroups where the α are congruent modulo 2^k. The group G' of Lemma 3.39 acts freely and transitively on any $\mathcal{I}'$.*

Proof. First, we need to show that $G' \times \mathcal{I}' \to \mathcal{I}'$ is well-defined. Let $(1+b\iota)$ be a representative of some element in G' and let $P + k\iota(P)$, for some $k \in \mathbb{Z}$, be the kernel of an isogeny leading

to a curve in $\mathcal{I}'$. We have

$$(1 + b\iota)(P + k\iota(P)) = P + k\iota(P) + b(\iota(P) - kP) \equiv P + k\iota(P) \pmod{b}$$

and as b is divisible by 2^k, $P + k\iota(P) \in \mathcal{I}'$ implies $(1 + b\iota)(P + k\iota(P)) \in \mathcal{I}'$. That the action is free and transitive follows from Proposition 3.18 and a counting argument as $|G|/|G'| = 2^{k-1} = |\mathcal{I}_1|/|\mathcal{I}'|$. □

Similarly, we can take subsets of $\mathcal{I}_2$ and $\mathcal{I}_3$ and restrict the acting group.

This gives rise to an attack strategy when N_2 is not large enough. Guessing k bits of the secret before applying the attack on the remaining part allows an attacker to reduce the requirements on the parameters to $N_2 > p(N_1/2^k)^4$. This is the same as when guessing the last bits of the secret.

Given such a partial isogeny, one computes the correct equivalence class $\mathcal{I}'$ from the kernel of the known part of the isogeny. Moreover, one needs to compute the lifts of elements of G' to endomorphisms of norm N_2 or eN_2. Computing the action of G' on the set $\mathcal{I}'$ allows one to test for the hidden shift property. Once it is satisfied, the secret can be recovered by solving an injective abelian hidden shift problem. Otherwise, one can make another guess on the k bits of the secret.

Apart from reducing the requirements on the unbalancedness, guessing part of the isogeny reduces the number of elements one needs to lift and the size of the hidden shift instance. Depending on the concrete parameter sets provided, one may combine exhaustive search and the attack presented in this paper to recover secrets more efficiently.

3.4 The attack on HHS by Childs–Jao–Soukharev

We begin by providing more detail on how the algorithm proposed by Childs, Jao and Soukharev [CJS14] succeeds to construct an isogeny between two given ordinary elliptic curves in quantum subexponential time. The provided strategy can further be applied to CSIDH [CLMPR18].

Recall the free and transitive group action from Section 3.1.2 of the class group on the set of isogenous ordinary curves with the same endomorphism ring. The hard problem is to find an isogeny between two isogenous ordinary elliptic curves with the same endomorphism ring, i.e. reversing this group action. Childs–Jao–Soukharev provide an algorithm that constructs such an isogeny in quantum subexponential time [CJS14] using a reduction to the hidden shift problem.

We summarise the core idea as another instance of our framework using malleability oracles. Let $\mathcal{I} := \mathrm{Cl}(\mathcal{O})$ and $\mathfrak{O} := \mathrm{Ell}_{q,n}(\mathcal{O})$. We can look at the group action defined in Section 3.1.2 as a one-way function

$$f : \mathcal{I} \to \mathfrak{O} \ , \ [x] \mapsto [x] \cdot j(E)$$

for a fixed curve E. Note that the class group $\mathrm{Cl}(\mathcal{O})$ acts on itself and therefore f has a malleability oracle with respect to the class group readily available everywhere on the image, i.e. f is malleable with respect to this group action.

Finding an isogeny $\varphi E \to E'$ is now equivalent to finding the ideal class $[\mathfrak{b}]$ in $\mathrm{Cl}(\mathcal{O})$ containing the ideal corresponding to the kernel of φ, i.e. we would like to compute the preimage of f at $j(E') = [\mathfrak{b}] \cdot j(E)$.

Childs–Jao–Soukharev observed that the functions $F_0, F_1 : \mathrm{Cl}(\mathcal{O}) \to \mathrm{Ell}_{q,n}(\mathcal{O})$, de-

fined by $F_0([x]) = [x] \cdot j(E)$ and $F_1([x]) = [x] \cdot j(E')$, are shifts of each other. Moreover, they are injective functions since the action of the class group on $\mathrm{Ell}_{q,n}(\mathcal{O})$ is free and transitive. The injective abelian hidden shift problem can be solved in quantum subexponential time, which allows one to recover $[\mathfrak{b}]$ and therefore an isogeny φ between the two given curves.

Analogously to the case for ordinary curves, the group action in CSIDH utilising supersingular curves can be attacked this way. Recall that CSIDH uses the $\mathbb{F}_p$-rational endomorphism ring $\mathcal{O}$ of the fixed starting curve E. In the Diffie–Hellman-type key exchange, recovering a party's secret key constitutes of computing their secret ideal class $[\mathfrak{b}] \in \mathrm{Cl}(\mathcal{O})$ which satisfies $[\mathfrak{b}] \cdot E = E_B$ for the party's public curve E_B. Through defining functions $F_0, F_1 : \mathrm{Cl}(\mathcal{O}) \to \mathrm{Ell}_p(\mathcal{O})$ as above by $F_0([x]) = [x] \cdot E$ and $F_1([x]) = [x] \cdot E_B$, it is possible to reduce finding Bob's secret key $[\mathfrak{b}]$ to an instance of the injective hidden shift problem: We have $F_1([x]) = F_0([x] \cdot [\mathfrak{b}])$ for any ideal class $[x] \in \mathrm{Cl}(\mathcal{O})$, where the functions are both injective due to the group action being free and transitive.

3.5 Improvements and outlook

In this chapter we constructed an abelian group action on the key space of the inherently non-commutative SIDH. Having this group action in place allowed us to construct a heuristic malleability oracle using the torsion point information provided in SIDH when overstretched and sufficiently unbalanced parameters are being used. This contradicted the commonly believed misconception that no such group action exists in the branch of isogeny-based cryptography where one considers the full isogeny graph of supersingular elliptic curves. We also embedded our attack in a more general framework that also captures other quantum attacks on schemes in isogeny-based cryptography.

While the attack does *not* apply to balanced parameters as specified in the original SIDH proposal [JD11] or the former NIST post-quantum candidate SIKE [JACCDHJK-

LLNRSU17], it still provided a novel cryptanalytic approach to SIDH. Interestingly, the obstruction to attack SIDH with balanced parameters in our case does not seem to be directly related to the hindrances in other attacks on unbalanced SIDH exploiting torsion point information [Pet17; QKLMPPS21] but to limitations of the KLPT algorithm [KLPT14] and the ones described in Remark 3.33 instead. Improvements to the lifting subroutine included in the KLPT algorithm would not only partially decrease the required unbalancedness of SIDH parameters in this work, but also improve various isogeny-based schemes such as Galbraith–Petit–Silva's signatures [GPS20] and SQISign [DKLPW20]. There are further technical details of our framework which could be improved. For one, studying the free and transitive group action proposed in Section 3.3.2 in more detail could lead to interesting results, in particular to an extension of the group action to a larger class of supersingular curves. This has already been the subject of follow-up work [CIIKLP23] where the full group action of $(\mathcal{O}/N_1\mathcal{O})^*$ is utilised instead of the restrictions to subgroups necessary for our attack strategy (cf. Section 3.3.2). It also remains an open problem to improve the framework further and to give conditions on the malleability oracle that are sufficient to invert one-way functions in quantum polynomial time. Finally, finding one-way functions which are malleable and thus providing applications of this work to areas beyond isogeny-based cryptography is left for future investigation.

CHAPTER 4

Adaptive attack on the Jao–Urbanik variant of SIDH

Personal contributions: *Chapter 4 is based on collaborative work with Andrea Basso, Péter Kutas, Simon-Philipp Merz and Christophe Petit, published as [BKMPW20]. I contributed to all parts of the development of the attack, especially to computing and checking the impact of malformed points on kernel subgroups utilised in the scheme and extracting the required pullbacks, as well as the writing of the corresponding publication.*

Semi-static key exchange protocols often find application in internet-based communication. In such settings, a single pair of corresponding static secret and public keys is used by a server whenever a client wishes to initiate a new session with a corresponding shared key. Hence it is generally reasonable to analyse the security of several key exchanges when a static key is used by one of the parties. As SIDH was found vulnerable against the adaptive GPST attacks (cf. Section 2.1.2), the k-SIDH protocol was proposed to salvage SIDH for settings where non-ephemeral keys are used. The k-SIDH construction basically consists of

generating k secret kernel subgroups per participant such that the associated public keys allow each party to complete k^2 SIDH key exchanges in total. The resulting j-invariants can then be hashed to produce a single secret shared by both parties.

In this chapter, we discuss cryptanalytic results regarding the Jao–Urbanik variant of SIDH which was designed to provide a non-interactive key exchange (NIKE) in light of the GPST attacks on SIDH and the inefficiency of computing k^2 SIDH secrets in k-SIDH; see Section 2.1.2 and Section 2.1.4. The scheme we focus on was introduced by Jao and Urbanik in response to the significant computational efforts required for k-SIDH when compared with SIDH. Additionally, the new proposal [UJ20] provided another alternative protocol which was hoped to be unaffected by adaptive attacks on static SIDH keys since such an active attack against 2-SIDH was devised in [DGLTZ20]. Dobson–Galbraith–LeGrow–Ti–Zobernig (DGLTZ) generalise their strategy to also apply when an arbitrary $k > 2$ is fixed for the k-SIDH protocol, though a number of oracle queries (which correspond to completed protocol interactions with the server) exponential in k is then needed to recover a static secret key.

Recall that the Jao–Urbanik scheme (presented in Section 2.1.4) utilises non-trivial automorphisms on the starting curve in an SIDH instance to derive multiple shared secret j-invariants from one single pair of public keys. The authors claimed that deploying this scheme with k' secret keys per party, thus resulting in a shared secret obtained by hashing the $3(k')^2$ shared j-invariants, would make this variant as secure as k-SIDH with k chosen in such a way that the secret hash also contains $3(k')^2$ j-invariants while being a lot more efficient due to the underlying relationship between secrets and curves. However, our attack demonstrates that the relationship underlying the distinct but related SIDH instances corresponding to a single pair of secret keys stemming from the non-trivial automorphism in Jao–Urbanik's scheme can also be exploited in an active setting. Note that our attack is not efficient enough to break the parameters proposed in [UJ20]. However, it effectively decreases the claimed security of Jao–Urbanik's scheme as well as the efficiency gains over k-SIDH anticipated by

the authors. We show in Section 4.4 how the same security against adaptive attacks can be achieved through less computational effort with k-SIDH.

Though the attack presented in this chapter builds upon results from the GPST attack [GPST16] and [DGLTZ20], there are several crucial differences between the GPST and DGLTZ strategies and ours: While the attack on 2-SIDH requires the treatment of the two SIDH secrets independently, in our case, we can exploit the interrelation of multiple secrets which stems from the use of the non-trivial automorphism. By matching up triples (or pairs, if the order-4 automorphism is utilised) of candidate curves which display the same relation to one another as the curves associated with automorphism-related generators of secret kernels when recovering the first bit for each of the secret keys, this strategy is much more efficient than exhaustive search over all possible combinations of curves. A similar approach, allowing us to only check a smaller number of possible combinations of guessed key bits and candidate pullbacks with the oracle, also improves how we compute key bits and pullbacks simultaneously during the iterative part of the attack rather than individually as in [DGLTZ20]. Furthermore, we reduce the complexity of trying different possible pullback candidates which appear identical to an attacker of k-SIDH by extracting the correct pullbacks explicitly at each step. On the whole, we manage to significantly reduce how many queries to the oracle are necessary to recover a secret key in Jao–Urbanik's scheme through utilising the relationships between isogenies or curves provided by the use of the non-trivial automorphism in the protocol's construction. Given that the adversary is recovering secret isogenies of power-of-two degrees, our attack will be successful after running $\mathcal{O}(32^{k/3})$ protocol instances with the static party. Contrasting to the strategy in [DGLTZ20], we therefore provide a nearly cube root speedup in the same setting.

This chapter is organised as follows. We begin by describing the attack model in which we position our attack and briefly present an overview of the DGLTZ attack [DGLTZ20] in Section 4.1. Our attack is explained in detail in Section 4.2, while we give some information

on generalising the attack in Section 4.3. Finally, we compare the Jao–Urbanik protocol to the k-SIDH scheme in Section 4.4 and highlight remaining open questions in Section 4.5.

4.1 Preliminaries

To set up the presentation of our results, we first give a brief introduction to the general adaptive attack setting and describe the GPST-based attack Dobson–Galbraith–LeGrow–Ti–Zobernig (DGLTZ) devised on 2-SIDH. Finally, we discuss the parameter selection as suggested by Jao and Urbanik for their protocol as well as the impact DGLTZ has on this scheme straightforwardly without any adjustment.

For an active attack like the GPST attack due to Galbraith–Petit–Shani–Ti [GPST16], we generally assume that Alice is an honest party running the SIDH key exchange protocol using a static secret key α, and that the attacker aims to recover this secret key while acting as Bob. As mentioned before, this scenario for example appears in online contexts such as using the TLS protocol where the server fulfils Alice's side of the protocol.

As Alice's secret key remains the same, the attacker can run several instances of the key exchange protocol with Alice to recover her key. The first instance is run honestly in order for the attacker to obtain a valid shared secret with Alice as a reference j-invariant against which to compare curves resulting from future dishonest protocol runs. More specifically, during the following executions of the protocol, the attacker sends maliciously constructed points to Alice who will use them to compute a new shared key. Based on his malformed public information, Bob as the attacker can then make a guess as to what the shared secret curve with Alice could be for this malicious key exchange. It is realistic in the online setting that Bob will find out whether his guess was correct or not, for example by receiving an error message from using an incorrect secret key for secure follow-up communications.

This means that an adversary can extract Alice's secret key bit by bit when using the key exchange scheme as an oracle. Recall the definition of the oracle $\mathcal{O}$ as in Eq. (2.2); $\mathcal{O}$ takes as input E_1, an elliptic curve, two linearly independent, full-order points R, S in the 2^{e_A}-torsion of E_1 as well as a second curve E_2, then it outputs 1 whenever the j-invariants of the curves $E_1/\langle R + [\alpha]S\rangle$ and E_2 coincide, and 0 if not.

4.1.1 The DGLTZ attack

The DGLTZ attack [DGLTZ20] adopts a similar strategy to the well-known GPST attack (cf. Section 2.1.2 based on [GPST16]).

Let us first specify some notation that will be used in the following. The k secret integers $\alpha^{(r)}$ Alice picks during the key generation phase each define a corresponding kernel generator $A^{(r)} = R + [\alpha^{(r)}]S$ in terms of a $E[2^{e_A}]$-basis R, S. As it is easier to present our attack clearly without the convolution of an arbitrary, likely large, number of secrets, we will mostly restrict to the case where $k = 2$, and consider secret values α, β and associated subgroups A, B. Where we use this simplification, it is straightforward to generalise the attack for larger k. Each of the secrets, say α for example, can be written in terms of its individual bits α_i such that $\alpha = K_i^{(a)} + \alpha_i 2^i + \alpha' 2^{i+1}$ when we define $K_i^{(a)}$ to be the sum of the first i bits, hence the partial key known to an attacker who is about to recover the i-th bit. The oracle used in [DGLTZ20, Section 4.1] is the following.

Definition 4.1 (Oracle in k-SIDH)**.** *Let H be some public hash function. Upon receipt of an elliptic curve E, two points R, S spanning $E[2^{e_A}]$ and a hash value h, the oracle reveals whether $h = H\big(j(E/\langle R + [\alpha^{(1)}]S\rangle)||\ldots||j(E/\langle R + [\alpha^{(k)}]S\rangle)\big)$.*

It is important to understand the difference between the oracle from Definition 4.1, which reveals information only about the input tuple of k static secret keys $(\alpha^{(1)}, \ldots, \alpha^{(k)})$, and the usual oracle in GPST-like attacks. The latter often allows the adversary to make

deductions about a single secret key from the provided information. This is not the case in either [DGLTZ20] or the attack described in this chapter.

To overcome this limitation in the information provided by an individual query to the oracle, different hash values must often be tested for each collection of malformed points. In particular, upon receiving Alice's public information an adversary aims to recover the first bits of each of Alice's secret values at the same time, and then repeatedly finds the next bits of each secret in one go. Like in the Galbraith–Petit–Shani–Ti attack, secret bits are extracted through oracle queries with malformed torsion point information. However, instead of simply computing the shared j-invariants resulting from completed SIDH squares with the malicious points for verification by the oracle, for k-SIDH, the hash of all resulting j-invariants concatenated is checked against the correct shared key. By the properties of the hash function, a match in hash values, and thus a positive response from the oracle, can clearly only be achieved if the next bits of each of the k different secret integers have been guessed correctly at the same time. Thus all i-th key bits have to be recovered simultaneously.

To begin with, let us recall how an adversary finds $\alpha_0^{(i)}$ for all secret keys in more detail. The oracle is called on queries which include guesses for the j-invariants j_i of the k elliptic curves which Alice might compute as shared secrets. Oracle inputs then comprise of $\big(E, R, [1+2^{e_A-1}]S, H(j_1||\dots||j_k)\big)$ and can be constructed utilising the following lemma which is a formalisation of results and ideas from [DGLTZ20, Section 5.1].

Lemma 4.2. *Let α be any of Alice's secret keys. Consider the isogeny path from E to E_A, and replace the last step in this path by the only other possible step that leaves the path non-backtracking. Let E'_A be the final curve of this path. Let $s \in \{0,1\}$. Let $R' := R - [s][2^{e_A-1}]S$ and $S' := [1+2^{e_A-1}]S$. Then the SIDH key computed by Alice is either E_A or E'_A. Moreover, it is E_A exactly whenever $\alpha_0 = s$.*

Proof. A straightforward computation gives $A' := R' + [\alpha]S' = R + [\alpha]S + [2^{e_A-1}(\alpha - s)]S = A + [2^{e_A-1}(\alpha_0 - s)]S$. Note that A' has order 2^{e_A} and $[2]A' = [2]A$, so the curves $E_B/\langle A\rangle$ and $E_B/\langle A'\rangle$ are both neighbours of $E_B/\langle [2]A\rangle$ in the 2-isogeny graph. Finally, we have equality $A = A'$ precisely when $\alpha_0 = s$. □

From the above result and the properties of the supersingular 2-isogeny graph, we can compute exactly how many possibilities there are for each curve: the respective $E_{A^{(i)}}$ in case the guess is correct, and the six codomain curves which are isogenous to it via a degree-4 isogeny. Therefore, seven candidate curves have to be checked per secret integer and the total number of candidate j-invariants is 7^k.

During the iterative step, queries of the form

$$(E, (R - [K_i^{(a)}][2^{e_A-i-1}]S, [1 + 2^{e_A-i-1}]S), H(j_1||\dots||j_k))$$

are used so that the j-invariants of elliptic curves $E/\langle A + [\alpha_i][2^{e_A-1}]S\rangle$, $E/\langle B + [K_i^{(b)} - K_i^{(a)}][2^{e_A-i-1}]S + [\beta_i][2^{e_A-1}]S\rangle$ will appear in the hash value. Pursuing the same strategy of exhaustively searching through all possibilities for the next bits as was done for the $\alpha_0^{(i)}$ is infeasible. Even for $k = 2$ exponentially many queries will be required since the number of candidates for each curve at the i-th bit recovery step is one more than the number of 2^{i+1}-neighbours of each correct curve. Let E_i denote the $(n-i)$-th curve in the isogeny path from E to E_B and $\psi_{B,i}$ the associated partial isogeny. To significantly reduce the number of necessary queries, DGLTZ analyse the relation of elliptic curves of the forms $E/\langle B + [K_i^{(b)} - K_i^{(a)}][2^{e_A-i-1}]S + [\beta_i][2^{e_A-1}]S\rangle$ and $E_i/\langle \psi_{B,i}(B + [K_i^{(b)} - K_i^{(a)}][2^{e_A-i-1}]S + [\beta_i][2^{e_A-1}]S)\rangle$ to each other in the isogeny graph, and find that these are isogenous curves connected by a degree-2 isogeny. It is however non-trivial to compute these two curves, so certain pullbacks, namely the intermediate points $\psi_i(B)$ and $[2^{e_A-i}]\psi_i(S)$ on E_i, have to be computed beforehand. These computations are made during each iterative step once the i-th key bits and thus the

partial keys K_{i+1} have been found, and utilise this new knowledge to query the oracle on

$$\left(E, (R - [K_{i+1}^{(a)}][2^{e_A-i-1}]S, [1 + 2^{e_A-i-1}]S), H(j_1||\dots||j_k)\right)$$

as input. From the oracle's response, the attacker can then deduce if the j-invariants $j_1, \dots, j_k$ correspond to curves of the form $E_{i+1}/\langle\psi_{B,i+1}(B + [K_{i+1}^{(b)} - K_{i+1}^{(a)}][2^{e_A-i-1}]S)\rangle$. Note during the bit recovery phase, 2-neighbours of these curves were computed. As stated previously, the oracle deployed for the DGLTZ attack does not facilitate the extraction of the correct pullbacks even if all possible points $\psi_{B,i+1}(B)$ and $[2^{e_A-i-1}]\psi_{B,i+1}(S)$ and combinations thereof are queried for. This is due to the fact that multiple combinations of pullbacks can result in a 1 being returned by the oracle: points $\psi_{B,i+1}(B)$ and $[2^{e_A-i-1}]\psi_{B,i+1}(S)$, as well as points $\psi_{B,i+1}(B)+C$ and $[2^{e_A-i-1}]\psi_{B,i+1}(S)+C$ where C is an order-2 point generating the isogeny between E_{i+1} and E_i. In order to be able to utilise the pullbacks in the following steps, one pullback $\psi_{B,i}(B)$ for B is selected, and two possible points for $[2^{e_A-i-1}]\psi_{B,i+1}(S)$ then form a candidate pullback set. The attacker can use [DGLTZ20, Lemma 2] below to find the required intermediate isogenies for each iteration.

Lemma 4.3. *Let $A^{(i)} = P + [\alpha^{(i)}]Q$ be the generator of the subgroup corresponding to the i-th secret isogeny and let $\psi_j^{(i)} := \phi_{e_A}^{(i)} \circ \phi_{e_A-1}^{(i)} \circ \cdots \circ \phi_{j+1}^{(i)}$. Then, we have*

$$\ker \phi_j^{(i)} = \langle[2^{j-1}]\psi_j^{(i)}(A^{(i)})\rangle, \qquad \ker \hat{\phi}_j^{(i)} = \langle[2^{e_A-1}]\psi_{j-1}^{(i)}(Q)\rangle.$$

At each step, computing the key bits simultaneously requires 24^k queries[1], while computing the pullbacks is possible with 16^k queries when the technical conditions which are addressed in the appendix of [DGLTZ20] are satisfied.

[1]Note that this estimation is not given in [DGLTZ20].

4.1.2 Jao–Urbanik's protocol

Setup

- prime $p = 2^{e_A} \cdot 3^{e_B} \cdot f - 1$.
- supersingular elliptic curve $E : y^2 = x^3 + 1$ with $j(E) = 0$
- non-trivial automorphism $\eta_6 : E \to E$
- torsion bases $\{P_A, \eta_6(P_A)\}$ for $E[2^{e_A}]$ and $\{P_B, \eta_6(P_B)\}$ for $E[3^{e_B}]$

Key Generation

• $\alpha \xleftarrow{\$} \{0, \dots, 2^{e_A} - 1\}$ • $G_A = \langle [\alpha]P_A + \eta_6(P_A) \rangle \subseteq E[2^{e_A}]$ • $\phi_A : E \to E_A = E/G_A$	$E_A, \phi_A(P_B), \phi_A(\eta_6(P_B))$ $\longrightarrow$ $\longleftarrow$ $E_B, \phi_B(P_A), \phi_B(\eta_6(P_A))$	• $\beta \xleftarrow{\$} \{0, \dots, 3^{e_B} - 1\}$ • $G_B = \langle [\beta]P_B + \eta_6(P_B) \rangle \subseteq E[3^{e_B}]$ • $\phi_B : E \to E_B = J/G_B$

Shared Key

$R_B := \phi_B(P_A), S_B := \phi_B(\eta_6(P_A))$ $\qquad$ $R_A := \phi_A(P_B), S_A := \phi_A(\eta_6(P_B))$

$$
\begin{aligned}
E_B/\langle [\alpha]R_B + S_B) \rangle &\cong E_{AB} \cong E_A/\langle [\beta]R_A + S_A \rangle \\
E_B/\langle -R_B + [\alpha+1]S_B \rangle &\cong E_{A\eta_6(B)} \cong E_A/\langle -[\beta+1]R_A + [\beta]S_A \rangle \\
E_B/\langle -[\alpha+1]R_B + [\alpha]S_B \rangle &\cong E_{A\eta_6^2(B)} \cong E_A/\langle -R_A + [\beta+1]S_A \rangle
\end{aligned}
$$

$$h = \mathrm{Hash}\big(j(E_{AB}), j(E_{A\eta_6^2(B)}), j(E_{A\eta_6(B)})\big)$$

Figure 4.1: The Jao–Urbanik protocol using one key and automorphism η_6 on an elliptic curve with j-invariant 0.

Recall the Jao–Urbanik scheme as it was presented in Section 2.1.4. A schematic overview of the key exchange when each party uses one key is given in Fig. 4.1. The security of this scheme is analysed in [UJ20, Section 4] where the authors consider an instantiation with secret isogenies of degree a power of general primes ℓ. While the possibility of exploiting the relationship between kernel subgroups, curves and isogenies from usage of the automorphism in an attack setting is identified, they overestimate the associated cost and thus the security of their scheme as some of the inherent structure of the construction is overlooked. As a conclusion to their brief security discussion, selecting $k' = 18$ keys when using 11-isogenies is proposed to achieve 256-bit security. Though this suggested parameter set (k', ℓ) remains

secure against the active attacks that Jao and Urbanik considered at the time[2] a more thorough security analysis needs to be performed.

In particular, it is imperative to explicitly define an attack model as well as a suitable oracle instead of the one the authors envision during their analysis. This oracle confirms whether multiple elliptic curves which are provided as input are all intermediate curves on the secret isogeny between E and $E/\langle A\rangle$.[3] As we will later show, the additional structure present in instances of the Jao–Urbanik protocol ensures that corresponding intermediate curves on the three related secret isogenies are isomorphic. Our result implies that using this type of oracle in this novel setting is suboptimal. Additionally, Dobson, Galbraith, LeGrow, Ti and Zobernig dismiss a straightforward generalisation to k-SIDH of the oracle used in [GPST16] as inefficient for any $k < 1$. Thus they are required to compute the aforementioned pullbacks in order to make valid and useful queries and have to accept the resulting increase of the attack complexity. In other words, in the k-SIDH setting, the cost of the call to an oracle which returns true if and only if all the guessed curves are on the correct path is not constant but exponential in k. This observation clearly applies to the Jao–Urbanik scheme as well.

Current impact of the DGLTZ attack on Jao–Urbanik's scheme

Applying the DGLTZ attack to the Jao–Urbanik protocol is not straightforward. The DGLTZ attack assumes that all the secret kernels are of the form $\langle [\alpha]P + Q\rangle$ which is not the case in the Jao–Urbanik scheme due to the following. To one secret the following three kernels are associated: $\langle [\alpha]P + Q\rangle$, $\langle -P + [\alpha+1]Q\rangle$, $\langle -[\alpha+1]P + [\alpha]Q\rangle$. The parity of the coefficient of Q in the second and the third kernel is different, thus in particular, it is impossible that both of them are odd (hence for every λ-multiple of the kernel the coeffi-

[2]Clearly the class of Castryck–Decru-type attacks from Section 2.1.2 also pose a threat to the Jao–Urbanik protocol whenever it uses isogeny chains of degree-ℓ isogenies where (ℓ, ℓ)-isogenies can be efficiently computed.

[3]It is shown how a comparable oracle can be implemented for regular SIDH in [GPST16].

cient of Q will be even). This difficulty could potentially be overcome, however a number of $\mathcal{O}(24^k)$ queries, where $k = 3k'$ and k' is the number of secrets per party in the Jao–Urbanik scheme, will still be required.

Therefore, we intend to utilise the relationship between the kernel subgroups and curves present in the protocol setting to devise an adaptive attack which provides a close to cube root speed-up. Our strategy is described in the next section and requires $\mathcal{O}(32^{\frac{k}{3}})$ oracle queries.

4.2 Adaptive attack against the Jao–Urbanik scheme

We now present our active attack on the Jao–Urbanik NIKE [UJ20]. As the version of the scheme utilising an order-6 automorphism is the one promising maximal security gains over a regular SIDH instance, we focus on the case where the base curve E is of j-invariant 0 and has the non-trivial automorphism η_6 of order 6. To simplify the exposition below, we focus on attacking Alice's torsion and denote by ℓ and n the prime and exponent, ℓ_A and e_A respectively, which she uses. Firstly, we consider the case where $\ell = 2$, and later broaden the scope of our attack in Section 4.3.1 by discussing a generalisation to arbitrary primes ℓ. Select a basis $\{P, Q\}$ of the 2^n-torsion of E whose elements satisfy the relation $Q = \eta_6(P)$. Denote by α one of the random secret integers Alice chose during key generation. Then, as shown in Section 2.1.4, three distinct isogeny kernels can be constructed from α with the help of η_6. The subgroup generators and corresponding isogenies are

$$A = [\alpha]P + Q, \quad A' = \eta_6(A) = -P + [\alpha + 1]Q, \quad A'' = \eta_6^2(A) = -[\alpha + 1]P + [\alpha]Q,$$

and

$$\psi_{A,0} : E \to E_A = E/\langle A \rangle, \quad \psi'_{A,0} : E \to E'_A = E/\langle A' \rangle, \quad \psi''_{A,0} : E \to E''_A = E/\langle A'' \rangle.$$

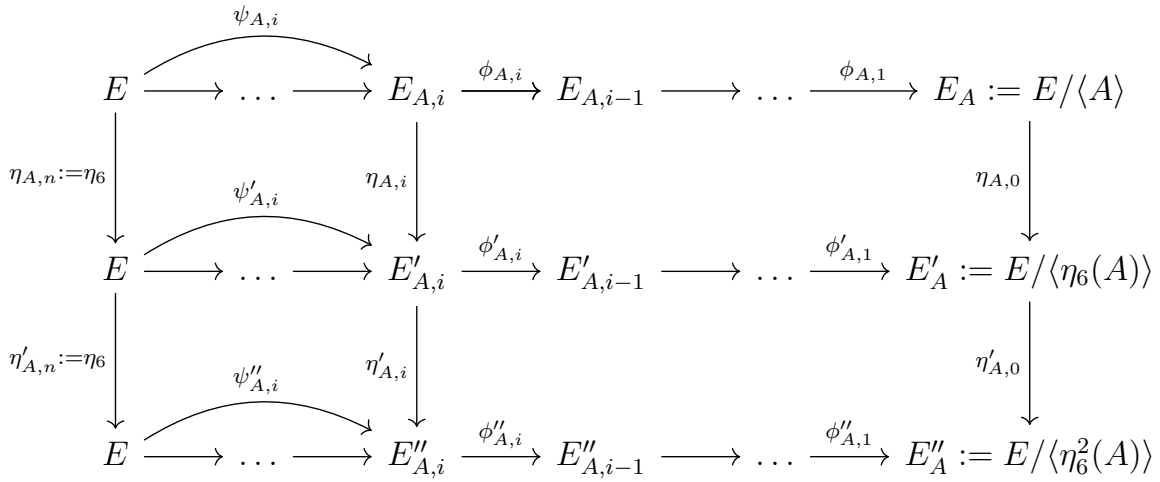

Figure 4.2: Overview of the related isogeny paths and curves in an instance of the Jao–Urbanik protocol.

Let γ, C, C', C'', and E_C, E'_C, E''_C denote any of Alice's secret keys other than α, as well as the three kernel generators based on γ, as well as the corresponding codomain curves respectively. We will generally indicate by the subscript C instances derived from non-α secret keys, but also omit subscripts when it is clear in the context from which secret key a curve, isogeny or kernel originates or when we state a universally applicable property.

The isogeny $\psi_{A,0}$ can be decomposed into n individual 2-isogenies. We index intermediate curves by $E_{A,i}$, with $E_{A,0} = E_A$ and $E_{A,n} = E$. The intermediate isogenies are denoted by $\phi_{A,i} : E_{A,i} \to E_{A,i-1}$. We also call $\psi_{A,i}$ the composition $\phi_{A,n} \circ \ldots \circ \phi_{A,i+1}$ from E to $E_{A,i}$. Notation related to the curves E'_A and E''_A is obtained similarly. Let furtermore the isomorphisms established in Lemma 4.5, $E_A \cong E'_A \cong E''_A$, be η and η', and indicate their shifts to intermediate curves by subscripts i. Also define the pullbacks

$$A_i := \psi_{A,i}(A), \text{ and } P_i := \psi_{A,i}(P).$$

Fig. 4.2 provides an overview of the notations we use, as well as a visual depiction of the interconnection between related curves and isogenies as in the Jao–Urbanik NIKE.

As discussed above, we cannot straightforwardly adapt either the GPST [GPST16] or DGLTZ [DGLTZ20] strategies to the Jao–Urbanik setting. Still, the two-stage structure of first recovering the first bit of each secret key, and then iteratively computing further key bits and pullbacks of kernel generators also builds the foundation for our attack: Section 4.2.2 describes how the parity of each secret key is discovered and the pullbacks A_1, A_1', A_1'' and $[2^{n-1}]P_1$, $[2^{n-1}]P_1'$, $[2^{n-1}]P_1''$ are computed for each key, while Section 4.2.3 shows how knowledge of all partial secret keys $K_{A,i}$ as well as the previously computed pullbacks allows an adversary to recover the next key bits of each secret together with new pullbacks on intermediate curves further along the isogeny path. More precisely, the second phase of the attack shows that if we let $K_{A,i}$ denote the partial secret key already recovered and express α as

$$\alpha = 2^{i+1}\alpha' + 2^i\alpha_i + K_{A,i},$$

α_i can be found via oracle queries using the partial keys and the i-th pullbacks.

Even though this strategy closely resembles that of the DGLTZ attack, we adapt it further to reflect the additional structure provided by the automorphism-related secrets, curves and isogenies that is present in Jao and Urbanik's scheme. In particular, we eliminate combinations of curves from our queries which violate the underlying relationship, reducing their total number during the bit computation, and find the precise pullbacks during each phase of the attack. The resulting attack implies that k'-SIDH is only marginally less secure than when Jao–Urbanik's key exchange is instantiated with k' secret keys. While we leave a more meticulous complexity analysis to Section 4.2.3, we can already conclude that the desired improvements of the Jao–Urbanik protocol over k-SIDH are limited.

Like in [DGLTZ20], we initially use the starting curve E as a basis for our attack. However in Section 4.3.2, we provide a strategy to shift the attack in order to work with queries using points on arbitrary curves. Note that this also generalises the DGLTZ key

recovery algorithm.

In [BKMPW20, Lemma 7, Lemma 8], some important results about the generators of the intermediate isogenies $\psi_{A,i}$, $\psi'_{A,i}$ and $\psi''_{A,i}$, as well as the relationship of the associated elliptic curves $E_{A,i}$, $E'_{A,i}$, $E''_{A,i}$ are highlighted. We present more detailed proofs of both results below.

Lemma 4.4. *For simplicity, denote subscripts of the form A, i by i. Then,*

$$\begin{aligned}
&\ker \psi_i = \langle [2^i]A \rangle, && \ker \psi'_i = \langle [2^i]A' \rangle, && \ker \psi''_i = \langle [2^i]A'' \rangle,\\
&\ker \phi_i = \langle [2^{i-1}]A_i \rangle, && \ker \phi'_i = \langle [2^{i-1}]A'_i \rangle, && \ker \phi''_i = \langle [2^{i-1}]A''_i \rangle,\\
&\ker \hat{\phi}_i = \langle [2^{n-1}]P_{i-1} \rangle, && \ker \hat{\phi}'_i = \langle [2^{n-1}]P'_{i-1} \rangle, && \ker \hat{\phi}''_i = \langle [2^{n-1}]P''_{i-1} \rangle.
\end{aligned}$$

Proof. The first identity follows from the fact that the subgroup generated by A has a unique subgroup of order 2^{n-i} for every i which implies that this subgroup must be the kernel of the 2^{n-i}-isogeny ψ_i. The same argument shows that the kernel of the partial isogeny ψ'_i is generated by $[2^i]A' = [2^i]\eta_6(A)$, and that $\ker \psi''_i$ is generated by $[2^i]A''$. Deductions about the kernels of ϕ_i and $\hat{\phi}_i$ follow from [DGLTZ20, Lemma 5.1], while an application of the automorphism η_6 confirms the remaining kernel generators. □

Lemma 4.5. *Let notation be as above. Then $E_{A,i}$, $E'_{A,i}$ and $E''_{A,i}$ are isomorphic.*

Proof. We have that $\ker \psi_{A,i} \subseteq \ker(\psi'_{A,i} \circ \eta_{A,n})$. Thus, there exists an isogeny $\eta_{A,i} : E_{A,i} \to E'_{A,i}$ such that $\psi'_{A,i} \circ \eta_{A,n} = \eta_{A,i} \circ \psi_{A,i}$. By examining the degrees, we find that $\deg \eta_{A,i} = 1$ and thus $\eta_{A,i}$ is an isomorphism. The same reasoning holds for $E''_{A,i}$. □

The isomorphisms $\eta_{A,i}$ and $\eta'_{A,i}$ are assumed to be known when $E_{A,i}$, $E'_{A,i}$ and $E''_{A,i}$ are known, since they can be easily computed (a 1-isogeny between two curves can be recovered in time $\mathcal{O}(1)$).

4.2.1 Our attack model

In this section, we describe our assumptions and our attack model: Firstly, let k' denote the number of Alice's secret keys. We assume that Alice has a static set of keys $\alpha^{(1)}, \ldots, \alpha^{(k')}$ all chosen at random from $\mathbb{Z}/2^n\mathbb{Z}$ and that the attacker impersonates Bob to recover Alice's secret keys. The attacker engages with Alice on sessions of Jao–Urbanik's protocol and sends particularly chosen data, not necessarily conforming to the protocol. By checking whether the two parties have obtained the same shared secret, the attacker may recover information on Alice's keys. We model this information leakage in terms of an oracle and represent each interaction with Alice as an oracle query.

An adaption of the second oracle presented in [DGLTZ20] to the η_6 variant of the Jao–Urbanik protocol gives an oracle $O'(E^{(1)}, \ldots, E^{(k')}, (R^{(1)}, S^{(1)}), \ldots, (R^{(k')}, S^{(k')}), h)$ that returns *true* if

$$h = \mathrm{Hash}(j_{1,1}||j_{1,2}||j_{1,3}|| \ldots ||j_{k',k'-1}||j_{k',k'}),$$

where $j_{r,s}$ denotes the concatenation of

$$j\left(E^{(r)}/\langle [\alpha^{(r)}]R^{(s)} + S^{(s)}\rangle\right), \quad j\left(E^{(r)}/\langle -R^{(s)} + [\alpha^{(r)}+1]S^{(s)}\rangle\right) \text{ and}$$
$$j\left(E^{(r)}/\langle -[\alpha^{(r)}+1]R^{(s)} + [\alpha^{(r)}]S^{(s)}\rangle\right).$$

Similarly to what is done for the third oracle in [DGLTZ20], we can simplify the oracle by assuming that the attacker generates one secret key and sends repeated copies of the same curve and points. Note that any information that can be recovered with querying with distinct curves can also be recovered by querying with repeated copies of the same curve.

Hence, we obtain the following oracle

$$\mathcal{O}(E, (R, S), h) = \mathcal{O}'(E, \ldots, E, (R, S), \ldots, (R, S), h), \tag{4.1}$$

which is the one we use in our attack. As noted in [DGLTZ20], the attacker could change one curve at each iteration to slightly obscure their intentions. All curves but one ($k'-1$ in this case) have to remain constant across iterations for the attack to succeed.

4.2.2 Exploiting the additional structure: first step

Let us focus on one of Alice's secrets α. The attack extends straightforwardly to all the keys. In order to recover the first bits of α, the attacker sends the modified points $P' = [1+2^{n-1}]P$, $Q' = Q$, so that Alice uses the following kernels in her computation of the shared secret, where we mark modified kernel generators and elliptic curves by $\hat{\cdot}$.

1. $\hat{A} = \langle [\alpha]P' + Q' \rangle = \langle [\alpha]P + Q + [\alpha_0][2^{n-1}]P \rangle$,
2. $\hat{A}' = \langle -P' + [\alpha+1]Q' \rangle = \langle -P + [\alpha+1]Q + [2^{n-1}]P \rangle$,
3. $\hat{A}'' = \langle -[\alpha+1]P' + [\alpha]Q' \rangle = \langle -[\alpha+1]P + [\alpha]Q - [\alpha_0+1][2^{n-1}]P \rangle$.

Note that, depending on the value of the least significant bit α_0, either the first or third curve computed has not been altered by using the modified points. Thus the attacker already knows one of $j(\hat{E}_A)$ or $j(\hat{E}_{A''})$, where $\hat{E}_A = E/\langle \hat{A} \rangle$, although at this stage not which one of the two.

The attacker now computes $\mathcal{E}_A^*$, the sets containing all six proper 4-neighbours of the curves E_A in Alice's public key, and their respective j-invariants. If $\alpha_0 = 0$, $\langle [\alpha]P' + Q' \rangle = \langle A \rangle$, and hence the first curve Alice obtains is isomorphic to her original E_A. The second curve is independent of α_0 and is a 4-neighbour of E'_A, since they share the 2-neighbour $E/\langle 2A' \rangle$. Similarly, the third curve is a 4-neighbour of E''_A since they share $E/\langle 2A'' \rangle$ as a 2-neighbour. Note that the intermediate 2-neighbours in this construction are isomorphic since their kernel generators differ only by an application of η_6. Hence, the three curves E_A, $E/\langle -P' + [\alpha+1]Q' \rangle$ and $E/\langle -[\alpha+1]P' + [\alpha]Q' \rangle$ are the three distinct 2-neighbours of

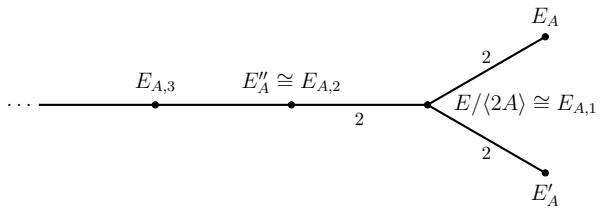

Figure 4.3: The isogeny paths between E_A, E'_A and E''_A.

$E/\langle 2A\rangle$ (distinctness follows from simple computations on the kernel generators), as depicted in Fig. 4.3.

Analogously if $\alpha_0 = 1$, we find that the three computed curves all share a common 2-neighbour. The attacker proceeds analogously for the choices of any other curve. This allows the attacker to match up candidate curves for E_A, E'_A and E''_A among the 4-neighbours of E_A, depending on which combination of first key bits they are querying for at the time: The attacker may choose any curve in $\mathcal{E}_A^*$ as a candidate curve for E'_A, depending on the guessed bit they may select E_A or E''_A to be equal to E_A and then select the unique curve in $\mathcal{E}_A^*$ which is also a 4-neighbour of E'_A as a candidate for the remaining curve. Querying the oracle for all possible combinations ($12^{k/3}$ combinations, six for each neighbour and one for the curve itself) gives the attacker the first bit of each secret.

Now, given the position of E_A, E'_A and E''_A in the isogeny graph, we know that $E/\langle 2A\rangle$ must be the first intermediate curve $E_{A,1}$ and similarly E''_A must be $E_{A,2}$; see Fig. 4.3. This means the attacker can easily recover the first two intermediate curves without additional oracle queries, unlike what happens in the DGLTZ attack. Since the isogenies between E_A and $E_{A,1}$ (i.e. $\phi_{A,1}$) and between $E_{A,1}$ and $E_{A,2}$ (i.e. $\phi_{A,2}$) are known, the attacker can compute the first pullbacks of A and $[2^{n-1}]P$ (up to odd scalar multiplication) by setting A_1 to be a generator of $\ker(\phi_{A,1})$ and $[2^{n-1}]P_{A,1}$ a generator of $\ker(\hat{\phi}_{A,2})$ by Lemma 4.4. Finally, the attacker obtains the further pullbacks via the isomorphisms as $A'_1 = \eta_{A,1}(A_1)$ and $A''_1 = \eta'_{A,1}(A_1)$. This approach can be easily repeated for every following curve.

4.2.3 Intermediate bit and pullback computation

Suppose we have recovered the first i bits of each key and have computed the relevant pullbacks. Let α be one of Alice's secrets keys and let γ denote any other secret key. Now, we want to recover the $(i+1)$-th bits and compute the new pullbacks. In the DGLTZ attack, the bit recovery and pulling back are two separate stages, but in order to exploit the additional structure of Jao–Urbanik's scheme, we combine them into one: The attacker does not actively recover the $(i+1)$-th key bits, but instead tries all the $2^{k'}$ possibilities and uses the pullback queries to validate both the bit guesses and the pullback candidates.

Using Lemma 4.4, it is possible to compute $\hat{\phi}_{i+1}$ and thus recover ϕ_{i+1}. With this information, the attacker can obtain candidates for the pullbacks of A and P. The same applies to ϕ'_{i+1} and ϕ''_{i+1}.

Hence, the attacker proceeds by querying the oracle with the points

$$P' = [1 + 2^{n-i-1}]P \text{ and } Q' = Q - [K_{A,i}][2^{n-i-1}]P,$$

which would lead Alice to make the following kernel computations

$$\begin{aligned}
\langle [\alpha]P' + Q' \rangle &= \langle A + [\alpha_i][2^{n-1}]P \rangle, \\
\langle -P' + [\alpha+1]Q' \rangle &= \langle A' - [K_{A,i}^2 + K_{A,i} + 1][2^{n-i-1}]P \\
&\quad + [K_{A,i}][\alpha_i][2^{n-1}]Q \rangle, \\
\langle -[\alpha+1]P' + \alpha Q' \rangle &= \langle A'' - [K_{A,i}^2 + K_{A,i} + 1][2^{n-i-1}]P \\
&\quad - [K_{A,i} + 1][\alpha_i][2^{n-1}]P \rangle,
\end{aligned}$$

$$\begin{aligned}
\langle [\gamma]P' + Q' \rangle &= \langle C + [K_{C,i} - K_{A,i}][2^{n-i-1}]P \\
&\quad + [\gamma_i][2^{n-1}]P \rangle, \\
\langle -P' + [\gamma+1]Q' \rangle &= \langle C' - [K_{C,i}K_{A,i} + K_{A,i} + 1][2^{n-i-1}]P \\
&\quad - [K_{A,i}][\gamma_i][2^{n-1}]P \rangle, \\
\langle -[\gamma+1]P' + [\gamma]Q' \rangle &= \langle C'' - [K_{C,i}K_{A,i} + K_{A,i} + 1][2^{n-i-1}]P \\
&\quad - [K_{A,i} + 1][\gamma_i][2^{n-1}]P \rangle.
\end{aligned}$$

All kernels can be shifted with ψ_{i+1} (e.g. $E/\langle C + [K_{C,i} - K_{A,i}][2^{n-i-1}]P + [\gamma_i][2^{n-1}]P \rangle = E_{C,i+1}/\langle C_{i+1} + [K_{C,i} - K_{A,i}][2^{n-i-1}]P_{C,i+1} + [\gamma_i][2^{n-1}]P \rangle$) similarly to the DGLTZ attack by applying [Sil09, Corollary III.4.11.]. Now, since the candidate pullbacks for A_{i+1} (preimages of A_i via $\phi_{A,i}$), C_{i+1} (preimages of C_i via $\phi_{C,i}$), $[2^{n-i-1}]P_{C,i+1}$ (preimages of $\left[\frac{1}{2}\right][2^{n-i}]P_{C,i}$), $[2^{n-i-1}]P_{A,i+1}$ (preimages of $\left[\frac{1}{2}\right][2^{n-i}]P_{A,i}$) and their isomorphic correspondents are known, the attacker can query the oracle with the hash values of all $2^{k'}2^{k'}8^{k'}$ possibilities (2 for each bit, 2 for the kernel generator pullback candidates and $4 \cdot 2$ for the P pullback candidates). Note that the attacker may try a candidate for the first curve and then shift it to the second curve using the isomorphisms η_i or η_i' (therefore reducing an a priori complexity of 32^k to $32^{k'}$). We show that if we find a match it means that we have found the correct pullbacks for C_{i+1} and $P_{C,i+1}$ as well as the correct key bits for C. To begin with, we include a result from [BKMPW20, Lemma 9].

Lemma 4.6. *Let $K_{A,i}$, $K_{C,i}$ be natural numbers. Then,*

1. *$K_{A,i}^2 + K_{A,i} + 1$ is odd.*
2. *It is not possible that all of $(K_{A,i} - K_{C,i})$, $(K_{A,i}K_{C,i} + K_{A,i} + 1)$ and $(K_{A,i}K_{C,i} + K_{C,i} + 1)$ have the same parity.*

Proof. The first claim is trivial. For the second claim, observe that the sum of these quantities

is even, thus it is not possible that all three of them are odd. If $K_{A,i} - K_{C,i}$ is even, then $K_{A,i}$ and $K_{C,i}$ have the same parity and then $K_{A,i}K_{C,i} + K_{A,i} + 1 = K_{A,i}(K_{C,i} + 1) + 1$ is odd. □

Now, we prove our main lemma.

Lemma 4.7. *If the oracle query returns* true*, then we have found* γ_i, C_{i+1} *and* $P_{C,i+1}$.

Proof. Suppose the attacker guesses that α_i is 0. It is clear from the above computation that we always get at least one match when we substitute C_{i+1}, γ_i and $P_{C,i+1}$. If $\gamma_i = 0$, then it follows from the computation of [DGLTZ20, Claim 1], that the number of matches for the first curve is exactly two. The other match corresponds to choosing $C_{i+1} + [2^i]C_{i+1}$ as the preimage of C_i and $[2^{n-i-1}]P_{C,i+1} + [2^i]C_{i+1}$ as the preimage of $\left[\frac{1}{2}\right][2^{n-i}]P_{C,i}$. Due to Lemma 4.6, it is not possible that $(K_{A,i}-K_{C,i})$, $(K_{A,i}K_{C,i}+K_{A,i}+1)$ and $(K_{A,i}K_{C,i}+K_{C,i}+1)$ are all odd. Assume for instance that $(K_{A,i} - K_{C,i})$ is odd and $(K_{A,i}K_{C,i} + K_{A,i} + 1)$ is even. Then we show that the second curve will not match as its kernel will be generated by $C'_{i+1} + [K_{C,i}K_{A,i} + K_{A,i} + 1][2^{n-i-1}]P_{C,i+1} + [2^i]C_{i+1}$. Hence it will be 4-isogenous to the queried curve. The other cases follow similarly.

When $\gamma_i = 1$, then there will be another match for the first curve. Namely when we pull back $\left[\frac{1}{2}\right][2^{n-i}]P_i$ as $[2^{n-i-1}]P_{i+1} + [2^{n-1}]P_{i+1}$. However, again a similar calculation to [DGLTZ20, Claim 1] (one has to distinguish cases depending on the parity of $K_{A,i}$ and $K_{C,i}$) shows that either the second or the third curve will not match. The calculations when the attacker guesses α_i to be 1 are analogous. □

Lemma 4.7 implies that for all secrets except α we know the correct bits and pullbacks (as otherwise we cannot receive 1 from the oracle). However, we have seen that the coefficient $K_{A,i}^2 + K_{A,i} + 1$ is odd, thus there will be multiple matches. In order to retrieve α_i and the

corresponding pullbacks we perform another query with different points, switching $K_{A,i}$ with $K_{C,i}$. For this, we can use the previously computed pullbacks and thus only query the oracle 32 times (corresponding to the 32 possibilities for the pullbacks and the bit). Since the correct pullbacks are computed, we are able to recover the isogenies $\phi_{A,i+1}$ and $\phi_{C,i+1}$ using Lemma 4.4 as before. Finally, since the next intermediate curves are now known, we compute the isomorphisms between them. Thus, we have proven the following theorem.

Theorem 4.8. *1. There exists an algorithm that recovers the first bit of each secret using $\mathcal{O}(12^{k'}) = \mathcal{O}(12^{\frac{k}{3}})$ queries to the oracle defined in (4.1).*

2. There exists an algorithm that recovers the intermediate bits and pullbacks using $\mathcal{O}(32^{k'}) = \mathcal{O}(32^{\frac{k}{3}})$ queries to the oracle defined in (4.1).

4.3 Generalising the attack

We now discuss a way to broaden the scope of our attack. We first give an analysis of how the attack works and performs when the ℓ^e-torsion is attacked where ℓ is a general prime, instead of assuming that Alice utilises isogenies of degree 2^n. We further describe how one can query from the public key curve E_B of the attacker as would be realistic in an attack setting, and how one can then lift the problem to recovering the path $E \to E_A$ as in Section 4.2.

4.3.1 Attack costs for general ℓ

So far, we have demonstrated our attack on the Jao–Urbanik protocol with parameter choice $\ell = 2$ for simplicity. However, in their proposal, the authors suggest the use of $\ell = 11$ or $\ell = 13$ and further compute that $k' = 18$ keys are necessary to obtain security against Grover's algorithm for $\ell = 11$; see [UJ20, Section 4]. Thus we briefly assess the cost of our attack and the DGLTZ attack for arbitrary ℓ. We divide the discussion into two parts. First, we estimate the number of queries needed for computing the first key bits and later

the number of queries needed in the iterative step.

The complexity estimate of our attack is a straightforward generalisation of Theorem 4.8. During the recovery of the first bit of every key, we query - as before - for any of the $\ell^{k'}$ possible first ℓ-adic digit combinations by first fixing the curve (either E_A or E''_A using notation as in Section 4.2.2) corresponding to the guessed key digit to be the curve given in Alice's public key. Then we select any of the $\ell(\ell+1)$ ℓ^2-neighbours of the correct curve to be E'_A and choose one of the remaining $\ell - 1$ curves which are ℓ^2-isogneous to both previously selected curves as the third curve associated to a given key. Hence, for each possible combination of first key digits we have $\big(\ell(\ell+1)(\ell-1)\big)^{k'}$ choices of curves. Thus, there exists an algorithm which recovers the first digit of each secret using $\mathcal{O}(\ell^{k'}\ell^{3k'}) = \mathcal{O}(\ell^{4k'}) = \mathcal{O}(\ell^{\frac{4k}{3}})$ oracle queries.

For the iterative step, we again first guess the i-th ℓ-adic digits and then compute candidate preimages for the first curve and shift them to the other two curves using the respective isomorphisms. There are $\ell^{k'}$ possibilities for the digits and $\ell^{2k'}$ possibilities for each preimage. This implies that we need $\mathcal{O}(\ell^{5k'})$ queries in total.

Hence, for general ℓ, we can summarise our findings in the following theorem.

Theorem 4.9. *1. There exists an algorithm that recovers the first digit of each secret using $\mathcal{O}(\ell^{4k'}) = \mathcal{O}(\ell^{\frac{4k}{3}})$ queries to the oracle defined in (4.1).*

2. There exists an algorithm that recovers the intermediate digits and pullbacks using $\mathcal{O}(\ell^{5k'}) = \mathcal{O}(\ell^{\frac{5k}{3}})$ queries to the oracle defined in (4.1).

4.3.2 Querying with E_B

The following lemma shows how to lift from the path $E_B \to E_{AB}$ to the path $E \to E_A$ which we have been using to query along in Section 4.2.

Lemma 4.10. *Let $\psi_{A,i}$ be the partial isogeny from E to E_i (as in Fig. 4.2) and let $\psi^B_{A,i}$ be the corresponding partial isogeny from E_B to E^B_i on the isogeny path from E_B to E_{AB}. Let A generate the kernel of the isogeny from E to E_A and let $A_B = \phi_B(A)$. Define $\delta_i : E^B_i \to E_i$ to be the isogeny which is the SIDH lift via ϕ_B. Assume we know $\psi^B_{A,i}(A_B)$ and $\psi^B_{A,i}(\phi_B(Q))$ for some $Q \in E$. Then we can compute $[3^{e_B}]\psi_{A,i}(A)$ and $[3^{e_B}]\psi_{A,i}(Q)$.*

Proof. By defining δ_i as the lifting isogeny between the two parallel isogeny paths, we have constructed a setting where $\delta_i \circ \psi'_i = \psi_i \circ \hat{\phi_B}$. From knowing $\psi^B_{A,i}(A_B)$, we can find

$$[3^{e_B}]\psi_{A,i}(A) = \psi_{A,i}(\hat{\phi_B}(\phi_B(A))) = \delta_i(\psi^B_{A,i}(A_B)).$$

The computation for the scaled image of Q works analogously. □

This lemma can be applied to compute the relevant pullbacks on the isogeny paths from E to E_A, E' to E'_A and E'' to E''_A in the following manner: First one computes a pullback candidate on the path starting from E_B. Then it is lifted with the above lemma to the path starting from E (using the fact that 3^{e_B} is odd and hence coprime to the degree of ϕ_A). Then it can further be shifted to the other two isomorphic curves via the automorphism η_6. Finally, these points can be shifted back with ϕ_B.

4.4 Comparison of k-SIDH and Jao–Urbanik's protocol

The algorithm described in Section 4.2 and in particular in Theorem 4.9 does not break the security parameters suggested by Jao and Urbanik. However, in order to assess the security gain of Jao–Urbanik's protocol, we compare it with the security of k-SIDH for arbitrary ℓ. Since the DGLTZ method requires an extra step which computes the i-th digits and then uses that information to compute candidate pullbacks, the overall complexity of the attack is ℓ^{4k} for k-SIDH. Table 4.1 gives an overview of the number of SIDH-instances and public key

	# SIDH instances	# public key exchanges	Attack cost
Jao–Urbanik with k' keys	$3(k')^2$	$(k')^2$	$\mathcal{O}(\ell^{5k'})$
k-SIDH with $k = k'$	$(k')^2$	$(k')^2$	$\mathcal{O}(\ell^{4k'})$
k-SIDH with $k = \frac{5}{4}k'$	$(\frac{5}{4}k')^2 \approx 1.56(k')^2$	$\approx 1.56(k')^2$	$\mathcal{O}(\ell^{4\frac{5}{4}k'}) = \mathcal{O}(\ell^{5k'})$

Table 4.1: Comparisons between Jao–Urbanik's scheme and k-SIDH

exchanges occurring when executing the different protocols, as well as the respective cost of attacking the ℓ-torsion.

We can observe that the Jao–Urbanik protocol with k' secrets is as secure as $\frac{5k'}{4}$-SIDH when comparing necessary oracle queries for an adaptive attack to recover the secret keys. Consequently, it is more efficient to use $\frac{5k'}{4}$-SIDH than the Jao–Urbanik scheme with k' keys and the same ℓ when measuring security with respect to the attacks prior to the full key recovery of SIDH, as the former has a computational cost equivalent to $3(k')^2$ SIDH exchanges whereas the latter has a computational cost equivalent to $1.56(k')^2$ SIDH exchanges. Note that the Jao–Urbanik scheme maintains a moderate advantage in public key size, since it requires sharing k' keys, compared to the $\frac{5}{4}k'$ keys shared in k-SIDH.

4.5 Improvements and outlook

We have introduced an adaptive attack against Jao–Urbanik's protocol with parameter $\ell = 2$. As shown in Section 2.1.2, this scheme is fully broken passively as it requires Alice to release the same public information to as in SIDH and an attacker can then run a Castryck–Decru-type attack. However, Jao and Urbanik suggest using $\ell = 11$ or $\ell = 13$, which would require an attacker to compute (ℓ, ℓ)-isogenies between abelian surfaces to recover secret keys. This is much less efficient than the attacks on standard parameters where chains of degree-2 and degree-3 isogenies are used due to more complicated arithmetic for larger primes ℓ.

The active attack presented in this chapter can more easily be adapted to work for general choices of ℓ as we showed in Section 4.3.1. The complexity of such an attack increases significantly, possibly reaching levels where the protocol is secure for the specified parameter sets against this type of attack. However, even in that case, our attack provides a nearly cubic speedup compared to a generic application of the DGLTZ attack against the Jao–Urbanik scheme. Assessing the security of k-SIDH and Jao–Urbanik's variant of it against adaptive attacks, we conclude that Jao–Urbanik's protocol does not seem to offer a sufficient security improvement over k-SIDH with the same number of secret keys to justify the roughly double the number of computations needed. It remains an interesting issue beyond the cryptanalysis of the Jao–Urbanik protocol how the existence of non-trivial automorphisms can aid in the construction and the cryptanalysis of isogeny-based schemes using the curves with j-invariants 0 or 1728, or those close to them.

Though outperformed by the full key recovery SIDH attacks, adaptive attacks on SIDH variants as such still allow some insight into vulnerabilities in isogeny-based constructions. It remains imperative to analyse not only the usage of ephemeral keys but also that of static ones as non-interactive key exchange is still a desirable primitive for certain cryptographic settings. The knowledge amassed through adaptive attacks such as GPST, DGLTZ and our attack can first of all contribute to the assessment of how much information in addition to the two given curves of the pure isogeny problem (cf. Problem 2.1) makes the resulting problem efficiently solvable. Furthermore, these attacks can be used as guides when analysing the security of future protocols, above all those which have been proposed as SIDH variants deploying countermeasures against the Castryck–Decru-type of attacks.

CHAPTER 5

Cryptanalysis of Genus-2 SIDH

Personal contributions: *Chapter 5 is based on collaborative work with Sabrina Kunzweiler and Yan Bo Ti, published as [KTW21]. I contributed to all results presented in this chapter, especially the case distinction algorithm and development of the subsequent attack which allows a full key recovery.*

Like when considering hyperelliptic curves for elliptic curve cryptography, examining higher-dimensional abelian varieties as base objects for isogeny-based protocols promises some interesting trade-offs between parameter sizes, efficiency and security. One of the first such proposed schemes, Genus-2 SIDH (G2SIDH) [FT19], adapts SIDH to using principally polarised supersingular or superspecial abelian surfaces in a natural generalisation to the two-dimensional setting. Hence it seems crucial to examine the hardness of the isogeny problem with given torsion point information for abelian surfaces, and in particular, whether an adaptive attack based on GPST and its descendants is possible in this new setting. Such an attack on a static-key implementation of G2SIDH was already tentatively assumed to exist by the authors of [FT19]. The implications of the existence of such an attack on G2SIDH

would be the same as the GPST attack had on SIDH. Specifically, an attack would indicate that in the genus-2 setting, the computational Diffie–Hellman problem can be reduced to the decisional Diffie–Hellman problem, and that CCA2-protection is needed when non-ephemeral keys are deployed. Before the publication of the results presented in this chapter, an attack bearing these connotations was not known.

In order to formulate such an efficient, polynomial-time adaptive attack, we first have to examine the keyspace originally proposed for the G2SIDH protocol in much more detail than was done in [FT19]. In particular, we begin by characterising the types of kernel subgroups of admissible, non-backtracking chains of (ℓ, ℓ)-isogenies as they appear in the G2SIDH protocol, and devise a way to normalise kernel generators to achieve a proper classification of all maximal ℓ^{e_ℓ}-Weil isotropic subgroups of $J[\ell^{e_\ell}]$ for some Jacobian J of a hyperelliptic genus-2 curve. This classification of subgroups along with choosing to work with a symplectic torsion basis instead of an arbitrary one allows us to suggest a significantly improved key generation procedure for G2SIDH. The original description of the scheme [FT19] called for a total of twelve secret integers fulfilling certain linear congruences. One can immediately see that the number of solutions to the congruences exceeds the number of admissible isogenies, so that either different solutions define the same isogenies or some solutions do not yield valid secret keys. Solving these linear equations during the key generation procedure is inconvenient and adds quite costly computations to the overall runtime estimate of the key exchange. If we tolerate a slight restriction of the keyspace analogous to the restriction widely accepted for SIDH, we can select secret keys uniformly at random by simply uniformly sampling a number of integers. We give the details of the classification of kernel subgroups and the new key generation method in Section 5.1.

After having determined what a party's secret scalars look like, i.e. what type of scalars an adversary wants to recover in an adaptive attack, we give a precise strategy for extracting such a secret through malformed interactions with an honest party modeled by an

oracle providing one bit of information per query. We give more information on our oracle and attack model in Section 5.2.1. Though our attack is based on the techniques of the foundational GPST attack [GPST16], GPST cannot be applied straightforwardly. The main difference between our adaptive attack and the SIDH attack lies in the number of secret scalars and the number of kernel generators associated with each cryptosystem which we overcome through distinguishing the different types of kernel subgroups.

The adaptive attack recovers Alice's secret isogeny which we assume, for ease of exposition, to be a chain of Richelot isogenies by first determining the type of kernel subgroup, i.e. the form the normalised kernel generators take in our classification. Depending on the type determined via our case distinction algorithm described in Section 5.2.3, we might have already discovered partial knowledge of the scalars defining the kernel subgroup. This knowledge is then leveraged to launch the adaptive attack as is presented in Section 5.2.4 and Section 5.2.5 for kernels of rank 2 and rank 3 respectively.

Strategies to generalise the active attack from Section 5.2 to arbitrary torsion bases instead of symplectic ones is presented in Section 5.3. Additionally, we revisit the SIDH scheme and the GPST attack on its static variant in Section 5.4 in order to draw parallels between the genus-1 and genus-2 settings and to justify and further explain the suggested use of a restricted keyspace and our attack strategy. We conclude in Section 5.5 with a discussion of the attack in light of the successful SIDH attacks which also impact G2SIDH, and give avenues for further exploration related to the contents of this chapter.

5.1 Secret keys in G2SIDH

The G2SIDH key exchange protocol as presented in Section 2.2.1 requires each party to sample a secret key. Such a private key can be expressed as an isogeny of principally polarised abelian surfaces; for example $\phi_A : J \to J_A$ in the case of Alice. We have seen in Section 1.3.1

that this isogeny corresponds to a maximal 2^{e_A}-isotropic subgroup of the starting variety J, so that Alice samples an isogeny of degree 2^{e_A}. Analogously, Bob's secret key corresponds to a maximal 3^{e_B}-isotropic subgroup of J defining an isogeny $\phi_B : J \to J_B$. As discussed in Remark 1.5, the sampled isogenies are required to be non-backtracking so that their degree is as desired. For $\ell \in \{2, 3\}$ and $n = e_A$ or e_B, respectively, we can therefore write the keyspace as

$$\mathcal{K}_\ell = \{G \subset J \mid G \text{ maximal } \ell^n\text{-isotropic and } G \not\subset J[m] \text{ for any } m < \ell^n\},$$

and characterise the elements of $\mathcal{K}_\ell$ via the following result.

Proposition 5.1 ([FT19, Proposition 2])**.** *Let $G \in \mathcal{K}_\ell$, then G is isomorphic to*

$$\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^n} \quad \text{or} \quad \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$$

for some $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$.

As the above proposition highlights again, the kernel subgroups making up the keyspace $\mathcal{K}_\ell$ have rank two or three, and in particular, are not cyclic as is the case in elliptic curve isogeny-based contexts.

In the following, we analyse the full keyspace and show that it is possible to sample (almost) uniformly from the entirety of $\mathcal{K}_\ell$. We further define $\mathcal{K}_\ell^{\text{res}} \subset \mathcal{K}_\ell$, a slight restriction of the keyspace which preserves the order of magnitude of the keyspace. True and straightforward random sampling is achievable when $\mathcal{K}_\ell^{\text{res}}$ is used. To accomplish this, we make use of *symplectic* bases for $J[\ell^n]$ which allow us to normalise secret keys and specify canonical representatives generating classes of the groups in $\mathcal{K}_\ell$. This classification is presented in more detail in Section 5.1.3.

5.1.1 The G2SIDH keyspace

Let us briefly recall the method of sampling secrets suggested in the original G2SIDH protocol [FT19] as described in Section 2.2.2. In particular, given a PPSSAS J and a basis $(P_1, \dots, P_4)$ for $J[\ell^n]$ the authors describe a method to generate a secret maximal isotropic subgroup of the ℓ^n-torsion which by Proposition 5.1 must be isomorphic to

$$\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$$

for some $0 \leq k \leq \lfloor \frac{n}{2} \rfloor$.

First, a random $k \in \{1, \dots, \lfloor \frac{n}{2} \rfloor\}$ is chosen. Note that to sample from all possible maximal ℓ^n-isotropic subgroups uniformly, k cannot be picked uniformly. We give more detail about the distribution of different values of k in Section 5.1.4.

Next, four scalars $\alpha_{1,1}, \dots, \alpha_{1,4} \in \mathbb{Z}/\ell^n\mathbb{Z}$ are picked at random. Since these scalars will produce the first kernel generator of full order ℓ^n, at least one of the scalars must not be divisible by ℓ. Finally, the remaining scalars $\alpha_{i,1}, \dots, \alpha_{i,4}$ for $i \in \{2, 3\}$ are found by solving the following linear congruences.

$$\begin{pmatrix} \alpha_{1,1}\alpha_{2,2} - \alpha_{1,2}\alpha_{2,1} & +x_{1,3}(\alpha_{1,1}\alpha_{2,3} - \alpha_{1,3}\alpha_{2,1}) & +x_{1,4}(\alpha_{1,1}\alpha_{2,4} - \alpha_{1,4}\alpha_{2,1}) \\ +x_{2,3}(\alpha_{1,2}\alpha_{2,3} - \alpha_{1,3}\alpha_{2,2}) & +x_{2,4}(\alpha_{1,2}\alpha_{2,4} - \alpha_{1,4}\alpha_{2,2}) & +x_{3,4}(\alpha_{1,3}\alpha_{2,4} - \alpha_{1,4}\alpha_{2,3}) \end{pmatrix} \equiv 0 \pmod{\ell^k}$$

and

$$\begin{pmatrix} \alpha_{1,1}\alpha_{3,2} - \alpha_{1,2}\alpha_{3,1} & +x_{1,3}(\alpha_{1,1}\alpha_{3,3} - \alpha_{1,3}\alpha_{3,1}) & +x_{1,4}(\alpha_{1,1}\alpha_{3,4} - \alpha_{1,4}\alpha_{3,1}) \\ +x_{2,3}(\alpha_{1,2}\alpha_{3,3} - \alpha_{1,3}\alpha_{3,2}) & +x_{2,4}(\alpha_{1,2}\alpha_{3,4} - \alpha_{1,4}\alpha_{3,2}) & +x_{3,4}(\alpha_{1,3}\alpha_{3,4} - \alpha_{1,4}\alpha_{3,3}) \end{pmatrix} \equiv 0 \pmod{\ell^{n-k}},$$

where $x_{i,j}$ denotes the integer such that $e_{\ell^n}(P_i, P_j) = e_{\ell^n}(P_1, P_2)^{x_{i,j}}$ for $i, j \in \{1, \dots, 4\}$ and

$e_{\ell^n}(P_1, P_2)$ a primitive ℓ^n-th root of unity.

This produces the tuple $(\alpha_{1,1}, \ldots, \alpha_{3,4})$ of secret scalars defining the generators of the kernel $G = \langle G_1, G_2, G_3 \rangle$ corresponding to the secret isogeny as

$$G_1 = \sum_{i=1}^{4} [\alpha_{1,i}]P_i, \quad G_2 = \sum_{i=1}^{4} [\alpha_{2,i}]P_i, \text{ and } G_3 = \sum_{i=1}^{4} [\alpha_{3,i}]P_i\,.$$

By construction, the points G_1, G_2 and G_3 are of order ℓ^n, ℓ^{n-k} and ℓ^k, respectively, and satisfy the necessary Weil pairing-conditions $e_{\ell^n}(G_1, G_2) = e_{\ell^n}(G_1, G_3) = e_{\ell^n}(G_2, G_3) = 1$.

This implies that $\langle G_1 \rangle \simeq \mathcal{C}_{\ell^n}$, $\langle G_2 \rangle \simeq \mathcal{C}_{\ell^{n-k}}$, $\langle G_3 \rangle \simeq \mathcal{C}_{\ell^k}$ and that $G = \langle G_1, G_2, G_3 \rangle$ is isotropic. However, the congruences above do not always guarantee that G is isomorphic to $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$. In the worst case, one could have chosen $\alpha_{2,1}, \ldots, \alpha_{3,4}$ such that $G_2 = \ell^k G_1$ and $G_3 = \ell^{n-k} G_1$. In this case $G = \langle G_1 \rangle$ is a cyclic group of order ℓ^n, and in particular not maximal ℓ^n-isotropic. This issue is not addressed in [FT19], but it is fixed using the selection process that we describe in the subsequent sections.

5.1.2 Symplectic bases

We begin by stating the defining characteristics of a symplectic basis. Recall that for an integer m coprime to p, the m-torsion of J is a finitely generated rank-4 group, i.e.

$$J[m] \xrightarrow{\sim} \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}.$$

Definition 5.2 (Symplectic basis)**.** *We say that a tuple (P_1, P_2, Q_1, Q_2) is a* basis *for $J[m]$ if it generates $J[m]$ as a group. We say that the basis (P_1, P_2, Q_1, Q_2) for $J[m]$ is* symplectic *with respect to the Weil pairing if*

$$e_m(P_i, Q_j) = \zeta^{\delta_{ij}}, \quad e_m(P_1, P_2) = e_m(Q_1, Q_2) = \zeta^0 = 1,$$

where ζ *is some primitive* m*-th root of unity and* $\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$

Due to the alternating property of the Weil-pairing, we always have $e_m(Q_j, P_i) = \zeta^{-\delta_{ij}}$ for a symplectic basis (P_1, P_2, Q_1, Q_2).

An alternative way of phrasing the definition is to say that a basis is symplectic if the associated pairing matrix is of the form

$$(\log(\zeta, e_m(P, Q)))_{P,Q \in \{P_1,P_2,Q_1,Q_2\}} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix},$$

where the logarithm is taken with respect to ζ.

We can always find a symplectic basis of the m-torsion of J by transforming an arbitrary basis into one satisfying the conditions of Definition 5.2. The straightforward process is formalised in Algorithm 5; see Section 5.3.1.

Lastly, it is essential to make note of the following result [KTW21, Lemma 1] which shows that we can use symplectic bases at any stage of the Genus-2 SIDH key exchange since they are preserved under the isogenies considered in the protocol.

Lemma 5.3. *Let* (P_1, P_2, Q_1, Q_2) *be a symplectic basis of* $J[m]$ *with respect to some primitive root* ζ*, and let* $\phi : J \to J'$ *be an isogeny whose degree is coprime to* m*. Then* $\big(\phi(P_1), \phi(P_2), \phi(Q_1), \phi(Q_2)\big)$ *is a symplectic basis of* $J'[m]$ *with respect to* $\zeta^{\deg(\phi)}$*.*

Proof. Observe that we have

$$e_m(\phi(P_i),\phi(Q_j)) = e_m(P_i,Q_j)^{\deg\phi} = 1 \text{ and } e_m(\phi(Q_i),\phi(P_j)) = e_m(P_j,Q_i)^{-\deg\phi} = 1$$

for all $i \neq j$. Likewise, we have that

$$e_m(\phi(P_i),\phi(Q_i)) = e_m(P_i,Q_i)^{\deg\phi} = \zeta^{\deg\phi}$$

for $i = 1, 2$. Finally, since $\langle\phi(P_1),\phi(P_2),\phi(Q_1),\phi(Q_2)\rangle = J'[m]$ and the torsion subgroup is of rank 4, we can conclude that $\big(\phi(P_1),\phi(P_2),\phi(Q_1),\phi(Q_2)\big)$ is a symplectic basis of $J'[m]$. □

5.1.3 Classification of secret keys

In this section we suggest a normalisation algorithm that produces canonical generators for each group $G \in \mathcal{K}_\ell$. For this purpose, we let

$$(P_1^*, P_2^*, P_3^*, P_4^*) := (P_1, P_2, Q_1, Q_2)$$

be a symplectic basis for $J[\ell^n]$.[1] Clearly, this method can be applied for arbitrary bases by including a change of basis transformation at the beginning of the algorithm, similar to how we describe attacking arbitrary bases in Section 5.3.2.

Let $(\alpha_{1,1},\ldots,\alpha_{3,4})$ denote the secret scalars defining the elements

$$G_1 = \sum_{i=1}^{4}[\alpha_{1,i}]P_i^*, \quad G_2 = \sum_{i=1}^{4}[\alpha_{2,i}]P_i^*, \text{ and } G_3 = \sum_{i=1}^{4}[\alpha_{3,i}]P_i^*,$$

of J which generate the subgroup $G = \langle G_1, G_2, G_3\rangle \in \mathcal{K}_\ell$. G must be maximal ℓ^n-isotropic in order to preserve the polarisation on the surfaces involved, and hence we can use the

[1] The notation $(P_1^*, P_2^*, P_3^*, P_4^*)$ is used here to make it easier to express the impact of permuting the basis elements. As it does not reflect that the basis is symplectic, we only use it in this section of the thesis.

Weil-pairing to deduce several conditions on the scalars $(\alpha_{1,1}, \ldots, \alpha_{3,4})$.

Normalising the secret scalars is based on the well-known Gaussian elimination procedure. Let

$$A = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} \end{pmatrix} \in M_{3,4}(\mathbb{Z}/\ell^n\mathbb{Z}).$$

Permuting columns if necessary, we can apply elementary row operations in order to produce a matrix of the form

$$A \sim_\sigma \begin{pmatrix} 1 & 0 & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{or} \quad A \sim_\sigma \begin{pmatrix} 1 & * & * & * \\ 0 & \ell^k & * & * \\ 0 & 0 & * & \ell^{n-k} \end{pmatrix}, \tag{5.1}$$

if $G \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^n}$ or $G \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$, respectively. We let σ denote the permutation in S_4 expressing the necessary permutation of the columns, and $*$ be a placeholder for a positive integer, possibly adhering to certain divisibility conditions. Note that the left matrix is a special case of the right by setting $k = 0$. Therefore we will centre the following discussion around the latter.

As desired, the process of normalising the subgroup generators does not change the appearance of the group. More precisely, let

$$A' = \begin{pmatrix} \alpha'_{1,1} & \alpha'_{1,2} & \alpha'_{1,3} & \alpha'_{1,4} \\ \alpha'_{2,1} & \alpha'_{2,2} & \alpha'_{2,3} & \alpha'_{2,4} \\ \alpha'_{3,1} & \alpha'_{3,2} & \alpha'_{3,3} & \alpha'_{3,4} \end{pmatrix}$$

be obtained from A by applying elementary row operations and potentially swapping columns. Let $\sigma \in S_4$ denote the corresponding permutation of the columns. Then $G = \langle G'_1, G'_2, G'_3 \rangle$, where

$$G'_1 = \sum_{i=1}^{4} [\alpha'_{1,i}] P^*_{\sigma(i)}, \quad G'_2 = \sum_{i=1}^{4} [\alpha'_{2,i}] P^*_{\sigma(i)}, \quad G'_3 = \sum_{i=1}^{4} [\alpha'_{3,i}] P^*_{\sigma(i)}\,.$$

We only used the knowledge of the group structure of G to obtain a presentation as an upper triangular matrix as in (5.1). Additionally, we also know that the Weil pairing $e_{\ell^n}(G_i, G_j) = 1$ for any $i, j \in \{1, 2, 3\}$. This property can be used to work out the relations between the non-zero entries of the matrices. [KTW21, Proposition 2] below summarises these results.

Proposition 5.4. *Let A be a matrix corresponding to a maximal ℓ^n-isotropic subgroup of the form $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$ for some integer $0 \leq k \leq \lfloor \frac{n}{2} \rfloor$. Then there exist a permutation $\sigma \in D_8 = \langle (1234), (13) \rangle$ and scalars $a \in \{0, \ldots, \ell^n - 1\}$, $b \in \{0, \ldots, \ell^{n-k} - 1\}$, $c \in \{0, \ldots, \ell^{n-2k} - 1\}$, $d \in \{0, \ldots, \ell^k - 1\}$ such that*

$$A \sim_\sigma A' = \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & s_\sigma\, \ell^k\, (b - cd) & \ell^k\, c \\ 0 & 0 & -s_\sigma\, \ell^{n-k}\, d & \ell^{n-k} \end{pmatrix} \in M_{3,4}(\mathbb{Z}/\ell^n\mathbb{Z}),$$

where $s_\sigma = \operatorname{sgn}(\sigma)$ denotes the sign of the permutation σ.

On the other hand, if A' is as above and $G' = \langle G'_1, G'_2, G'_3 \rangle$, where

$$G'_1 = \sum_{i=1}^{4} [\alpha'_{1,i}] P^*_{\sigma(i)}, \quad G'_2 = \sum_{i=1}^{4} [\alpha'_{2,i}] P^*_{\sigma(i)}, \quad G'_3 = \sum_{i=1}^{4} [\alpha'_{3,i}] P^*_{\sigma(i)}$$

for some $\sigma \in D_8$, then G' is maximal ℓ^n-isotropic.

Proof. Following the Gaussian elimination process one obtains a matrix A' of the form given in (5.1). Note that the rank-2 case is just the special case obtained by setting $k = 0$. Examining this process more closely, one sees that σ can be chosen to lie in the dihedral group $D_8 = \langle (1234), (13) \rangle$.[2]

Let us write

$$A' = \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & \ell^k x & \ell^k c \\ 0 & 0 & \ell^{n-k} y & \ell^{n-k} \end{pmatrix}$$

for some $a, b, c, d, x, y \in \{0, \ldots, \ell^n - 1\}$, now including the divisibility by ℓ-powers which was omitted in (5.1). First, note that after adding a multiple of the second line to the first line of A', we may assume that $d \in \{0, \ldots, \ell^k - 1\}$. Similarly, we can achieve $b \in \{0, \ldots, \ell^{n-k} - 1\}$ and $c \in \{0, \ldots, \ell^{n-2k} - 1\}$. It remains to show that x and y are determined by the scalars a, b, c, d. This is done using the Weil pairing. For the following computation, it is important to note that

$$e_{\ell^n}(P^*_{\sigma(1)}, P^*_{\sigma(3)}) = e_{\ell^n}(P^*_{\sigma(2)}, P^*_{\sigma(4)})^{s_\sigma} \tag{5.2}$$

for all $\sigma \in D_8$.

Let G'_1, G'_2, G'_3 be the generators corresponding to the matrix A'. Then

$$\begin{aligned} e_{\ell^n}(G'_1, G'_2) &= e_{\ell^n}(P^*_{\sigma(1)} + [d]P^*_{\sigma(2)} + [a]P^*_{\sigma(3)} + [b]P^*_{\sigma(4)}, \ell^k \cdot (P^*_{\sigma(2)} + [x]P^*_{\sigma(3)} + [c]P^*_{\sigma(4)})) \\ &= e_{\ell^n}(P^*_{\sigma(1)}, P^*_{\sigma(3)})^{\ell^k x} \cdot e_{\ell^n}(P^*_{\sigma(2)}, P^*_{\sigma(4)})^{\ell^k(cd-b)}. \end{aligned}$$

Using Property (5.2), we obtain the condition $\ell^k x = s_\sigma \ell^k (b - cd)$. Computing the Weil pairing on G'_2 and G'_3 shows that $\ell^{n-k} y = -s_\sigma \ell^{n-k} d$.

[2] In the rank-2 case, we moreover have $\sigma \in V_4 = \langle (13), (24) \rangle \subset D_8$.

For the other direction, it remains to show that the group $G' = \langle G'_1, G'_2, G'_3 \rangle$ is maximal ℓ^n-isotropic. This can be done by verifying that G' meets the criteria from Definition 1.4. □

The corollary below follows straightforwardly from Proposition 5.4 and significantly simplifies the type distinction of secret kernel subgroups during the adaptive attack which is presented in Section 5.2.3.

Corollary 5.5. *Let $(P_1^*, P_2^*, P_3^*, P_4^*)$ be a symplectic basis for $J[\ell^n]$ and let $G \subset J$ be an isotropic group isomorphic to $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$. Assume that $G_1 = P^*_{\sigma(1)} + [d]P^*_{\sigma(2)} + [a]P^*_{\sigma(3)} + [b]P^*_{\sigma(4)} \in G$ for some permutation $\sigma \in D_8$ and scalars a, b, d. Then*

$$\mathcal{C}_{\ell^k} \times \mathcal{C}_{\ell^k} \simeq \ell^{n-k} \left\langle P^*_{\sigma(2)} + [s_\sigma \cdot b]P^*_{\sigma(3)},\ [-s_\sigma \cdot d]P^*_{\sigma(3)} + P^*_{\sigma(4)} \right\rangle \subset G.$$

Proposition 5.4 further implies that each subgroup $G \in \mathcal{K}_\ell$ can be represented by a tuple of the form (a, b, c, d, k, σ), where

$$a \in \{0, \ldots, \ell^n - 1\},\ b \in \{0, \ldots, \ell^{n-k} - 1\},\ c \in \{0, \ldots, \ell^{n-2k}\},\ d \in \{0, \ldots, \ell^k - 1\},$$

for some $0 \leq k \leq \left\lfloor \frac{n}{2} \right\rfloor$ and $\sigma \in D_8$.

Clearly, such a characterisation is not unique in most cases. However, tweaks to the elimination algorithm, such as enforcing a specific order of application to the permutations $\sigma \in D_8$, can result in a deterministic procedure. Thus we can find canonical representatives (a, b, c, d, k, σ), where non-trivial permutations σ imply some additional constraints on the parameters a, b, c, d defining the generators.

Definition 5.6 (Classification)**.** *Let $(P_1^*, P_2^*, P_3^*, P_4^*)$ be a symplectic basis for $J[\ell^n]$ and*

denote by $\mathbf{P}^*$ *the column vector* $\begin{pmatrix} P_1^* & P_2^* & P_3^* & P_4^* \end{pmatrix}^T$. *For a group* $G = \langle G_1, G_2 \rangle \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^n}$ *in* $\mathcal{K}_\ell$, *we say that* G_1, G_2 *are the* canonical generators *if one of the following is true for some* $a, b, c \in \mathbb{Z}/\ell^n\mathbb{Z}$.

2.1 $\begin{pmatrix} G_1 \\ G_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a & b \\ 0 & 1 & b & c \end{pmatrix} \mathbf{P}^*$.

2.2 $\begin{pmatrix} G_1 \\ G_2 \end{pmatrix} = \begin{pmatrix} 1 & b & a & 0 \\ 0 & c & -b & 1 \end{pmatrix} \mathbf{P}^*$, *and* $\ell \mid c$.

2.3 $\begin{pmatrix} G_1 \\ G_2 \end{pmatrix} = \begin{pmatrix} a & 0 & 1 & b \\ -b & 1 & 0 & c \end{pmatrix} \mathbf{P}^*$, *and* $\ell \mid a, b$.

2.4 $\begin{pmatrix} G_1 \\ G_2 \end{pmatrix} = \begin{pmatrix} a & b & 1 & 0 \\ b & c & 0 & 1 \end{pmatrix} \mathbf{P}^*$, *and* $\ell \mid a, b, c$.

For a group $G = \langle G_1, G_2, G_3 \rangle \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$ *with* $0 < k < \frac{n}{2}$ *in* $\mathcal{K}_\ell$, *we say that* G_1, G_2, G_3 *are the* canonical generators *if one of the following is true for some* $a \in \{0, \ldots, \ell^n - 1\}$, $b \in \{0, \ldots, \ell^{n-k} - 1\}$, $c \in \{0, \ldots, \ell^{n-2k} - 1\}$, $d \in \{0, \ldots, \ell^k - 1\}$.

3.1 $\begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & \ell^k(b - cd) & \ell^k c \\ 0 & 0 & -\ell^{n-k} d & \ell^{n-k} \end{pmatrix} \mathbf{P}^*$.

3.2 $\begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} 1 & b & a & d \\ 0 & \ell^k c & -\ell^k(b - cd) & \ell^k \\ 0 & \ell^{n-k} & \ell^{n-k} d & 0 \end{pmatrix} \mathbf{P}^*$, *and* $\ell \mid c$.

3.3 $\begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} a & d & 1 & b \\ -\ell^k(b - cd) & \ell^k & 0 & \ell^k c \\ \ell^{n-k} d & 0 & 0 & \ell^{n-k} \end{pmatrix} \mathbf{P}^*$, *and* $\ell \mid a$.

3.4 $\begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} a & b & 1 & d \\ \ell^k(b - cd) & \ell^k c & 0 & \ell^k \\ -\ell^{n-k} d & \ell^{n-k} & 0 & 0 \end{pmatrix} \mathbf{P}^*$, *and* $\ell \mid a, c$.

3.5 $\begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} d & 1 & b & a \\ \ell^k & 0 & \ell^k c & \ell^k(b - cd) \\ 0 & 0 & \ell^{n-k} & -\ell^{n-k} d \end{pmatrix} \mathbf{P}^*$, *and* $\ell \mid b, d$.

3.6 $\begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} b & 1 & d & a \\ \ell^k c & 0 & \ell^k & -\ell^k(b - cd) \\ \ell^{n-k} & 0 & 0 & \ell^{n-k} d \end{pmatrix} \mathbf{P}^*$, *and* $\ell \mid b, c, d$.

$$3.7 \quad \begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} d & a & b & 1 \\ \ell^k & -\ell^k(b-cd) & \ell^k c & 0 \\ 0 & \ell^{n-k} d & \ell^{n-k} & 0 \end{pmatrix} \mathbf{P}^*, \quad \text{and } \ell \mid a, b, d.$$

$$3.8 \quad \begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} b & a & d & 1 \\ \ell^k c & \ell^k(b-cd) & \ell^k & 0 \\ \ell^{n-k} & -\ell^{n-k} d & 0 & 0 \end{pmatrix} \mathbf{P}^*, \quad \text{and } \ell \mid a, b, c, d.$$

For a group $G = \langle G_1, G_2, G_3 \rangle \simeq \mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^k} \times \mathcal{C}_{\ell^k}$ with $k = \frac{n}{2}$ in $\mathcal{K}_\ell$, we say that G_1, G_2, G_3 are the canonical generators *if one of the following is true for some $a \in \{0, \ldots, \ell^n - 1\}$, $b, d \in \{0, \ldots, \ell^k - 1\}$.*

$$4.1 \quad \begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & \ell^k b & 0 \\ 0 & 0 & -\ell^k d & \ell^k \end{pmatrix} \mathbf{P}^*.$$

$$4.2 \quad \begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} a & d & 1 & b \\ -\ell^k b & \ell^k & 0 & 0 \\ \ell^k d & 0 & 0 & \ell^k \end{pmatrix} \mathbf{P}^*, \text{ and } \ell \mid a.$$

$$4.3 \quad \begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} d & 1 & b & a \\ \ell^k & 0 & 0 & \ell^k b \\ 0 & 0 & \ell^k & -\ell^k d \end{pmatrix} \mathbf{P}^*, \text{ and } \ell \mid b, d.$$

$$4.4 \quad \begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \begin{pmatrix} d & a & b & 1 \\ \ell^k & -\ell^k b & 0 & 0 \\ 0 & \ell^k d & \ell^k & 0 \end{pmatrix} \mathbf{P}^*, \text{ and } \ell \mid a, b, d.$$

Moreover we say that a group $G \in \mathcal{K}_\ell$ is of Type *2.i, 3.i or 4.i for $i \in \{1, \ldots, 8\}$ depending on which of the cases above applies.*

Table 5.1 summarises the classification of the groups in $\mathcal{K}_\ell$ defined above. The classification of the groups in $\mathcal{K}_\ell$ also allows us to determine the cardinality of $\mathcal{K}_\ell$. The number of groups of a given type can be directly read off from the description and is provided in the last column of Table 5.1. Adding up the numbers for Types 2.1, 2.2, 2.3, 2.4, we obtain $\ell^{3n-3}(\ell^2+1)(\ell+1)$, the number of maximal isotropic subgroups of rank 2. Adding up the

	type	σ	condition on (a, b, c, d)	cardinality
$k = 0$	2.1	id	-	ℓ^{3n}
	2.2	(24)	$\ell \mid c$	ℓ^{3n-1}
	2.3	(13)	$\ell \mid a, b$	ℓ^{3n-2}
	2.4	(13)(24)	$\ell \mid a, b, c$	ℓ^{3n-3}
$0 < k < \frac{n}{2}$	3.1	id	-	ℓ^{3n-2k}
	3.2	(24)	$\ell \mid c$	$\ell^{3n-2k-1}$
	3.3	(13)	$\ell \mid a$	$\ell^{3n-2k-1}$
	3.4	(13)(24)	$\ell \mid a, c$	$\ell^{3n-2k-2}$
	3.5	(12)(34)	$\ell \mid b, d$	$\ell^{3n-2k-2}$
	3.6	(1234)	$\ell \mid b, c, d$	$\ell^{3n-2k-3}$
	3.7	(1432)	$\ell \mid a, b, d$	$\ell^{3n-2k-3}$
	3.8	(14)(23)	$\ell \mid a, b, c, d$	$\ell^{3n-2k-4}$
$2k = n$	4.1	id	-	ℓ^{2n}
	4.2	(13)	$\ell \mid a$	ℓ^{2n-1}
	4.3	(12)(34)	$\ell \mid b, d$	ℓ^{2n-2}
	4.4	(1432)	$\ell \mid a, b, d$	ℓ^{2n-3}

Table 5.1: Classification of maximal ℓ^n-isotropic subgroups.

numbers for Types $3.1-3.8$, we find that there are $\ell^{3n-2k-4}(\ell^2+1)(\ell+1)^2$ groups isomorphic to $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^{n-k}} \times \mathcal{C}_{\ell^k}$, where $0 < k < \frac{n}{2}$. Finally the sum over the numbers for Types $4.1-4.4$ is equal to $\ell^{2n-3}(\ell^2+1)(\ell+1)$, the number of groups isomorphic to $\mathcal{C}_{\ell^n} \times \mathcal{C}_{\ell^k} \times \mathcal{C}_{\ell^k}$, where $2k = n$. These cardinalities coincide with the numbers provided in [FT19, Proposition 3].

5.1.4 Uniform sampling from the restricted keyspace

In the previous section we described a classification of the groups in $\mathcal{K}_\ell$. This can be used to sample uniformly from the entire keyspace. Here, we introduce a slightly restricted keyspace $\mathcal{K}_\ell^{\text{res}}$ that allows a particularly easy way of sampling from the keyspace which chooses elements uniformly at random. For the convenience of the reader, Fig. 5.1 provides an explicit

Setup

- prime $p = 2^{e_A} \cdot 3^{e_B} \cdot f - 1$
- superspecial hyperelliptic curve $H/\mathbb{F}_{p^2}$ with Jacobian J
- symplectic bases $(P_{A,1}, P_{A,2}, Q_{A,1}, Q_{A,2})$ for $J[2^{e_A}]$ and $(P_{B,1}, P_{B,2}, Q_{B,1}, Q_{B,2})$ for $J[3^{e_B}]$

Key Generation

• $a_i \xleftarrow{\$} \{0, \ldots, 2^{e_A} - 1\}$ for $i = 1, 2, 3$ • $A_1 = P_{A,1} + [a_1]Q_{A,1} + [a_2]Q_{A,2}$ $A_2 = P_{A,2} + [a_2]Q_{A,1} + [a_3]Q_{A,2}$ • $\phi_A : J \to J_A = J/\langle A_1, A_2 \rangle$	$J_A, \phi_A(P_{B,i}), \phi_A(Q_{B,i})$ for $i \in \{1, 2\}$ $\longrightarrow$ $\longleftarrow$ $J_B, \phi_B(P_{A,i}), \phi_B(Q_{A,i})$ for $i \in \{1, 2\}$	• $b_i \xleftarrow{\$} \{0, \ldots, 3^{e_B} - 1\}$ for $i = 1, 2, 3$ • $B_1 = P_{B,1} + [b_1]Q_{B,1} + [b_2]Q_{B,2}$ $B_2 = P_{B,2} + [b_2]Q_{B,1} + [b_3]Q_{B,2}$ • $\phi_B : J \to J_B = J/\langle B_1, B_2 \rangle$

Shared Key

$$J_B/\langle \phi_B(A_1), \phi_B(A_2) \rangle = \quad J/\langle A_1, A_2, B_1, B_2 \rangle \quad = J_A/\langle \phi_A(B_1), \phi_A(B_2) \rangle$$

Figure 5.1: G2SIDH with restricted keyspace $\mathcal{K}_\ell^{res}$.

description of the G2SIDH protocol in this setting.

For some fixed symplectic basis (P_1, P_2, Q_1, Q_2) of $J[\ell^n]$, we define the restricted keyspace as

$$\mathcal{K}_\ell^{\text{res}} = \{\langle P_1 + [a]Q_1 + [b]Q_2,\ P_2 + [b]Q_1 + [c]Q_2 \rangle \mid a, b, c \in \mathbb{Z}/\ell^n\mathbb{Z}\}\,.$$

In the terminology of the previous section this means that $\mathcal{K}_\ell^{\text{res}}$ is the set of all groups of Type 2.1 (cf. Definition 5.6 and Table 5.1).

First of all, note that every secret key $sk \in \mathcal{K}_\ell^{\text{res}}$ is indeed a maximal ℓ^n-isotropic subgroup as per Proposition 5.4. A very beneficial feature of the new keyspace is that every secret key $sk \in \mathcal{K}_\ell^{\text{res}}$ is uniquely encoded by a tuple $(a, b, c) \in (\mathbb{Z}/\ell^n\mathbb{Z})^3$. This means that a

secret key can be sampled by choosing three random integers $a, b, c \in \mathbb{Z}/\ell^n\mathbb{Z}$.

Moreover, the restricted keyspace still has the same order of magnitude as the original keyspace. To see this recall the number of maximal ℓ^n-isotropic subgroups from [FT19, Theorem 2]:

$$\#\mathcal{K}_\ell = \ell^{2n-3}(\ell^2+1)(\ell+1)\left(\ell^n + \frac{\ell^{n-1}-1}{\ell-1}\right) = \ell^{3n} \cdot \underbrace{\frac{(\ell^2+1)(\ell+1)}{\ell^3}\left(1+\frac{\ell^{n-1}-1}{\ell^n(\ell-1)}\right)}_{\alpha_\ell}.$$

For n which are large (as is the case in the cryptographic applications we consider), we find that $\alpha_2 \approx \frac{45}{16}$ and $\alpha_3 \approx \frac{140}{81}$.

As discussed in the context of (2.1), the keyspace of the SIDH protocol is restricted in a similar fashion for practical reasons. In the genus-2 setting, the restriction only marginally reduces the size of the keyspace: While $\mathcal{K}_\ell$ has cardinality $(\ell+1)\ell^{n-1}$, we effectively consider the ℓ^n different ℓ^n-isogenies from a fixed starting curve corresponding to the kernels of Type 2.1 in the restricted keyspace.

Remark 5.7. *It is also possible to develop a key generation algorithm which uniformly samples from the unrestricted keyspace. This would require a more thorough analysis of the proportions of the subgroup types corresponding to the different possible canonical generators in $\mathcal{K}_\ell$.*

We consider the case where $\ell = 2$. [FT19, Theorem 2] and [FT19, Proposition 3] give explicit formulae to compute the distribution of subgroups of rank two among all admissible subgroups is

$$\frac{2^n}{2^n + 2^{n-1} - 1} \approx \frac{2}{3}$$

for large n.

Performing the same computation on rank-3 subgroups, for large n we have

$$\frac{3 \cdot 2^{n-2k}}{3 \cdot 2^n - 2} \approx \frac{1}{2^{2k}},$$

where k is the parameter determining the subgroup structure.

Therefore, we obtain a method to almost uniformly sample the keyspace. First, $0 \leq k \leq N$ is determined for some bound $N \leq \lfloor \frac{n}{2} \rfloor$, weighted according to the proportion stated above. Next, one has to make a choice of canonical generators based on the distribution of the different types presented in Table 5.1. Finally, uniformly selecting the required scalars will ensure the near-uniform sampling from the keyspace.

5.2 Adaptive attack on G2SIDH

The attack as presented in this section is able to recover Alice's secret kernel when she uses a static secret kernel which is maximal 2^n-isotropic. In particular, we will describe a method that can recover secret kernels of various group structures. In the exposition to come, the scalars θ_i are used to ensure Weil pairing countermeasures are unable to detect our attack. This method is employed in tandem with the symplectic transformations that are primarily used to isolate the bit under attack. The adaptive attack on G2SIDH is similar to adaptive attacks on SIDH and some of its variants[3]. It interacts with an oracle by sending points on some starting variety that correspond to the auxiliary points provided in the protocol. The oracle is "weak" in the sense that only one bit is returned per query. By sending malformed points, the adaptive attack is able to recover scalars that determine the secret kernels.

The first step of the adaptive attack is to recover the kernel structure used by Alice, and a strategy for this is presented in Section 5.2.3. The next step then recovers the scalars

[3]See the description of the GPST attack in Section 2.1.2 or Section 5.4.2 and attacks on 2-SIDH and the Jao–Urbanik protocol in Chapter 4.

associated with the kernel structure determined in the first step and is divided into two parts depending on the rank of the kernel structure: Section 5.2.4 describes the technique for rank-2 kernels and Section 5.2.5 for kernel subgroups of rank 3. In each case, we will recover the first bit of the secret scalars before iteratively determining the remaining bits.

In the following, we will assume that all users of the G2SIDH protocol (or at least Alice, the honest party whose key we want to recover) are using a symplectic basis as described in Section 5.1.2. This attack will still work on users not using a symplectic basis as one can perform a linear transformation from an arbitrary torsion basis into a symplectic basis. For clarity, we present the attack directly on a symplectic torsion basis here and describe the extension to arbitrary bases in Section 5.3.

Notations and setup

Let us fix some notation. Let J be the starting variety, and let J_A be the codomain of the secret isogeny with kernel $\langle A_1, A_2, A_3\rangle$, where the orders of the points are 2^n, 2^{n-k}, 2^k respectively.

Furthermore, suppose $\langle P_1, P_2, Q_1, Q_2\rangle = J[2^n]$ is a symplectic basis such that $e_{2^n}(P_i, Q_j) = \zeta^{\delta_{ij}}$, where ζ is a primitive 2^n-th root of unity, and $e_{2^n}(P_1, P_2) = e_{2^n}(Q_1, Q_2) = \zeta^0 = 1$.

We write $\phi_B : J \to J_B$ for Bob's secret isogeny. Then $(\phi_B(P_1), \phi_B(P_2), \phi_B(Q_1), \phi_B(Q_2))$ is a symplectic basis for $J_B[2^n]$ as per Lemma 5.3. To ease notation, we set

$$R_1 = \phi_B(P_1),\ R_2 = \phi_B(P_2),\ S_1 = \phi_B(Q_1),\ S_2 = \phi_B(Q_2).$$

We will assume that Alice is the party under attack, and that she is using twelve secret scalars $\alpha_{1,1}, \ldots, \alpha_{3,4}$ which define a maximal 2^n-isotropic subgroup of $J[2^n]$ as was suggested in [FT19]. We can write any of the secret scalars, say a, as $a = \sum_{i=0}^{n-1} 2^i a_i$ for bits $a_i \in \{0, 1\}$.

For $i = 1, \ldots, n-1$, let us then denote the partial key consisting of the first i bits of a as $K_i^a = \sum_{j=0}^{i-1} 2^j a_j$ so that $a = K_i^a + 2^i a_i + 2^{i+1} a'$ for some a'. This convention will help us keep track of the known information at each step of the attack below.

5.2.1 Attack model and oracle

The attack we present in the following assumes that an honest Alice uses a static key which a malicious Bob is trying to learn through repeatedly providing malformed torsion point information during the G2SIDH protocol execution. Bob's overall goal is to recover Alice's full key or a valid tuple of scalars forming an equivalent key which corresponds to the same sequence of Richelot isogenies.

It is customary in similar attacks to consider two distinct oracles which can model the information obtained by the attacker which differ in their inherent strength. On input of a variety J and four points $R_1', \ldots, R_4' \in J[2^n]$, one provides the isomorphism invariants of the codomain variety J/G_A of the isogeny corresponding to the kernel subgroup $G_A = \langle \sum_{i=1}^4 [\alpha_{1,i}] R_i', \sum_{i=1}^4 [\alpha_{2,i}] R_i', \sum_{i=1}^4 [\alpha_{3,i}] R_i' \rangle$. The second, less powerful oracle is the one we will utilise to model our attack in the following, as is done in [GPST16].

Our oracle, which replaces Alice in an honest execution of the protocol,

$$O\left(J, (R_1', R_2', R_3', R_4'), J'\right)$$

returns 1 whenever the subgroup $G_A = \langle \sum_{i=1}^4 [\alpha_{1,i}] R_i', \sum_{i=1}^4 [\alpha_{2,i}] R_i', \sum_{i=1}^4 [\alpha_{3,i}] R_i' \rangle$ is isotropic and the variety J/G_A has the same isomorphism invariants as the second input variety J'. Otherwise, it returns 0. Moreover we assume that the oracle checks whether an input is valid and returns $\perp$ if this is not the case. Here, we say that a tuple $(J, (R_1', R_2', S_1', S_2'), J')$ is a *valid* input if (R_1', R_2', S_1', S_2') is a symplectic basis for $J[2^n]$ and $e_{2^n}(R_i', S_i') = e_{2^n}(P_i, Q_i)^{3^{e_B}}$. Note that an honest run of the protocol generates the valid input $(J_B, (R_1, R_2, S_1, S_2), J_{AB})$.

For ease of reading, we will represent malformed points to be queried as linear combinations of R_1, R_2, S_1, S_2 and laid out in a 4×4 matrix. That is, for any points R_1', R_2', S_1', S_2' that the adversary sends to the oracle, we can write

$$\begin{pmatrix} R_1' \\ R_2' \\ S_1' \\ S_2' \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \\ S_1 \\ S_2 \end{pmatrix},$$

and we will represent the queries R_1', R_2', S_1', S_2' by the 4×4 matrix.

Remark 5.8. *In comparison to attacks on elliptic curve SIDH and its variants, the genus-2 setting would allow us to consider another type of oracle distinguishing between different cases of 0 outputs for O above. More precisely, this oracle would specify whether it returns 0 because the subgroup generated by the malformed input points is not a maximal 2^n-isotropic subgroup of $J[2^n]$ (and thus does not specify a variety isogenous to J with this kernel subgroup), or because the given subgroup is isotropic but yields a variety different from the reference variety J.*

However, the additional information learned from such an oracle does not always lead to a reduction in the number of necessary queries made by an attacker. For example, the simultaneous recovery of a bit each of the scalars b and c when the secret kernel is of rank 3 as described in Section 5.2.5 requires three queries with the oracle O. This number could be reduced to two queries with the distinguishing oracle whenever the scalar d is odd. This only happens with probability $1/2$, so we restrict to using the oracle O so that our strategy works for arbitrary scalars.

5.2.2 Symplectic transformations

When constructing malformed torsion points for the oracle queries, we need to make sure that the input is still valid. In our setting, an oracle query

$$O\left(J_B, (R'_1, R'_2, S'_1, S'_2), J_{AB}\right)$$

is valid if and only if (R'_1, R'_2, S'_1, S'_2) is a symplectic basis and

$$e_{2^n}(R'_i, S'_j) = e_{2^n}(R_i, S_j) \quad \text{for } i, j \in \{1, 2\}.$$

A change of basis $t : (R'_1, R'_2, S'_1, S'_2) \leftarrow (R_1, R_2, S_1, S_2)$ with this property is called a *symplectic transformation*. The matrices corresponding to symplectic transformations are called *symplectic matrices*. We are going to write M_t for the matrix corresponding to the transformation t.

Using symplectic transformations has yet another advantage. Let $G = \langle G_1, G_2, G_3 \rangle \subset J$ be maximal 2^n-isotropic and $t : J[2^n] \to J[2^n]$ a symplectic transformation, then $G' = \langle t(G_1), t(G_2), t(G_3) \rangle$ is maximal 2^n-isotropic as well. Note that this is not true for general isomorphisms of $J[2^n]$.

One can easily verify that the following matrices are symplectic. We will use different combinations of these to construct the transformations for the oracle queries.

$$M_{t_0} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; M_{t_1} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; M_{t_2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix},$$

$$M_{t_3} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; M_{t_4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, \; M_{t_5} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Proposition 5.9. *The following matrices are symplectic for any values $x, x_0, x_1, x_2, x_3, x_4, x_5$ and ivertible elements $\theta_1, \theta_2 \in \mathbb{Z}/2^n\mathbb{Z}$.*

$$M_1 = \begin{pmatrix} \theta_1 & \theta_2 x & 0 & 0 \\ 0 & \theta_2 & 0 & 0 \\ 0 & 0 & \theta_1^{-1} & 0 \\ 0 & 0 & -\theta_1^{-1}x & \theta_2^{-1} \end{pmatrix}, \qquad M_2 = \begin{pmatrix} 1 & 0 & x_1 & x_5 \\ 0 & 1 & x_5 & x_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$M_3 = \begin{pmatrix} \theta_1(1 + x_0x_1 - x_4x_5(1 + x_0x_1)) & \theta_2x_2x_5 & \theta_1^{-1}x_1(1 + x_4x_5) & \theta_2^{-1}x_5 \\ \theta_1x_0x_5 & \theta_2(1 + x_2x_3 + x_4x_5(1 + x_2x_3)) & \theta_1^{-1}x_5 & \theta_2^{-1}x_3(1 + x_4x_5) \\ \theta_1x_0 & \theta_2x_4(1 + x_2x_3) & \theta_1^{-1} & \theta_2^{-1}x_3x_4 \\ \theta_1x_4(1 + x_0x_1) & \theta_2x_2 & \theta_1^{-1}x_1x_4 & \theta_2^{-1} \end{pmatrix}.$$

Proof. It is easy to check that M_1 is symplectic since the scalars satisfy $\theta_1\theta_1^{-1} = \theta_2\theta_2^{-1} = 1$. The matrix M_2 can be easily written in terms of the transformations t_i, namely $M_2 = M_{t_1}^{x_1} \cdot M_{t_3}^{x_3} \cdot M_{t_5}^{x_5}$. Finally, M_3 can be written as

$$M_3 = M_1 \cdot M_{t_0}^{x_0} \cdot M_{t_1}^{x_1} \cdot M_{t_2}^{x_2} \cdot M_{t_3}^{x_3} \cdot M_{t_4}^{x_4} \cdot M_{t_5}^{x_5}.$$

□

All our queries to the oracle are obtained by combining the transformations in the proposition above. In order to choose a transformation, it is necessary to examine the effect of a transformation on a secret subgroup. To illustrate this, assume that Alice uses a group $\langle A_1, A_2, A_3\rangle$ of Type 3.1. This means $A_1 = R_1 + [d]R_2 + [a]S_1 + [b]S_2$, $A_2 = 2^k(R_2 + [b-cd]S_1 + [c]S_2)$ $A_3 = 2^{n-k}([-d]S_1 + S_2)$. As in Section 5.1.3, we let A be the associated matrix, i.e. here

$$A = \begin{pmatrix} 1 & d & a & b \\ 0 & 2^k & 2^k(b-cd) & 2^k c \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix}.$$

Applying a basis transformation t corresponds to computing $A' = A \cdot M_t$. As an example, consider the second basis transformation from the above proposition.

$$A' = A \cdot M_2 = \begin{pmatrix} 1 & d & a + x_1 + dx_5 & b + x_5 + dx_3 \\ 0 & 2^k & 2^k(b-cd) + 2^k x_5 & 2^k c + 2^k x_3 \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix} = A + \begin{pmatrix} 0 & 0 & x_1 + dx_5 & x_5 + dx_3 \\ 0 & 0 & 2^k x_5 & 2^k x_3 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

This means that the matrices A and A' correspond to the same group G_A if and only if $\langle [x_1 + dx_5]S_1 + [x_5 + dx_3]S_2, [2^k]([x_5]S_1 + [x_3]S_2)\rangle \subset G_A$.

5.2.3 Case distinction of kernel subgroups

Recall that in [FT19], Alice's secret can be described by $(\alpha_{1,1}, \ldots, \alpha_{3,4})$. A priori we do not know, if the group G_A defined by these scalars has rank 2 or 3. Moreover we do not know which canonical form is obtained when normalising the generators (cf. Definition 5.6, Table 5.1). In total, when $k = 0$ there are 4 types of maximal 2^n-isotropic groups, 8 different

types when $0 < k < \frac{n}{2}$ and 4 different types when $k = \frac{n}{2}$. The type can be recovered by sending at most $4k+4$ queries that mimic the normalisation process outlined in Section 5.1.3. The approach is illustrated in the decision tree in Fig. 5.2, where each node is labelled with the condition we want to test for. Note that at most two queries have to be made per "equivalence" node while at most four queries are necessary to test for divisibility by a power of 2. We provide details for one of the paths in the decision tree below.

Assuming that the key $(\alpha_{1,1}, \ldots, \alpha_{3,4})$ is drawn uniformly at random from the entire key space $\mathcal{K}_2$, the algorithm illustrated by the decision tree will in many cases terminate at an early stage: Recall from Section 5.1.4 that roughly one third of the key space consists of groups of Type 2.1. In that case the algorithm terminates after three queries; one to find that one of $\alpha_{1,1}, \alpha_{2,1}, \alpha_{3,1}$ is odd, and another two to determine that one of $\alpha_{2,2}, \alpha_{3,2}$ is odd. In total, the rank-2 subgroups constitute two thirds of the key space, in which case the algorithm terminates after having made at most six queries. Finally, if we encounter a rank-3 group, it will usually not be necessary to perform many iterations to find k because the probability that $k > k_0$ for some fixed k_0 is less than $\frac{1}{3 \cdot 2^{2k_0}}$.

Observe that an attacker obtains some information about the value of certain bits during the course of the type distinction. In particular for rank-3 groups, we recover normalised scalars $b \pmod{2^k}$ and $d \pmod{2^k} = d$ via the iterative queries. At each step of the iteration, we aim to find out whether 2^{k_0+1} divides the coefficients of $P_{\sigma^{-1}(2)}$ and $P_{\sigma^{-1}(4)}$ in the canonical generators of $\langle A_2 \rangle$ and $\langle A_3 \rangle$. In order to achieve this, we need to eliminate the possibility that an oracle query returns 0 because $\langle A_1' \rangle \neq \langle A_1 \rangle$. Hence, we need to query twice for each possible further bit of the coefficients of $P_{\sigma^{-1}(2)}$ and $P_{\sigma^{-1}(4)}$ in $\langle A_1 \rangle$. Therefore we recover the first k bits of b and d fully while we determine the type of G_A. This information can then be used to drastically reduce the number of queries in the main attack algorithm presented in Section 5.2.5, and we thus assume knowledge of $b \pmod{2^k}$ and d for any rank-3 kernel subgroups.

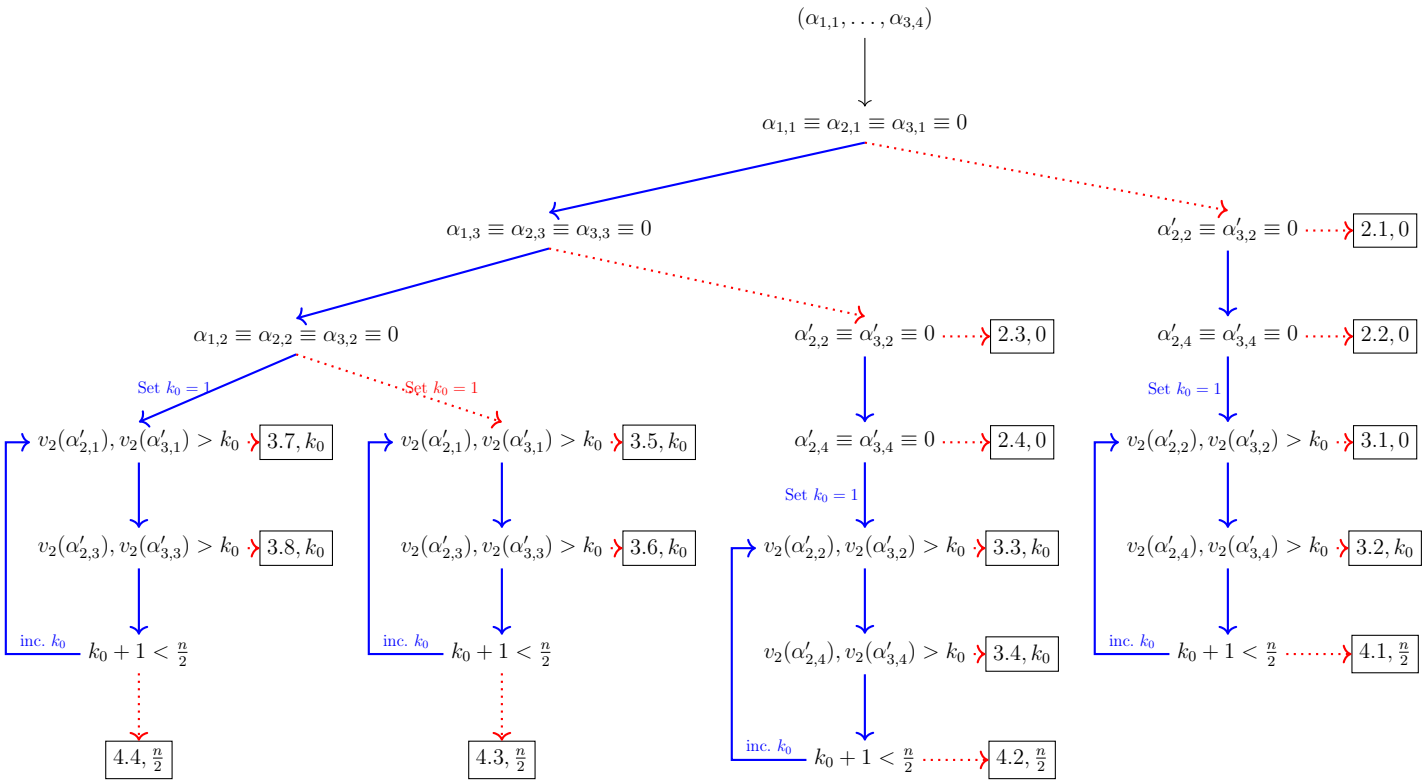

Figure 5.2: Strategy for type distinction of normalised kernel generators as in Table 5.1.

Annotations to Fig. 5.2 We begin with Alice's scalars $(\alpha_{1,1}, \ldots, \alpha_{3,4})$. Each node below represents one or multiple malformed queries which determine whether the displayed condition holds. All equivalence conditions are viewed modulo 2 here. For example, the first query node corresponds to checking whether $\alpha_{1,1} \equiv \alpha_{2,1} \equiv \alpha_{3,1} \equiv 0 \pmod 2$ which can be done with the transformation ${t_1}^{2^{n-1}}$. At each node, a true response indicates that the next query can be found along the blue and solid arrow, while the red and dotted path is taken when the condition is not fulfilled. Note that when an odd scalar is found, the subsequent conditions use further normalised scalars denoted by $\alpha'_{i,j}$. Leaves show which type of normalised generators define the secret subgroup Alice uses (as classified in Table 5.1), followed by k which indicates the order of the generators of the subgroup. For distinguishing types of rank-3 subgroups, it is necessary to use iterative queries to find the correct type and determine the value of k. At each step, we test whether $k = k_0$ for increasing values

of $0 < k_0 < \frac{n}{2} - 1$ by checking if certain scalars are divisible by 2^{k_0+1}. We use that for any integer x, $v_2(x)$ denotes the largest integer such that $2^{v_2(x)}$ divides x. If a scalar is found to not satisfy the divisibility condition, we can again normalise at this position and deduce the type of the subgroup along with k indicating the order of its generators.

We now provide an example that illustrates how the decision tree in Fig. 5.2 can be used to determine the canonical form of the normalised generator for an admissible kernel subgroup (cf. Definition 5.6 and Table 5.1) by presenting the queries along one path which allow us to classify one type of rank-3 subgroup. The queries required to check the conditions along other paths are very similar and hence omitted.

Suppose Alice's secret is of the form $(\alpha_{1,1}, \ldots, \alpha_{3,4})$ and let A be the 3×4 matrix defined by these scalars. We do not initially know the rank of the group G_A defined by these scalars, which canonical form is obtained when normalising the generators, nor their respective order. We proceed as follows to find the type of G_A according to our classification from Section 5.1.3.

Step 1 We start by testing whether $\alpha_{1,1} \equiv \alpha_{2,1} \equiv \alpha_{3,1} \equiv 0 \pmod 2$ with the query (R'_1, R'_2, S'_1, S'_2) obtained from the transformation

$$M_{t_1}{}^{2^{n-1}} = \begin{pmatrix} 1 & 0 & 2^{n-1} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Applied to A this transformation yields

$$A' = A \cdot M_t = A + 2^{n-1} \cdot \begin{pmatrix} 0 & 0 & \alpha_{1,1} & 0 \\ 0 & 0 & \alpha_{2,1} & 0 \\ 0 & 0 & \alpha_{3,1} & 0 \end{pmatrix}.$$

The matrices A and A' define the same group G_A if and only if

$$[2^{n-1}][\alpha_{1,1}]S_1,\ [2^{n-1}][\alpha_{2,1}]S_1,\ [2^{n-1}][\alpha_{3,1}]S_1 \in G_A.$$

This is the case if and only if $\alpha_{1,1}, \alpha_{2,1}$ and $\alpha_{3,1}$ are all even. One direction is easy. For the other direction, note that if $[2^{n-1}][\alpha_{j,1}]S_1 \in G_A$, then isotropy implies $1 = e_{2^n}([\alpha_{i,1}]R_1 + [\alpha_{i,2}]R_2 + [\alpha_{i,3}]S_1 + [\alpha_{i,4}]S_2,\ [2^{n-1}\alpha_{j,1}]S_1) = e_{2^n}(R_1, S_1)^{2^{n-1}\alpha_{i,1}\cdot\alpha_{j,1}}$ for all $i \in \{1, 2, 3\}$. This leads to the first case distinction depending on the output of the oracle which signifies whether the permutation $\sigma \in D_8$ corresponding to the normalisation of G_A fixes the first basis point:

$$O\left(J_B, (R_1', R_2', S_1', S_2'), J_{AB}\right) = \begin{cases} 1 & : \text{Types } \ 2.3, 2.4, 3.3 - 3.8, 4.2 - 4.4. \\ 0 & : \text{Types } \ 2.1, 2.2, 3.1, 3.2, 4.1. \end{cases}$$

Assume that the answer is 0. This implies that at least one of the coefficients of R_1 is invertible and we can perform a first normalisation step. We obtain elements $\alpha_{i,2}', \alpha_{i,3}', \alpha_{i,4}'$ for $i \in \{1, 2, 3\}$ such that

$$A \sim \begin{pmatrix} 1 & \alpha_{1,2}' & \alpha_{1,3}' & \alpha_{1,4}' \\ 0 & \alpha_{2,2}' & \alpha_{2,3}' & \alpha_{2,4}' \\ 0 & \alpha_{3,2}' & \alpha_{3,3}' & \alpha_{3,4}' \end{pmatrix}.$$

Step 2 Now we test whether one of $\alpha'_{2,2}$ or $\alpha'_{3,2}$ is invertible. This requires at most two queries. First, we send the basis obtained from the transformation

$$M_{t_3}^{2^{n-1}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2^{n-1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Similarly to the strategy above, $O(J, J', (R'_1, R'_2, S'_1, S'_2)) = 1$ if and only if all of $\alpha'_{1,2}, \alpha'_{2,2}, \alpha'_{3,2}$ are even. On the other hand if $O(J, J', (R'_1, R'_2, S'_1, S'_2)) = 0$, we only know that at least one of the three coefficients of R_2 is odd. We want to distinguish between the two cases where $\alpha'_{1,2}$ is odd and both $\alpha'_{2,2}, \alpha'_{3,2}$ are even, or where at least one of $\alpha'_{2,2}, \alpha'_{3,2}$ is odd. Therefore, we also send the query obtained from

$$M_{t_1}^{2^{n-1}} M_{t_3}^{2^{n-1}} M_{t_5}^{2^{n-1}} = \begin{pmatrix} 1 & 0 & 2^{n-1} & 2^{n-1} \\ 0 & 1 & 2^{n-1} & 2^{n-1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Taking into account that the output of the previous query was 0, the answer of this query is 1 if $\alpha'_{1,2}$ is odd and both $\alpha'_{2,2}, \alpha'_{3,2}$ are even, and it is 0 otherwise.

Note that we are done with the case distinction if both of the previous queries returned

0. In that case we can simply normalise the coefficient of R_2 to 1 and find that

$$A \sim \begin{pmatrix} 1 & 0 & \alpha''_{1,3} & \alpha''_{1,4} \\ 0 & 1 & \alpha''_{2,3} & \alpha''_{2,4} \\ 0 & 0 & \alpha''_{3,3} & \alpha''_{3,4} \end{pmatrix}.$$

Using the fact that the group G_A is isotropic, we see that $\alpha''_{3,3} = \alpha''_{3,4} = 0$, and G_A is a rank-2 group of Type 2.1.

On the other hand if one of the queries returned 1, then none of $\alpha'_{2,2}, \alpha'_{3,2}$ are invertible. This leaves the following possibilities for the group structure.

$$2.2 \quad \begin{pmatrix} 1 & b & a & 0 \\ 0 & c & -b & 1 \end{pmatrix}$$

where c is even, and the three rank-3 types

$$3.1 \quad \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & \ell^k(b-cd) & \ell^k c \\ 0 & 0 & -\ell^{n-k}d & \ell^{n-k} \end{pmatrix}, \quad 3.2 \quad \begin{pmatrix} 1 & b & a & d \\ 0 & \ell^k c & \ell^k(cd-b) & \ell^k \\ 0 & \ell^{n-k} & \ell^{n-k}d & 0 \end{pmatrix}, \quad 4.1 \quad \begin{pmatrix} 1 & d & a & b \\ 0 & \ell^k & \ell^k b & 0 \\ 0 & 0 & -\ell^k d & \ell^k \end{pmatrix}.$$

Note that we can also deduce the parity of d (resp. b) for Type 3.1 and 4.1 (resp. Type 3.2) from the previous queries.

Step 3 In this step we distinguish between Type 2.2 and the possible rank-3 types. For that purpose, we check whether one of $\alpha'_{2,4}$ or $\alpha'_{3,4}$ is invertible using the transformations

$$M_{t_2}^{2^{n-1}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2^{n-1} & 0 & 1 \end{pmatrix} \quad \text{and} \quad M_{t_4} \cdot M_{t_1}^{2^{n-1}} \cdot M_{t_4}^{-1} = \begin{pmatrix} 1 & 2^{n-1} & 2^{n-1} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2^{n-1} & 2^{n-1} & 1 \end{pmatrix}.$$

If the queries show that one (or both) of $\alpha'_{2,4}$ or $\alpha'_{3,4}$ is invertible, then G_A has Type 2.2. Otherwise, if both $\alpha'_{2,4}$ and $\alpha'_{3,4}$ are even we know that G_A is a rank-3 group, and we continue with the next step.

Step 4 In order to distinguish between Types 3.1, 3.2 and Type 4.1, we have to compare the elements $\alpha'_{2,2}, \alpha'_{3,2}$ and $\alpha'_{2,4}, \alpha'_{3,4}$. Recall that all of these scalars are necessarily even, hence we can find positive integers $k_{2,2}, k_{2,4}, k_{3,2}, k_{3,4}$ and odd numbers $\beta_{2,2}, \beta_{2,4}, \beta_{3,2}, \beta_{3,4}$ such that $\alpha'_{i,j} = 2^{k_{i,j}}\beta_{i,j}$ for $(i,j) \in I = \{(2,2),(2,4),(3,2),(3,4)\}$. Our goal is to determine

$$k = \min\{k_{i,j} \mid (i,j) \in I\}.$$

This minimum can be found iteratively. We start with $k_0 = 1$ and increase k_0 by one if the following queries are not successful. Before describing the queries, note that

$$2^{n-k}\langle R_2 + [\alpha'_{1,4}]S_1,\ S_2 - [\alpha'_{1,2}]S_1\rangle \subset G_A \tag{5.3}$$

as per Corollary 5.5. Property (5.3) will be used multiple times in this step.

The first query is to test whether G_A is of Type 3.1 with $k = k_0$. Recall that we know the parity of $\alpha'_{1,2}$ and $\alpha'_{1,4}$ from the previous queries. For each iterative step, we have determined $\alpha'_{1,2} \pmod{2^{k_0}}$ and $\alpha'_{1,4} \pmod{2^{k_0}}$ from the previous queries. We write $\alpha'_{1,2} = K_{\alpha'_{1,2}} + 2^{k_0}\alpha'_{1,2,k_0} + 2^{k_0+1}\alpha''_{1,2}$ and $\alpha'_{1,4} = K_{\alpha'_{1,4}} + 2^{k_0}\alpha'_{1,4,k_0} + 2^{k_0+1}\alpha''_{1,4}$. We send the

following query

$$(M_{t_3} {M_{t_1}}^{x_i} {M_{t_5}}^{y_i})^{2^{n-2k_0-1}} = \begin{pmatrix} 1 & 0 & 2^{n-2k_0-1}x_i & 2^{n-2k_0-1}y_i \\ 0 & 1 & 2^{n-2k_0-1}y_i & 2^{n-2k_0-1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

first with $y_1 = -K_{\alpha'_{1,2}}$ and $x_1 = y_1^2$. This transformation leaves the kernel unchanged if and only if

$$\begin{aligned} 2^{n-k_0-1}\alpha_{1,2,k_0}([-\alpha_{1,2}]S_1 + S_2) &\in G_A, \\ 2^{n-2k_0-1}\alpha_{2,2}([-\alpha_{1,2}]S_1 + S_2) &\in G_A, \\ 2^{n-2k_0-1}\alpha_{3,2}([-\alpha_{1,2}]S_1 + S_2) &\in G_A. \end{aligned}$$

Using Property (5.3), this translates to the conditions $\alpha'_{1,2,k_0} = 0$ and 2^{k_0+1} divides $\alpha'_{2,2}$ and $\alpha'_{3,2}$, i.e. $k_{2,2} > k_0$ and $k_{3,2} > k_0$. If the oracle query returns 0, we resend the query with $y_2 = -(K_{\alpha'_{1,2}} + 2^{k_0})$ and $x_2 = y_2^2$. If these malformed points produce a group distinct from G_A, we can deduce that one of $k_{2,2}, k_{3,2}$ equals k_0. Hence, the group must be of Type 3.1 with $k = k_0$. Otherwise, $O\left(J, (R'_1, R'_2, S'_1, S'_2), J'\right) = 1$ implies that we have determined the correct next bit of $\alpha'_{1,2}$ and that $k_{2,2} > k_0$ and $k_{3,2} > k_0$. We thus proceed with the next set of queries to test whether the group is of Type 3.2. We send the two queries corresponding to the transformation $M_t = M_{t_1}^{2^{n-2k_0-1}x_i} M_{t_2}^{2^{n-2k_0-1}} M_{t_2}^{(2^{n-2k_0-1}y_i)^2} M_{t_1}^{-1} M_{t_4}^{2^{n-2k_0-1}y_i} M_{t_1} M_{t_4}^{-2^{n-2k_0-1}y_i}$ such that

$$M_t = \begin{pmatrix} 1 & 2^{n-2k_0-1}y_i & 2^{n-2k_0-1}x_i & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2^{n-2k_0-1} & -2^{n-2k_0-1}y_i & 1 \end{pmatrix}$$

with $y_1 = -K_{\alpha'_{1,4}}$, $y_2 = -(K_{\alpha'_{1,4}} + 2^{k_0})$ and $x_i = -y_i^2$. With analogous reasoning as before, we can deduce from the oracle responses whether both $k_{2,4} > k_0$ and $k_{3,4} > k_0$, and learn the next bit of $\alpha'_{1,4}$ if any query returns 1. Thus we can either determine the type of G_A to be 3.2 with $k = k_0$, or increase k_0 by 1 and repeat the queries in this step.

If we have not managed to determine that G_A is of Type 3.1 or 3.2 with $k_0 < \frac{n}{2}$, we conclude that indeed G_A must be of Type 4.1 with $k = \frac{n}{2}$.

5.2.4 Recovering kernels of rank 2

As discussed above, there are multiple canonical forms for rank-2 kernels. In this section, we assume that we have applied the case distinction method from above to find the correct canonical form of the kernel generators. We illustrate the attack for Type 2.1, where

$$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a & b \\ 0 & 1 & b & c \end{pmatrix} \cdot \begin{pmatrix} R_1 \\ R_2 \\ S_1 \\ S_2 \end{pmatrix}.$$

Should the generators be of a different canonical form, slight alterations to the malformed points in the exposition of the attack below will suffice to still recover the correct scalars.

Parity bits We want to employ symplectic transformations so that the Weil pairing countermeasure is unable to detect that malformed points have been sent. Table 5.2 presents transformations that return information about the parity bits, and Fig. 5.3 illustrates how one can use the transformations to get an optimal adaptive attack.

transformation	same j-invariant iff
${t_0}^{2^{n-1}}$	$a \equiv b \equiv 0$
${t_2}^{2^{n-1}}$	$b \equiv c \equiv 0$
${t_4}^{2^{n-1}}$	$b \equiv ac$
${t_0}^{2^{n-1}}{t_1}^{2^{n-1}}$	$a + 1 \equiv b \equiv 0$
${t_0}^{2^{n-1}}{t_3}^{2^{n-1}}$	$a \equiv b + 1 \equiv 0$
${t_2}^{2^{n-1}}{t_1}^{2^{n-1}}$	$b + 1 \equiv c \equiv 0$
${t_2}^{2^{n-1}}{t_3}^{2^{n-1}}$	$b \equiv c + 1 \equiv 0$

Table 5.2: Table of symplectic transformations and how parity bits affect the codomain. The equivalences in the second column are all modulo 2.

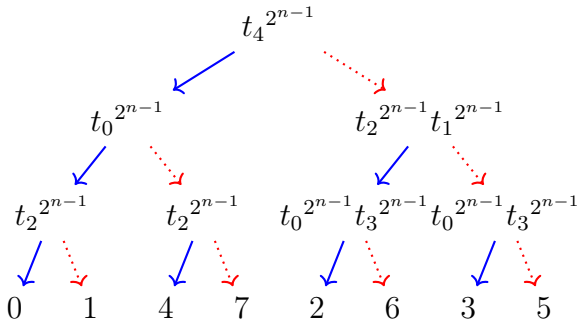

Figure 5.3: Optimal strategy for recovering parity bits. The top node represents the first malformed query which will use the t_4 transformation to determine whether $b \equiv ac \pmod 2$, as shown in Table 5.2. A true response indicates that the next query can be found along the blue and solid arrow, while the red and dotted path is taken when the condition is false. Leaves are decimal representations of the parity bits a_0, b_0, c_0, i.e. 6 corresponds to $[a_0, b_0, c_0] = [1, 1, 0]$.

As an example, we examine how the first transformation, ${t_4}^{2^{n-1}}$, affects the kernel

generators. This step corresponds to sending malformed points obtained via the matrix

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2^{n-1} & 1 & 0 \\ 2^{n-1} & 0 & 0 & 1 \end{pmatrix}$$

and leads to Alice using

$$A' = \begin{pmatrix} 1 & 0 & a & b \\ 0 & 1 & b & c \end{pmatrix} \cdot M = \begin{pmatrix} 1+2^{n-1}b & 2^{n-1}a & a & b \\ 2^{n-1}c & 1+2^{n-1}b & b & c \end{pmatrix} \sim A + \begin{pmatrix} 0 & 0 & 0 & 2^{n-1}(b^2+ac) \\ 0 & 0 & 2^{n-1}(b^2+ac) & 0. \end{pmatrix}$$

during her internal computations. Note that in the last step, Gaussian elimination is used to normalise A'. We can observe that $O\left(J_B, (R'_1, R'_2, S'_1, S'_2), J_{AB}\right) = 1$ if and only if $2^{n-1}(b^2 + ac) \equiv 0 \pmod{2^n}$. This occurs whenever $b \equiv ac \pmod 2$, as displayed in Table 5.2, and from the response we can determine whether $[a_0, b_0, c_0]$ is among $\{[0,0,0],[0,0,1],[1,0,0],[1,1,1]\}$ or $\{[0,1,0],[0,1,1],[1,0,1],[1,1,0]\}$.

Iterative step The recovery of subsequent bits will not follow the optimal strategy from the recovery of the parity bits. However, it will still recover one bit of information on average per query.

Suppose now that we have learned the first i bits of each key scalar. Then we know K_i^a, K_i^b and K_i^c, where $a = \sum_{j=0}^{n-1} 2^j a_j = K_i^a + \sum_{j=i}^{n-1} 2^j a_j$ (and similarly for b and c).

Now assume that $i < n-3$ and set $e_i := n - i - 1$. By the lemma below, the element $T_i = 1 - 2^{e_i}$ is thus a quadratic residue modulo 2^n.

Lemma 5.10 ([GPST16, Lemma 4]). *Let $n \geq 5$ and $i \in \{1, \ldots, n-4\}$. Then $T_i := 1-2^{n-i-1}$ is a quadratic residue modulo 2^n.*

In the following, $\theta_i \in \{0, \ldots, 2^n - 1\}$ denotes one of the square roots of T_i, i.e. $\theta_i^2 \equiv T_i \pmod{2^n}$. Note that θ_i is necessarily odd, hence there exists an inverse θ_i^{-1} modulo 2^n. Intuitively, θ_i is a masking scalar that allows us to defeat the Weil pairing countermeasure.

We use three different sets of malformed points to determine a_i and c_i, and then learn b_i with one further query.

First, we send the malformed points obtained from

$$\theta_i^{-1} \begin{pmatrix} T_i & 0 & -2^{e_i}K_i^a & 0 \\ 0 & 1 & 0 & 2^{e_i}K_i^c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & T_i \end{pmatrix}.$$

Upon which, the subgroup computation[4] will entail

$$A' = \theta^{-1} \begin{pmatrix} T_i & 0 & -2^{e_i}K_i^a + a & T_i b \\ 0 & 1 & b & 2^{e_i}K_i^c + T_i c \end{pmatrix}$$
$$\sim \begin{pmatrix} 1 & 0 & a + T_i^{-1}2^{e_i}(K_i^a - a) & b \\ 0 & 1 & b & c + 2^{e_i}(K_i^c - c) \end{pmatrix} = A + \begin{pmatrix} 0 & 0 & T_i^{-1}2^{n-1}a_i & 0 \\ 0 & 0 & 0 & 2^{n-1}c_i \end{pmatrix}.$$

Hence A' defines the same group as A, exactly when both a_i and c_i are zero.

If we have not yet recovered the two bits in question, we proceed with sending mal-

[4] To verify the computation below note that $T_i^{-1} \equiv 1 + 2^{e_i}T_i^{-1} \pmod{2^n}$.

formed points corresponding to the transformation matrix

$$\theta_i^{-1}\begin{pmatrix} T_i & 0 & -2^{e_i}(K_i^a + 2^i) & 0 \\ 0 & 1 & 0 & 2^{e_i}K_i^c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & T_i \end{pmatrix}.$$

In this case

$$A' \sim A + \begin{pmatrix} 0 & 0 & T_i^{-1}2^{n-1}(a_i+1) & 0 \\ 0 & 0 & 0 & 2^{n-1}c_i \end{pmatrix}.$$

The groups associated to A and A' coincide exactly when $a_i = 1$ and $c_i = 0$.

If both queries fail to recover the bits a_i and c_i, i.e. $(a_i, c_i) \notin \{(0,0),(1,0)\}$, then we can conclude that $c_i = 1$. To find the bit a_i, we then send the third set of malformed points obtained from

$$\theta_i^{-1}\begin{pmatrix} T_i & 0 & -2^{e_i}K_i^a & 0 \\ 0 & 1 & 0 & 2^{e_i}K_{i+1}^c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & T_i \end{pmatrix}.$$

Here, the oracle will return 1 exactly when $a_i = 0$. If this is not the case, then $a_i = 1$.

After these series of queries, we have recovered the bits a_i and c_i, hence we know K_{i+1}^a and K_{i+1}^c. It remains to recover the bit b_i. This is done by querying the oracle on the points

corresponding to the matrix

$$\theta_i^{-1} \begin{pmatrix} 1 & 0 & 2^{e_i} K_{i+1}^a & 2^{e_i} K_i^b \\ 0 & 1 & 2^{e_i} K_i^b & 2^{e_i} K_{i+1}^c \\ 0 & 0 & T_i & 0 \\ 0 & 0 & 0 & T_i \end{pmatrix}.$$

Here,

$$A' \sim A + \begin{pmatrix} 0 & 0 & 2^{e_i}(K_{i+1}^a - a) & 2^{e_i}(K_i^b - b) \\ 0 & 0 & 2^{e_i}(K_i^b - b) & 2^{e_i}(K_{i+1}^c - c) \end{pmatrix} = A + \begin{pmatrix} 0 & 0 & 0 & 2^{n-1} b_i \\ 0 & 0 & 2^{n-1} b_i & 0 \end{pmatrix}.$$

The oracle returns 1 exactly when $b_i = 0$. Otherwise, we know that $b_i = 1$.

It follows from Theorem 5.9 that all of the transformations used in these queries are symplectic. Therefore they constitute valid queries in our oracle model. As a consequence the attack is not detectable by the Weil pairing.

Note that we are not able to use these transformation for $i \in \{n-3, n-2, n-1\}$. We suggest to use a brute force method to deduce the last three bits of each scalar. This is consistent with the adaptive attack described in [GPST16].

5.2.5 Recovering kernels of rank 3

Now suppose Alice's secret kernel subgroup has rank 3, i.e. $k > 0$. Let $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$ be fixed. We assume that the attacker has determined the type of Alice's secret subgroup as outlined in Section 5.2.3, and therefore knows k. We present the attack for a kernel of Type

3.1, hence the generators are of the form

$$\begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} = \begin{pmatrix} 1 & d & a & b \\ 0 & 2^k & 2^k(b-cd) & c \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix} \cdot \begin{pmatrix} R_1 \\ R_2 \\ S_1 \\ S_2 \end{pmatrix}$$

for some $(a, b, c, d) \in \{0, \ldots, 2^n - 1\} \times \{0, \ldots, 2^{n-k} - 1\} \times \{0, \ldots, 2^{n-2k} - 1\} \times \{0, \ldots, 2^k - 1\}$, where $b \pmod{2^k}$ and d are known from the case distinction algorithm. As usual, we denote the resulting variety $J_B/\langle A_1, A_2, A_3\rangle$ by J_{AB}.

We again fix

$$e_i = n - i - 1, \quad T_i = 1 - 2^{e_i} \in \mathbb{Z}/\ell^n\mathbb{Z}, \quad \theta_i^2 = T_i \in \mathbb{Z}/\ell^n\mathbb{Z}$$

for $1 \leq i \leq n - 4$, where θ_i is any one of the two square roots. Recall that θ_i exists since $T_i \equiv 1 \pmod 8$.

Recovering $a \pmod{2^{k-1}}$ We first recover the parity of the secret scalar a by sending the malformed points obtained from the transformation matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2^{n-1} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

These allow us to recover the bit a_0 since

$$A' = \begin{pmatrix} 1+2^{n-1}a & d & a & b \\ 0 & 2^k & 2^k(b-cd) & 2^k c \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix} \sim A + \begin{pmatrix} 0 & 0 & 2^{n-1}a^2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

This means that A and A' correspond to the same group if and only if $a_0 = 0$, hence we can deduce a_0 from the oracle response.

Now, we iteratively recover the bit a_i for $i = 1, \ldots, k-2$ using the knowledge of $K_i^a = \sum_{j=0}^{i-1} 2^j a_j$ obtained from the previous steps. Fix

$$\alpha = -2^{e_i}(dK_k^b + K_i^a), \quad \delta = -2^{e_i} d.$$

and send the malformed points obtained from the transformation

$$\theta_i^{-1} \begin{pmatrix} T_i & \delta & \alpha & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\delta & T_i \end{pmatrix}.$$

This transformation applied to A yields

$$A' = \theta^{-1} \begin{pmatrix} T_i & \delta + d & \alpha + a - \delta b & T_i b \\ 0 & 2^k & 2^k(b - cd - \delta c) & 2^k T_i c \\ 0 & 0 & 2^{n-k}(-d-\delta) & 2^{n-k} T_i \end{pmatrix} \sim \begin{pmatrix} 1 & d & a + 2^{e_i}(a - K_i^a) & b \\ 0 & 2^k & 2^k(b-cd) & 2^k c \\ 0 & 0 & -2^{n-k}d & 2^{n-k} \end{pmatrix}$$

$$\sim A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_i & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

To verify the above simplifications, note that $T_i^{-1} = 1 + 2^{e_i}$ (when $k \geq 1$ and $i \leq k-1$ as is the case here). Hence, we can determine the desired bit from the oracle response since $O\left(J_B, (R_1', R_2', S_1', S_2'), J_{AB}\right) = 1$ implies $a_i = 0$, and $a_i = 1$ otherwise.

Recovering $a \pmod{2^{n-k-1}}$ **and** c We recover the bits a_i and c_{i-k+1} for $i = k-1, \ldots, n-k-2$ simultaneously. Recall that we know the first k bits of b, i.e. K_k^b, as well as d from the type distinction of kernel subgroups. In the following we assume that d is an odd integer. The queries can be easily adapted to the case where d is even by shifting the indices of c accordingly.

In the first query we send the malformed points obtained from the transformation

$$\theta_i^{-1} \begin{pmatrix} T_i & \delta & \alpha & \beta \\ 0 & 1 & \beta & \gamma \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\delta & T_i \end{pmatrix}$$

where $\alpha = -2^{e_i}K_i^a$, $\beta = -2^{e_i-1}K_{i-k+1}^c d$, $\gamma = 2^{e_i}K_{i-k+1}^c$, $\delta = -2^{e_i-1}d$. Then we obtain

$$A' \sim A + \begin{pmatrix} 0 & 0 & 2^{n-1}a_i - 2^{n-k-1}d^2c_{i-k+1} & 2^{n-k-1}dc_{i-k+1} \\ 0 & 0 & 2^{n-1}dc_{i-k+1} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

This means that A and A' define the same group if

$$[2^{n-1}a_i]S_1 + [2^{n-k-1}dc_{i-k+1}]([-d]S_1 + S_2), [2^{n-1}dc_{i-k+1}]S_1 \in G_A.$$

Recall that we assume d odd and note that $[2^{n-k}]([-d]S_1+S_2) \in G_A$. Hence the oracle returns 1 if and only if $a_i = c_{i-k+1} = 0$. If the oracle returns 0, we proceed with a query to test if $a_i = 1$ and $c_{i-k+1} = 0$. This is achieved by setting $\alpha = -2^{e_i}(K_i^a + 2^i)T_i^{-1}$ in the query above. Similarly, we test for $a_i = 0$ and $c_{i-k+1} = 1$ by setting $\beta = -2^{e_i-1}(K_{i-k+1}^c - 2^{i-k+1})dT_i^{-1}$ and $\gamma = 2^{e_i}(K_{i-k+1}^c + 2^{i-k+1})$.

Recovering b Recall that K_{n-k-1}^a, K_k^b, c and d are known from previous oracle queries. We now utilise this knowledge to find the remaining bits of b. Again, assuming d to be odd here allows us to perform the queries below for any $k \leq i < n - \max\{k, 3\}$ which can be adapted via some shift in indices to accommodate even d. Let

$$\alpha = 2^{e_i}(K_{n-k-1}^a + K_i^b d - cd^2), \quad \beta = 2^{e_i}cd, \quad \gamma = 2^{e_i}c, \quad \delta = 2^{e_i}d.$$

We then send malformed points obtained via

$$\theta_i^{-1} \begin{pmatrix} 1 & \delta & \alpha & \beta \\ 0 & T_i & \beta & -\gamma \\ 0 & 0 & T_i & 0 \\ 0 & 0 & -\delta & 1 \end{pmatrix}$$

to the oracle resulting in

$$A' \sim \begin{pmatrix} 1 & \delta + T_i d & \alpha + \beta d - \delta b + T_i \alpha & \beta - \gamma d + b \\ 0 & 2^k & 2^k(T_i^{-1}\beta + b - dc - T_i^{-1}\delta c) & 2^k T_i^{-1}(c - \gamma) \\ 0 & 0 & 2^{n-k}(-T_i d - \delta) & 2^{n-k} \end{pmatrix} = A + \begin{pmatrix} 0 & 0 & 2^{e_i} d(K_i^b - b) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

so that the oracle returns 1 if $b_i = 0$, and 0 if $b_i = 1$.

Recovering a It remains to recover the last bits of a, given K^a_{n-k-1} as well as b, c and d. Let $i = n-k-1, \ldots, n-3$. We again fix

$$\alpha = 2^{e_i}(K_i^a + bd - cd^2), \quad \beta = 2^{e_i} cd, \quad \gamma = 2^{e_i} c, \quad \delta = 2^{e_i} d$$

and query the oracle with the symplectic transformation

$$\theta_i^{-1} \begin{pmatrix} 1 & \delta & \alpha & \beta \\ 0 & T_i & \beta & -\gamma \\ 0 & 0 & T_i & 0 \\ 0 & 0 & -\delta & 1 \end{pmatrix}.$$

We obtain

$$A' \sim A + \begin{pmatrix} 0 & 0 & 2^{e_i}(K_i^a - a) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Hence, we can deduce the bit a_i from the response of the oracle whereby

$$O\left(J_B, (R'_1, R'_2, S'_1, S'_2), J_{AB}\right) = 1$$

implies $a_i = 0$, and $a_i = 1$ otherwise.

Since the square root of $T_i = 1 - 2^{e_i}$ is not defined when $i \geq n - 3$, we cannot scale the malformed points in order to obtain a valid symplectic transformation. Therefore, the last three bits of a need to be recovered by brute force.

5.2.6 Complexity of the attack

Kernels of rank-2 Taking into account that the case distinction strategy outlined in Section 5.2.3 requires at most 6 queries to determine any type of rank-2 kernel subgroup as well as the information we learn about the parity of Alice's scalars throughout the process, we find that this attack requires at most $6 + 4(n-4) = 4n - 10$ queries, each corresponding to one isogeny computation. This leaves 3 bits per secret scalar, hence 9 bits in total, to be recovered through brute force.

Kernels of rank-3. If Alice's kernel has rank 3, we can learn the type of the subgroup as well as the scalars d and $b \pmod{2^k}$ following Section 5.2.3 with at most $4 + 4k$ queries. We further require $k - 1$ queries, one for each of the first $k - 1$ bits of a, and then 3 queries for each step of the parallel recovery of a_i and c_{i-k+1}, summing to $4+4k+k-1+3(n-2k-2) = 3n - k - 3$ queries thus far. Each remaining bit of b and a, potentially bar the last $4 - k$ and 3 bits respectively, requires exactly one query to recover, adding $n - 6$ queries. This leads to a total number of at most $4n - k - 9$ queries to recover Alice's secret key while leaving 3 bits of a as well as $4 - k$ bits of b and $3 - k$ bits of c (if $k < 4$ and $k < 3$, respectively) to brute force.

5.3 Generalising the attack

For the attack presented in Section 5.2, we assume that Alice, the party whose secret scalar we want to recover, utilises a public basis of her torsion subgroup which is symplectic. Though we will show in Section 5.4 that working with symplectic bases in G2SIDH is a natural generalisation from the elliptic curve SIDH protocol, using this specific type of basis was not included in the initial proposal of the scheme by Flynn and Ti [FT19] and can hence not be assumed. In an instance where arbitrary bases are used, the malformed points presented in the previous section can easily be detected by the party under attack. Upon receiving Bob's public information $(J_B, R_1, R_2, R_3, R_4)$, Alice can compute the Weil pairing of the points in the public key and compare it to the Weil pairing of the original basis $\{P_{A,i}\}_{i=1}^4$ contained in the public parameters of the scheme. Should it not be true that

$$e_{\ell_A^{e_A}}(R_i, R_j) = e_{\ell_A^{e_A}}(P_{A,i}, P_{A,j})^{\deg \phi_B}$$

for some pair $1 \leq i, j \leq 4$, by the properties of the Weil pairing, Alice can conclude that the provided torsion information is malformed in some way. Even if this does not necessarily indicate a malicious second party, she can then abort the protocol to thwart a possible attack.

However, it is still possible for an attacker to perform the adaptive attack to recover the other party's static secret key in this scenario. First, some pre-computation is required to generate a symplectic torsion basis, then our attack can be "translated" to the general setting through a change of basis.

5.3.1 Symplectic basis algorithm

We begin by describing an algorithm which deterministically computes a symplectic basis from an arbitrary one in a straightforward way. Let J be a PPSSAS over some finite field $\mathbb{F}_{p^2}$

and m an integer not divisible by p. Suppose we are given an arbitrary basis (R_1, R_2, R_3, R_4) for $J[m]$, Algorithm 5 constructs a basis (P_1, P_2, Q_1, Q_2) for $J[m]$ satisfying the required pairing conditions (cf. Definition 5.2). The algorithm resembles the Gram–Schmidt process for orthonormalisation.

Algorithm 5: Converting an arbitrary set of generators of the torsion subgroup to a symplectic basis.

Data: Basis (R_1, R_2, R_3, R_4) for $J[m]$
Result: Symplectic basis (P_1, P_2, Q_1, Q_2) for $J[m]$
1 Set $P_1 \leftarrow R_1$;
2 **if** $\mathrm{ord}(e(P_1, R_2)) = m$ **then**
3 | Set $Q_1 \leftarrow R_2$;
4 **else if** $\mathrm{ord}(e(P_1, R_3)) = m$ **then**
5 | Set $Q_1 \leftarrow R_3$;
6 | Set $R_3 \leftarrow R_2$;
7 **else**
8 | Set $Q_1 \leftarrow R_4$;
9 | Set $R_4 \leftarrow R_2$;
10 Set $\zeta \leftarrow e(P_1, Q_1)$;
11 Set $\lambda_1 \leftarrow \log(\zeta, e(Q_1, R_3))$, $\lambda_2 \leftarrow \log(\zeta, e(P_1, R_3))$;
12 Set $P_2 \leftarrow R_3 + [\lambda_1]P_1 - [\lambda_2]Q_1$;
13 Set $\mu_1 \leftarrow \log(\zeta, e(Q_1, R_4))$, $\mu_2 \leftarrow \log(\zeta, e(P_1, R_4))$, $\mu_3 \leftarrow \log(\zeta, e(P_2, R_4))$;
14 Set $Q_2 \leftarrow [\mu_3^{-1}](R_4 + [\mu_1]P_1 - [\mu_2]Q_1)$;
15 Return (P_1, P_2, Q_1, Q_2);

We first fix P_1 to be any element of the arbitrary basis and then begin to construct a symplectic basis around it. Then one finds an element Q_1 such that $e(P_1, Q_1)$ has full order m by determining which of the remaining three elements from the input basis already fulfils this condition. The pairing value of P_1 and P_2 then allows us to fix ζ_m. It remains to construct P_2 and Q_2 which should be "orthogonal" to P_1 and Q_1 respectively. A precise procedure for this is described in Lines 11 to 14 of Algorithm 5.

5.3.2 Attack on an arbitrary basis

As shown in Algorithm 5, we are able to obtain a symplectic basis from an arbitrary basis using a 4×4 change of basis matrix. In particular, such a basis is of the form

$$\mathcal{T} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ \gamma_1 & -\gamma_2 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ \mu_3^{-1}\mu_1 & -\mu_3^{-1}\mu_2 & 0 & \mu_3^{-1} \end{pmatrix}$$

up to swapping certain columns depending on the result of the `if` branch of Algorithm 5, such that for example $\mathcal{T} \cdot (R_1\ R_2\ R_3\ R_4)^T = (P_1\ P_2\ Q_1\ Q_2)^T$ if Line 3 is executed. This matrix $\mathcal{T}$, together with its inverse, allows us to transform elements written in terms of one basis into the other. Each time we need to query the oracle on a particular set of malformed points (R_1, R_2, S_1, S_2) expressed in the symplectic basis, we translate these points with the inverse of the matrix $\mathcal{T}$ to the original setting and get the corresponding malformed points in terms of the arbitrary basis as

$$\mathcal{T}^{-1} \cdot (R_1\ R_2\ S_1\ S_2)^T.$$

We still obtain the same bit of information in return: Either the oracle confirms that the malformed variety coincides with the reference variety, or it indicates the opposite. This ultimately allows one to recover the secret for a symplectic basis which is equivalent to knowing the secret isogeny. Note that the attack is still not detectable by the Weil pairing if the transformations from the previous sections are applied.

5.4 Another look at SIDH and GPST

In the previous sections we studied different aspects of the generalisation of the SIDH scheme to abelian surfaces. For that purpose it was necessary to introduce different notions that did not appear in the description of the SIDH protocol for elliptic curves. In order to give some intuition on the different terms, we explain their meaning in the case of elliptic curves and demonstrate the analogies to our setting here.

5.4.1 Revisiting the SIDH keyspace

A secret key of Alice is a cyclic subgroup of E order 2^{e_A}, and similarly Bob's secret key is a cyclic subgroup of order 3^{e_A}. In analogy to the definition in Section 5.1, we denote

$$\mathcal{K}_\ell = \{G \subset E \mid G \text{ cyclic and } \#G = \ell^n\},$$

for $\ell \in \{2, 3\}$ and $n = e_A$ or e_B, respectively.

It is easy to see that a group $G \in \mathcal{K}_\ell$ is maximal ℓ^n-isotropic with respect to the Weil-pairing on E.[5] Moreover $G \not\subset E[m]$ for any $m < \ell^n$, since a generator of G has order ℓ^n. In particular, we may use the equivalent definition

$$\mathcal{K}_\ell = \{G \subset E \mid G \text{ maximal } \ell^n\text{-isotropic and } G \not\subset E[m] \text{ for any } m < \ell^n\},$$

which resembles the definition in Section 5.1.4 more closely.

An important ingredient in the classification of secret keys for G2SIDH is the use of symplectic bases for the torsion groups $J[\ell^n]$. In the elliptic curve setting one automatically works with symplectic bases.

[5] Isotropy follows from the fact that G is cyclic. To see that G is maximal with this property, consider some $R \in G$ with $G = \langle R \rangle$ and note that for any $R' \in E \setminus G$, the Weil pairing $e_{\ell^n}(R, R')$ is non-trivial.

Lemma 5.11. *Let $\ell \neq p$ be a prime and $n > 0$. Then every basis (P, Q) for $E[\ell^n]$ is symplectic with respect to the Weil pairing.*

It is well known that the keyspace of SIDH, $\mathcal{K}_\ell$, can be divided into two disjoint sets as follows

$$\mathcal{K}_\ell = \{\langle P + [a]Q\rangle \mid a \in \mathbb{Z}/\ell^n\mathbb{Z}\} \cup \{\langle [\ell a]P + Q\rangle \mid a \in \mathbb{Z}/\ell^n\mathbb{Z}\}.$$

In the terminology of Section 5.1.3, this means that there are two types of groups in $\mathcal{K}_\ell$ as opposed to the multitude of types in the genus-2 setting (cf. Definition 5.6 and Table 5.1). To make the analogy more explicit, we introduce the following terminology.

Setup

- prime $p = 2^{e_A} \cdot 3^{e_B} \cdot f - 1$
- supersingular elliptic curve E
- torsion bases (P_A, Q_A) for $E[2^{e_A}]$ and (P_B, Q_B) for $E[3^{e_B}]$

Key Generation

• $\alpha \xleftarrow{\$} \{0, \ldots, 2^{e_A} - 1\}$ • $A = P_A + [\alpha]Q_A$ • $\phi_A : E \to E_A = E/\langle A\rangle$	$E_A, \phi_A(P_B), \phi_A(Q_B)$ ⟶ ⟵ $E_B, \phi_B(P_A), \phi_B(Q_A)$	• $\beta \xleftarrow{\$} \{0, \ldots, 3^{e_B} - 1\}$ • $B = P_B + [\beta]Q_B$ • $\phi_B : E \to E_B = E/\langle B\rangle$

Shared Key

$$E_B/\langle \phi_B(P_A) + [\alpha]\phi_B(Q_A)\rangle = \quad E/\langle A, B\rangle \quad = E_A/\langle \phi_A(P_B) + [\beta]\phi_A(Q_B)\rangle$$

Figure 5.4: SIDH protocol with restricted keyspace.

Definition 5.12 (Classification)**.** *Let (P, Q) be a basis for $E[\ell^n]$. For a group $G = \langle G_1\rangle \simeq \mathcal{C}_{\ell^n}$ in $\mathcal{K}_\ell$, we say that G_1 is the* canonical generator *of G if one of the following is true for some $a \in \mathbb{Z}/\ell^n\mathbb{Z}$.*

1.1 $G_1 = \begin{pmatrix} 1 & a \end{pmatrix} \cdot \begin{pmatrix} P \\ Q \end{pmatrix}$.

1.2 $G_1 = \begin{pmatrix} a & 1 \end{pmatrix} \cdot \begin{pmatrix} P \\ Q \end{pmatrix}$ *and* $\ell \mid a$.

Moreover we say that a group $G \in \mathcal{K}_\ell$ is of Type 1.i *for $i \in \{1, 2\}$ depending on which of the cases above applies.*

Note that the entire keyspace has cardinality $\ell^{n-1}(\ell + 1)$. However in practice one restricts to the groups of Type 1.1. This restricted keyspace has cardinality ℓ^n [JACCD-HHJKKLLNPRSU22, Section 1.3.9]. This is very similar to the restriction suggested in Section 5.1.4. A sketch of the SIDH protocol using the restricted keyspace is provided in Fig. 5.4.

5.4.2 Revisiting GPST

Recall the adaptive GPST attack on SIDH due to Galbraith, Petit, Shani and Ti [GPST16] which we presented in Section 2.1.2. In order to illustrate the connection to our adaptive attack on G2SIDH, we use a different terminology in this section.

A major obstruction when devising an attack strategy for G2SIDH was to avoid detection by the Weil pairing. We overcame potential detection by only allowing symplectic transformations of the basis elements for the oracle queries. Indeed this strategy is also followed in [GPST16] albeit not explicitly phrased in this way.

Assume that Alice uses a fixed secret key x_A, y_A defining the group $G_A = \langle [x_A]P_A + [y_A]Q_A \rangle \in \mathcal{K}_2$. Her public key is of the form $(E_A, \phi_A(P_B), \phi_A(Q_B))$, where E_A is the codomain of the isogeny $\phi_A : E \to E_A$ with kernel G_A. It is assumed that the attacker has access to the oracle

$$O\left(E_1, (R, S), E_2\right) = \begin{cases} \perp & \text{if } \; e_{2^n}(R, S) \neq e_{2^n}(P_A, Q_A)^{3^m}, \\ 1 & \text{if } \; E_2 \simeq E_1/\langle [x_A]R + [y_A]S \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

Honestly running the protocol, the attacker first generates Bob's ephemeral values

$(E_B, R = \phi_B(P_A), S = \phi_B(Q_A))$ and computes the elliptic curve E_{AB}.[6] Then they send different queries to the oracle with fixed curves E_B and E_{AB}, while the basis (R, S) is modified in each step. In order to create a valid query it is necessary that the Weil pairing on the malformed basis elements (R', S') coincides with that on (R, S). Phrased differently, the basis transformation

$$(R' = [m_{1,1}]R + [m_{1,2}]S,\ S' = [m_{2,1}]R + [m_{2,2}]S) \leftarrow (R, S)$$

has to be a symplectic transformation. Note that this transformation can be represented by the 2×2 matrix

$$M = \begin{pmatrix} m_{1,1} & m_{1,2} \\ \\ m_{2,1} & m_{2,2} \end{pmatrix}.$$

It is easy to see that this matrix is symplectic if and only if $\det(M) = 1$. [7]

Case distinction The first step in the attack is to distinguish between groups of Type 1.1 and 1.2. This is done by sending the malformed points $(R + [2^{n-1}]S, S)$ to the oracle. These malformed points are obtained from the transformation

$$M = \begin{pmatrix} 1 & 2^{n-1} \\ \\ 0 & 1 \end{pmatrix}.$$

Note that

$$\langle [x_A](R + [2^{n-1}]S) + [y_A]S \rangle = \langle [x_A]R + [y_A]S + [2^{n-1}x_A]R \rangle.$$

[6] Note that by construction $O(E_B, E_{AB}, (R, S)) = 1$.

[7] Note that this criterion is unique to the case of 2×2 matrices. For $2n \times 2n$ matrices with $n > 1$ the condition $\det(M) = 1$ is necessary but not sufficient.

This coincides with $\langle [x_A]R + [y_A]S \rangle$ if and only if x_A is even. It follows that

$$O\left(E_B, (R + [2^{n-1}]S,\ S), E_{AB}\right) = \begin{cases} 1 & \text{if } G_A \text{ is of Type 1.2,} \\ 0 & \text{if } G_A \text{ is of Type 1.1.} \end{cases}$$

In the following we assume that G_A is of Type 1.1 and denote the canonical generator by $A_1 = R + [a]S$. In order to be consistent with the notation from Section 5.2, we set $A = \begin{pmatrix} 1 & a \end{pmatrix}$, hence $A_1 = A \cdot \begin{pmatrix} R & S \end{pmatrix}^T$.

First bit recovery In order to find the first bit of a, the attacker queries the oracle on the malformed points $\begin{pmatrix} R' & S' \end{pmatrix}^T = M_0 \cdot \begin{pmatrix} R & S \end{pmatrix}^T$, where

$$M_0 = \begin{pmatrix} 1 & 0 \\ 2^{n-1} & 1 \end{pmatrix}.$$

Note that $A \cdot \begin{pmatrix} R' & S' \end{pmatrix}^T = A' \cdot \begin{pmatrix} R & S \end{pmatrix}^T$, where

$$A' = A \cdot M_0 = \begin{pmatrix} 1 & a \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 2^{n-1} & 1 \end{pmatrix} = \begin{pmatrix} 1 + 2^{n-1}a & a \end{pmatrix}.$$

It follows that

$$O(E_B, E_{AB}, (R', S')) = \begin{cases} 1 & \text{if } a \text{ is even,} \\ 0 & \text{if } a \text{ is odd.} \end{cases}$$

Iterative step Assume the attacker has recovered the first i bits of a. Then we know K_i^a, where $a = K_i^a + \sum_{j=i}^{n-1} a_j 2^j$. If $i < n-3$, there exists θ satisfying $\theta^2 \equiv 1 + 2^{n-i-1} \pmod{2^n}$, [GPST16, Lemma 3.3]. Necessarily θ is an odd integer, hence invertible modulo 2^n. Consider

the transformation $\begin{pmatrix} R' & S' \end{pmatrix}^T = M_i \cdot \begin{pmatrix} R & S \end{pmatrix}^T$, where

$$M_i = \theta^{-1} \cdot \begin{pmatrix} 1 & -2^{n-i-1}K_i^a \\ 0 & 1+2^{n-i-1} \end{pmatrix}.$$

Clearly M_i is symplectic, hence the tuple $(E_B, E_{AB}, (R', S'))$ defines a valid query. Here, we obtain

$$A' = A \cdot M_i = \begin{pmatrix} 1 & a \end{pmatrix} \cdot \theta^{-1} \cdot \begin{pmatrix} 1 & -2^{n-i-1}K_i^a \\ 0 & 1+2^{n-i-1} \end{pmatrix} = \theta^{-1} \begin{pmatrix} 1 & a + 2^{n-i-1}a - 2^{n-i-1}K_i^a \end{pmatrix}$$
$$= \theta^{-1} \begin{pmatrix} 1 & a + 2^{n-1}a_i \end{pmatrix}.$$

This shows that $\langle A \cdot \begin{pmatrix} R & S \end{pmatrix}^T \rangle = \langle A' \cdot \begin{pmatrix} R & S \end{pmatrix}^T \rangle$ if and only if a_i is even. In conclusion

$$O\left(E_B, (R', S'), E_{AB}\right) = \begin{cases} 1 & \text{if } a_i \text{ is even,} \\ 0 & \text{if } a_i \text{ is odd.} \end{cases}$$

Thus the attacker can iteratively recover the first $n-2$ bits of a. For the remaining two bits, the authors suggest to use a brute force method.

Remark 5.13. *It is a priori not obvious how to find symplectic transformations which can be used to recover the parity of some bit a_i of the secret scalar. In [GPST16], the authors use the following strategy.*

First, they find a transformation $\begin{pmatrix} R' & S' \end{pmatrix} \leftarrow \begin{pmatrix} R & S \end{pmatrix}$ using some clever reverse engineering such that

$$\langle R + [a]S \rangle = \langle R' + [a]S' \rangle \quad \Leftrightarrow \quad a_i \text{ is even.}$$

In the second step, the authors multiply R' and S' by a scalar to make the transformation

symplectic. This corresponds to multiplying the associated matrix by a scalar such that its determinant is 1*. The latter only works if the determinant is a square modulo* 2^n*. This condition also prevents the use of the same attack approach for recovering the last bits.*

*The first step could be translated directly to the G2SIDH setting. However, the scaling in the second step is not possible in the genus-*2 *case. Let* (R_1, R_2, S_1, S_2) *be an arbitrary basis for* $J[2^n]$*. Then in general there will not exist an integer* λ *with the property that* $(\lambda R_1, \lambda R_2, \lambda S_1, \lambda S_2)$ *is symplectic. As a consequence, it was necessary to use a different strategy for finding suitable symplectic transformations.*

5.5 Improvements and outlook

In this chapter, we have made two contributions. We first thoroughly analysed the generators of maximal ℓ^n-isotropic subgroups of principally polarised superspecial abelian surfaces. Our classification of normalised generators allowed us to propose a key generation algorithm for G2SIDH which is very simple and allows for uniform sampling from a slightly reduced keyspace. Secondly, we presented a polynomial-time adaptive attack on static G2SIDH which utilises our prior classification of kernel types.

For the attack, we make the assumption that Alice's secret isogeny is a chain of Richelot isogenies and therefore explicitly work with elements of $J[2^n]$. This strategy can be translated to recover a key for more general small primes ℓ and therefore ℓ^{e_ℓ}-torsions of J due to Proposition 5.4. The resulting attack on G2SIDH with a different prime ℓ may not return a bit of information with every single query, but may require a small number of additional queries to determine a bit of information. The attack can still be carried out successfully, though it might only be of theoretical interest due to inefficient isogeny computation and the new passive attacks on SIDH-based schemes.

As discussed in Section 2.2.3, G2SIDH is now fully broken due to a generalisation of techniques from [CD23] by Robert [Rob23]. However, it remains to be seen whether the innovative, yet novel and therefore largely untested, countermeasures to these types of attacks will prove themselves in the one-dimensional setting and whether they will translate to abelian surfaces without problems. While masking the isogeny degree as suggested in [FMP23] for the MD-SIDH protocol will be tricky in the two-dimensional setting as computing general (ℓ, ℓ)-isogenies (especially for primes larger than $\ell = 5, 7$) is still inefficient and thus does not leave much room for varying isogeny degrees, masking the torsion point information as for M-SIDH seems more feasible. Overcoming the technicalities preventing a straightforward adaptation and explicitly constructing G2SIDH with scaled torsion information is left for further work.

It further remains to explore how one can improve the naive method for computing symplectic torsion bases which we presented in Section 5.3.1. The lack of a faster algorithm is for example a hindrance for the more efficient ways of computing $(2, 2)$- and $(3, 3)$-isogeny chains due to Kunzweiler [Kun22] and Kunzweiler–Decru [DK23].

CHAPTER 6

Improved quantum algorithms for finding fixed-degree isogenies between supersingular elliptic curves

Personal contributions: *The results presented in Chapter 6 are based on collaborative work with Benjamin Benčina, Péter Kutas, Simon-Philipp Merz, Christophe Petit and Miha Stopar and have been submitted for review under the title* Improved quantum algorithms for finding fixed-degree isogenies between supersingular elliptic curves. *Hence the corresponding article is yet to be published. My main contributions within the results presented in this chapter were to the theoretical analysis of the isogeny finding strategies, especially those utilising Cornacchia's algorithm and Coppersmith's variant for bivariate equations, as well as formalising ideas and writing up further sections of the article and pseudocode for its submission.*

In this chapter we examine the complexity of one of the computational problems integral to isogeny-based cryptography. Finding *any* isogeny between two given supersingular

elliptic curves, i.e. solving the pure isogeny problem (cf. Problem 2.1), is a natural algorithmic task which is known to be equivalent to computing the curves' endomorphism rings. When the isogeny is additionally required to have a specified degree d, the problem appears to be somewhat different in nature. However, this variant is still considered a hard problem, and one which isogeny-based cryptosystems rely on for security. We formalise this problem as follows.

Problem 6.1 (Fixed-degree isogeny problem)**.** *Given supersingular elliptic curves E and E' defined over the field $\mathbb{F}_{p^2}$ with p^2 elements, and given a positive integer d, find an isogeny $E \to E'$ of degree d if such an isogeny exists.*

Variants of the pure isogeny problem such as the fixed-degree version Problem 6.1, where a solution is required to fulfill certain further properties, arise in cryptanalysis from the additional public information supplied by various isogeny-based protocols. The existence of a solution with specific attributes supplies additional information for solving the problem. Thus, it is a priori not clear whether finding a solution with specific properties, when such a solution is known to exist, will lead to an easier or harder problem compared to the pure isogeny problem.

In 2021, Wesolowski proved that the pure isogeny problem between supersingular elliptic curves reduces to the computation of their endomorphism rings [Wes22b] which was previously only proved under certain heuristics [KLPT14; PL17; EHLMP18]. Yet, it is not clear how the hardness of finding an isogeny of a specific degree compares to the hardness of the pure isogeny problem in general.

Classically, the best known methods for solving Problem 6.1 are based on exhaustive search, meet-in-the-middle search or more general collision finding algorithms tailored to the concrete amount of memory available [CLNRV20]. Regarding quantum algorithms, Tani's claw finding algorithm [Tan09] was considered to solve Problem 6.1. However, Jaques and

Schanck argued that the algorithm's cost of accessing memory renders it more expensive than its classical counterpart [JS19], and the algorithm was widely dismissed. Finally, Fouotsa, Kutas, Merz and Ti gave a reduction of Problem 6.1 to the problem of computing the curves' endomorphism rings, if additionally the image of a sufficiently large torsion subgroup is known under the secret isogeny [FKMT22]. In an updated version of their article, the authors show that it is also sufficient if the image of a slightly larger torsion subgroup is only known up to some scalar [FKMT21].

We propose several new algorithms to solve Problem 6.1 in this chapter. Our general approach is divided into three distinct steps.

1. Compute the endomorphism rings $\mathcal{O}$ and $\mathcal{O}'$ of E and E'.

2. Construct a connecting ideal I between the quaternion orders $\mathcal{O}$ and $\mathcal{O}'$ and find a representation of d by the norm form associated to $\mathrm{Hom}(E, E')$.

3. Convert the ideal back to some suitable isogeny representation: Depending on the smoothness of the degree d, this can be as a composition of rational maps or a less straightforward representation.

The main focus of our work is solving the norm equation in Item 2 when the isogeny degree is relatively small. In general, this task can be seen as the quaternion version of the isogeny problem for fixed degrees: We first compute an LLL-reduced basis of $\mathrm{Hom}(E, E')$ and write the norm form with respect to this basis. The problem can thus be expressed as finding a solution to

$$Q(x_1, x_2, x_3, x_4) = d \tag{6.1}$$

where Q is a quadratic form. Firstly note that when we have a smooth isogeny degree $d = \ell^e$ for some prime ℓ, a solution can be found ℓ-adically. Note furthermore that using the LLL algorithm to reduce our basis allows us to bound the coefficients of Q. We then consider

multiple approaches to solving Equation (6.1) which combine the guessing of certain variables (using Grover's algorithm) and methods for solving quadratic Diophantine equations such as Cornacchia's algorithm and multivariate variants of Coppersmith's method.

In the first approach, we simply guess two variables and consider the remaining bivariate equation. Such an equation can be transformed into one efficiently solvable by Cornacchia's algorithm. For an ideal norm and isogeny degree of $d \approx p^{1/2+\epsilon}$ for some $\epsilon > 0$, the cost of this approach is $p^{2\epsilon}$ to test all possible guesses, or p^{ϵ} using a quantum computer and Grover search [Gro96]. Adding the costs of precomputing endomorphism rings, we obtain total costs for isogeny computation of $\max\{p^{1/2}, p^{2\epsilon}\}$ classically and $\max\{p^{1/4}, p^{\epsilon}\}$ quantumly.

Our second approach uses Coppersmith-type methods for solving the norm equation. Here we have various algorithms available depending on how many variables we want to guess. If we guess two variables we can use bounds derived by Coron [Cor07] to show that the algorithm succeeds whenever $\epsilon < 1/4$. The advantage of this approach is that it requires less heuristics than the Cornacchia approach (while achieving the same complexity). If we guess only one variable we can use multivariate Coppersmith techniques, either a generalisation of Coron's approach or a more rigorous variant due to Bauer and Joux [BJ07].

As an additional application of our algorithms, we discuss utilising the trivariate Coppersmith approach for solving another quaternion problem relevant to isogeny-based cryptography, namely that of embedding given quadratic orders into maximal orders. We provide a heuristic method which embeds orders of small discriminant in polynomial time, or alternatively detects that no such embedding exists. Based on our experiments, the approach works for quadratic orders of discriminant up to $p^{0.8}$.

Our contributions are presented in the following way. We begin in Section 6.1 with an exposition of several relevant techniques and algorithms for solving multivariate integer

equations which we utilise for our methods, followed by a summary of state-of-the-art isogeny and endomorphism ring computations. The following sections describe our methods for solving the quaternion version of the isogeny computation problem. Two main avenues for finding elements of prescribed norm in the connecting ideal of two maximal quaternion orders are explored: We first utilise a well-known algorithm solving bivariate quadratic equations over the integers due to Cornacchia; the theoretical analysis of this approach can be found in Section 6.2. Secondly, we use Coppersmith's algorithms and variants thereof to solve bivariate and trivariate norm equations in Section 6.3 and present some experimental results making use of this strategy. The order embedding problem, as another application for solving multivariate equations with the provided algorithms, is discussed in Section 6.4. Lastly, we make explicit how solving the norm equation equips us with enough information to finally compute a degree-d isogeny between the two given curves in Section 6.5. In Section 6.6, we provide a summary of our results and ways to improve these further.

6.1 Preliminaries

In the following we first present algorithms for solving Diophantine equations due to Coppersmith, as well as variants thereof proposed by Coron as well as Bauer and Joux. We furthermore give an overview of the current best-known methods for computing isogenies and endomorphism rings, allowing us to position our improved algorithms in relation to the state of the art.

6.1.1 Coppersmith's methods

Inspired by lattice-techniques from Håstad [Has86] and Girault–Toffin–Valleé [GTV90], Coppersmith's methods can find small roots of polynomial equations over either $\mathbb{Z}$ or any integer ring $\mathbb{Z}/N\mathbb{Z}$ by forming lattices from the polynomial's coefficients and those of "related" polynomials and extracting the roots from the reduced lattice. These algorithms have found

many applications in cryptography from the cryptanalysis of RSA with small public exponent when some part of the message is known [Cop97] or when the private exponent d is smaller than some bound $N^{0.29}$ [BD99] to the polynomial-time factorisation of $N = p^r q$ for large r [BDH99].

Several variants of Coppersmith's original algorithms [Cop96a; Cop96b; Cop97] exist, while Howgrave-Graham [How97] offers an alternative approach. The latter is often argued to be simpler to analyse [Cor07] but the two approaches work equally well asymptotically, and can both be generalised to handle polynomials with multiple variables [Cor07; BJ07]. In the following we focus on three multivariate variants for which an implementation was publicly available.

Bivariate approach of Coron

Coron's algorithm [Cor07] finds small roots of bivariate integer polynomials and follows Howgrave-Graham's approach. In this variant, the lattice reduction is applied to a full rank lattice that admits a natural triangular basis, hence the determinant can be easily computed.

Suppose we are given an irreducible polynomial $P(x, y) = \sum_{i,j} p_{i,j} x^i y^j$ of total degree δ with coefficients in $\mathbb{Z}$ and the promise that it has an integer root (x_0, y_0) where $x_0 < X$ and $y_0 < Y$. Our goal is now to recover this small root (x_0, y_0) of P. Let $a := P(0, 0) = p_{0,0}$ and denote by W the quantity $\|P(xX, yY)\|_\infty$ where $\|P(x, y)\|_\infty = \max_{i,j} |p_{i,j}|$. We then generate an integer n coprime to a such that $W \leq n < 2 \cdot W$, and define the normalised modular polynomial

$$q(x, y) := a^{-1} P(x, y) \pmod{n}.$$

We then consider two further types of polynomials. For all monomials $x^i y^j$ such that $0 \leq$

$i, j \leq k$ for some chosen constant k, we form polynomials of the form

$$q_{ij} = X^{k-i}Y^{k-j}x^i y^j q(x, y).$$

For the remaining monomials up to degree $\delta + k$, i.e. the monomials with $0 \leq i, j \leq \delta + k$ but not both $i, j < k + 1$, we form

$$q_{ij}(x, y) = nx^i y^j.$$

Note that all these polynomials also have a root at (x_0, y_0) such that $q_{ij}(x_0, y_0) = 0 \pmod{n}$.

Let M be the set of all monomials of the polynomials q_{ij}, and denote by m the number of elements in M. Notice that we have precisely m polynomials q_{ij}. Form a matrix M_1 by labeling each column with a monomial in M, and let the coefficients of the polynomials q_{ij} be the elements in the corresponding rows. Denote by L_1 the lattice generated by the rows of M_1. By applying LLL reduction [LLL82] to L_1 and considering the vectors of the LLL-reduced basis $(b_1, \ldots, b_m)$ of L_1 in order, we obtain a polynomial h defining the hyperplane of the lattice containing the small solutions to the original polynomial P. Hence, h also admits (x_0, y_0) as a root modulo n, but has smaller coefficients than P due to LLL-reduction. If the solution (x_0, y_0) is sufficiently small, the polynomial h will also admit it as a root over the integers such that $h(x_0, y_0) = 0$ also holds. Then, the zeros can easily be computed. More precisely, if we define $\|h(x, y)\|^2 := \sum_{i,j} |h_{ij}|^2$ for h_{ij} the coefficient of the monomial $x^i y^j$ in a polynomial h, we have the following result due to Howgrave-Graham [How97].

Lemma 6.2. *Let $h(x, y) \in \mathbb{Z}[x, y]$ be a sum of at most ω monomials. Suppose that $h(x_0, y_0) = 0 \pmod{n}$, where $|x_0| \leq X$ and $|y_0| \leq Y$ and $\|h(xX, yY)\| < n/\sqrt{\omega}$, then $h(x_0, y_0) = 0$ holds over the integers.*

Due to the way the polynomial $h(x, y)$ has been constructed, it cannot be a multiple

of $P(x, y)$ when the conditions of Lemma 6.2 hold. Since $P(x, y)$ is assumed to be irreducible and $h(x, y)$ is not a multiple of $P(x, y)$, $Q(x) = \text{Resultant}_y(h(x, y), P(x, y))$ gives a non-zero integer polynomial such that $Q(x_0) = 0$. Using any standard root-finding algorithm, x_0 can be recovered first, and finally y_0 can be computed by straightforwardly solving $P(x_0, y) = 0$.

The preformance of Coron's bivariate algorithm can be summarised in the following two theorems.

Theorem 6.3 ([Cor07, Theorem 2]). *Let $P(x, y)$ be an irreducible polynomial over $\mathbb{Z}$ of maximum degree δ in each variable. Let X and Y be upper bounds on the desired integer solution (x_0, y_0), and denote $W = \max_{i,j}|p_{ij}|X^iY^j$. If $XY < W^{\frac{2}{3\delta}}$, then in time polynomial in $(\log W, 2^\delta)$, one can find all integer roots (x_0, y_0) of P bounded by X and Y.*

Theorem 6.4 ([Cor07, Theorem 3]). *With the hypothesis of Theorem 6.3, except that P has total degree δ, the bound is $XY < W^{\frac{1}{\delta}}$.*

Multivariate approach of Coron

Coron's method as described above can also be extended to handle polynomial equations in more than two variables [Cor07, Section 3.3], but the extension is only heuristic.

In the trivariate case, polynomials defining the lattice will now be of the forms $x^iy^jz^lX^{k-i}Y^{k-j}Z^{k-l}q(x, y, z)$ and $nx^iy^jz^t$ which again preserve the desired root and evaluate to 0 over $\mathbb{Z}/n\mathbb{Z}$ at (x_0, y_0, z_0). Note that given a polynomial $P(x, y)$, a bivariate algorithm only needs to compute one polynomial $h(x, y)$ that is algebraically independent from P to be able to compute (x_0, y_0) such that $P(x_0, y_0) = 0$. On the other hand, when given a polynomial $P(x, y, z)$, we require two polynomials $h_1(x, y, z)$ and $h_2(x, y, z)$, where P, h_1 and h_2 are pairwise algebraically independent. The heuristic nature of the algorithm stems from the difficulty to guarantee algebraic independence (while linear independence when seen as vectors is guaranteed). In practice however, algebraic dependencies are rarely an issue. The

method similarly generalises to more variables.

While Coron does not state a formal claim about the performance of this variant (even up to an algebraic independence assumption), it is a priori similar to the following method, which does handle algebraic dependencies.

Bauer–Joux approach

In contrast to Coron's algorithm which generalised the simplification found by Howgrave-Graham, the approach by Bauer and Joux [BJ07] extends the original bivariate approach by Coppersmith [Cop96a] to three variables. It also uses truncated Gröbner bases to handle so-called algebraic dependencies. A similar approach without using Gröbner bases was already proposed by e.g. [Jut98]; the main contribution of [BJ07] is a criterion for rigorous success, although it is worth noting that their algorithm still uses heuristics.

While Coron's approach works directly in the lattice generated by polynomials that share a common root (x_0, y_0, z_0) we wish to find, the Bauer–Joux approach aims to find a vector that is orthogonal to the vector s_0, which we define later, that is derived from the root. This yields a polynomial that shares the root (x_0, y_0, z_0) with the initial polynomial.

Again, let $P(x, y, z)$ be a polynomial with integer coefficients and (x_0, y_0, z_0) a small root. Having $P(x, y, z)$ and knowing the bounds $|x_0| < X$, $|y_0| < Y$, $|z_0| < Z$, the goal is to recover the root (x_0, y_0, z_0). Let (S, M) be an admissible pair of sets of monomials for P as in [BJ07], and denote by s and m the number of elements in the sets S and M, respectively. Normally, we pick a set of monomials S, then multiply them with the monomials of P to obtain the set M.

The algorithm generates the following $m \times (m+s)$ rational matrix M_1. The left $m \times m$ submatrix D_M is a diagonal rational matrix with $X^{-i}Y^{-j}Z^{-k}$ in the row corresponding to the monomial $x^i y^j z^k \in M$. The columns of the right $m \times s$ submatrix R_1 are the integer

coefficients of the polynomials $x^f y^g z^h \cdot P$ for $x^f y^g z^h \in S$, where the coefficient goes into the row belonging to the corresponding monomial [BJ07, Figure 1].

Denote by L_1 the lattice generated by the rows of M_1. Since $s < m$, there exists a sublattice $L_1' \subset L_1$ such that its vectors have the last s components equal to zero. We achieve this by noting that R_1 is an integer matrix, so we compute a unimodular transformation U that transforms R_1 into a matrix that has an $s \times s$ identity matrix on the top and zeros everywhere else, then apply U to D_M as well, and take its bottom $(m - s)$ rows as a basis of L_1' (ignoring the zeros in the last s components). Denote $M_1' = UM_1$.

Denote by $r_0 = (x_0^i y_0^j z_0^k \mid x^i y^j z^k \in M)$ the solution vector, and observe that $s_0 = r_0 M_1 = \left((\frac{x_0}{X})^i (\frac{y_0}{Y})^j (\frac{z_0}{Z})^k \;\middle|\; x^i y^j z^k \in M \right) || (0, \ldots, 0)$, where $||$ refers to concatenation of vectors, gives a short vector in L_1'. We denote $r = m - s$, compute an LLL-reduced basis $(b_1, \ldots, b_r)$ of L_1', and let $(b_1^\star, \ldots, b_r^\star)$ be its Gram-Schmidt orthogonalisation. Then when $\|s_0\| < \|b_r^\star\|$, we know that the inner product $\langle b_r^\star, s_0 \rangle = 0$, i.e. $b_r^\star$ yields a polynomial $P_1'(x, y, z)$ that annihilates $\left(\frac{x_0}{X}, \frac{y_0}{Y}, \frac{z_0}{Z}\right)$. By a change of variables we obtain $P_1(x, y, z) = P_1'\left(\frac{x}{X}, \frac{y}{Y}, \frac{z}{Z}\right)$ that has (x_0, y_0, z_0) as a root. Note that we may get other polynomials that annihilate (x_0, y_0, z_0) by considering $b_{r-1}^\star$ and so on, making the next step unnecessary.

The second step is to compute the Gröbner basis $\mathcal{G}$ of the ideal $I = (P, P_1)$, truncated at the maximal degree of the monomials in the set M. We then repeat the previous procedure almost exactly. Denote by t the number of elements in the set $\mathcal{G}$. We construct the $m \times (m+t)$ rational matrix M_2 the same way we constructed M_1 in the previous step, except that we use the polynomials from $\mathcal{G}$ in the columns of the right $m \times t$ matrix, instead of $\{m \cdot P \mid m \in S\}$. The rest of the procedure is identical, and we obtain P_2 that annihilates (x_0, y_0, z_0). Note that we cannot guarantee that P_2 is algebraically independent from P and P_1, making this algorithm heuristic, although [BJ07] gives a criterion for algebraic independence. The approach is summarised in the following theorem.

Theorem 6.5 ([BJ07, Theorem 1])**.** *If S and M are admissible sets for P, we can find in polynomial time $P_1(x, y, z)$ which has (x_0, y_0, z_0) as a root over the integers and is algebraically independent from P, provided that*

$$X^{s_x} Y^{s_y} Z^{s_z} < W_1^s 2^{-(6+c)s(d_x^2+d_y^2+d_z^2)}$$

where we assume that $(m-s)^2 \leq cs(d_x^2 + d_y^2 + d_z^2)$ for some constant c. In this formula, W_1 denotes $\|P(xX, yY, zZ)\|_\infty$, and d_x, d_y, d_z denote the maximum degree of P in x, y, z respectively. We obtain s_x by adding up the exponents of x in all monomials in the set $M \setminus S$, i.e. $s_x := \sum_{x^i y^j z^k \in M \setminus S} i$, with analogous definitions of s_y and s_z.

6.1.2 State of the art on isogeny computation

In this section, we briefly survey the current state of the art for finding isogenies between two supersingular elliptic curves and closely related algorithms. First, we discuss the problem of computing endomorphism rings of supersingular elliptic curves. Then we review algorithms that recover *any* isogeny between two given supersingular curves, before we discuss algorithms that recover an isogeny of a given degree under the premise that such an isogeny exists.

Computing endomorphism rings of supersingular elliptic curves.

The problem of computing endomorphism rings of elliptic curves was first studied by Kohel in his thesis [Koh96]. For supersingular elliptic curves over $\overline{\mathbb{F}}_p$ this is considered to be a hard problem. Kohel gave an algorithm with $O^*(p)$ time and memory costs which was later improved to $O^*(p^{1/2})$ by Galbraith. The current best algorithm in [EHLMP20] runs in $O\left((\log p)^2 p^{1/2}\right)$ with low memory requirements.

Classical algorithms to find isogenies of arbitrary degree.

For any prime p, the full supersingular isogeny graph with its roughly $p/12$ isomorphism classes of supersingular elliptic curves over $\overline{\mathbb{F}}_p$ is connected. Thus, one could use a simple collision search to find a path between two given elliptic curves in $O^*(p^{1/2})$ time and memory.

Delfs and Galbraith showed how to find isogenies in the same time but requiring significantly less memory [DG16]. Their algorithm splits the isogeny computation into two parts. First, a random walk from both given curves is computed until a connection to the subgraph of supersingular elliptic curves defined over the base field $\mathbb{F}_p$ is found. There are roughly $\sqrt{p}$ subfield curves in the full isogeny graph and therefore the step requires $O^*(p^{1/2})$ bit operations. In the second step, one searches a subfield isogeny connecting both curves defined over $\mathbb{F}_p$. Using a meet-in-the-middle strategy the isogeny can be recovered in $O^*(p^{1/4})$, or alternatively using a different collision finding algorithm requiring less memory. The concrete complexity of the Delfs–Galbraith algorithm was analysed and further improved in [CCS22]. However, the improvements did not change the asymptotic complexity of $O^*(p^{1/2})$.

Assuming GRH, the problem of finding an isogeny between two supersingular curves is polynomial time and memory equivalent to computing their endomorphism rings [Wes22b]. Using the previously mentioned algorithm by Eisenträger, Hallgren, Leonardi, Morrison and Park [EHLMP20], the endomorphism rings of supersingular elliptic curves can be computed in $O^*(p^{1/2})$. A connecting isogeny (of rather large degree) can then be computed in classical polynomial time using the KLPT algorithm or a rigorous variation due to Wesolowski [KLPT14; Wes22b].

Classical algorithms to find isogenies of fixed degree.

Computing an unknown isogeny of known degree d between two d-isogenous supersingular elliptic curves can always be done using an exhaustive search over all $O(d)$ degree d isogenies

(or equivalently their kernels). In fact, if d is a prime this is the best known method prior to the results of this paper.

When d is a smooth integer a meet-in-the-middle approach with $O^*(\sqrt{d})$ time and memory complexity can be used. However, for large d the required memory becomes unrealistic. Limiting the available memory leads to the conclusion that a van Oorschot–Wiener collision search whose concrete complexity depends on the amount of memory available is more efficient to compute the isogeny [CLNRV20].

When the endomorphism rings of the two supersingular curves are known (or have been precomputed), d does not need to be smooth but merely the product of two factors of roughly the same size to make a meet-in-the-middle approach work. This is because the isogenies corresponding to the factors, which are potentially of large degree, can be replaced by a powersmooth isogeny using for instance the KLPT algorithm [KLPT14]. While this approach adds to the overhead of the meet-in-the-middle, the powersmooth isogenies can still be computed in order to find a collision.

Computing endomorphism rings and then using an algorithm such as KLPT to compute a connecting isogeny will in general not return an isogeny of the sought degree. However, if $d \approx p^{1/2}$ or shorter, the sought isogeny is usually the shortest one between the two curves. Galbraith, Petit, Shani and Ti showed how this relative shortness could be exploited to recover the isogeny from the endomorphism rings [GPST16, Section 4.2]. They used the fact that the smallest element in the connecting ideal, which can be computed efficiently using the endomorphism rings [KV10], corresponds to the small degree d isogeny to compute first the small element and then the isogeny. The approach works in polynomial time. The result trivially generalises to isogenies slightly larger than $p^{1/2}$ by exhaustively searching over linear combinations of the smallest elements in the connecting ideal.

For larger degrees, Fouotsa, Kutas, Merz and Ti showed that an isogeny of a specific

degree can be computed efficiently, if additionally to the endomorphism rings (masked) torsion point images under the sought isogeny are known [FKMT22]. More precisely, they show how to efficiently recover an isogeny of degree $d < \frac{sT}{16}$, where s denotes the degree of the isogeny of smallest degree connecting the two given curves and T the size of the subgroup with known torsion point images. Depending on d, this requires images on a smaller subgroup compared to recent SIDH attacks which allow to compute a connecting isogeny from the images without requiring the endomorphism rings [CD23; Rob23]. Further, an updated version of the reduction by Fouotsa, Kutas, Merz and Ti shows that the reduction still applies if the images of a slightly larger subgroup are given only up to an unknown scalar [FKMT21, Theorem 4.2] - a setting where the SIDH attacks are not known to apply. Thus, when images under the sought after isogeny are available for a sufficiently large subgroup, the complexity of computing an isogeny of degree d could be computed efficiently after computing the endomorphism rings in $O^*(p^{1/2})$.

When d is larger than p, the isogeny is usually not unique. One can compute the endomorphism ring of both curves in time $O^*(p)$, and a connecting ideal in polynomial time. If $d > p^3$ and d has at least two factors, one can then use a variant of KLPT algorithm [KLPT14] to compute an ideal of the correct norm in the same class, and then translate this to some representation of an isogeny.

Quantum algorithms

Some of the previously mentioned algorithms can be accelerated using quantum computation.

Utilising Grover's search [Gro96], the endomorphism ring computation from [EHLMP20] can be run in $O^*(p^{1/4})$ time and constant memory. Similarly, Biasse, Jao and Sankar showed how to accelerate the Delfs–Galbraith algorithm to run in $O^*(p^{1/4})$ [BJS14]. Note that this algorithm can be used to find a connecting isogeny of arbitrary degree but also to find loops in the isogeny graph, i.e. to compute endomorphisms.

To compute degree-d isogenies between two supersingular elliptic curves, Grover's quantum search algorithm [Gro96] brings the complexity of the exhaustive search over all degree-d isogenies, e.g. if d is prime, to $O^*(\sqrt{d})$ with constant memory.

For a sufficiently smooth degree d, Tani's claw finding algorithm with complexity $d^{1/3}$ [Tan09] has been suggested, but been widely dismissed since. For instance, Jaques and Schanck pointed out that Tani's algorithm assumes unrealistic costs of accessing memory. They argued that it is more efficient to use the classical hardware dedicated to access memory for Tani's algorithm for a classical attack instead [JS19]. In particular, Tani's algorithm does not seem to lead to a quantum speed-up.

6.2 Solving the norm equation with Cornacchia's algorithm

In this section, we describe a new and improved algorithm of finding elements of prescribed norm in the connecting ideal between the endomorphism rings of the two given supersingular elliptic curves. Our method is based on Cornacchia's algorithm and solves the following problem for $d = p^{1/2+\epsilon}$ for some $\epsilon > 0$.

Problem 6.6. *Let $\mathcal{O}, \mathcal{O}'$ be maximal orders in the quaternion ramified at p and ∞, $B_{p,\infty}$ and let I be a connecting ideal of $\mathcal{O}$ and $\mathcal{O}'$. Find an element of reduced norm d in I, if it exists.*

To solve Problem 6.6, we compute the reduced norm form Q of I. Then, finding an element of reduced norm d in I means solving the equation

$$Q(x_1, x_2, x_3, x_4) = d. \tag{6.2}$$

The ideal I is a four-dimensional lattice, say with basis $\phi_1, \ldots, \phi_4$, and we are looking for x_i such that $Q(x_1, x_2, x_3, x_4) = \|\sum_{i=1}^{4} x_i \phi_i\| = d$. Now, as long as we consider Q with respect to a reduced basis of I, we can compute bounds for the x_i. More precisely, for an LLL reduced basis $\phi_1, \ldots, \phi_4$, one has the bound $|x_i| \leq c\frac{d}{\|\phi_i\|}$, where c is a (small) constant that depends on the parameters used in the LLL reduction.

More explicitly, let $G := \left(g_{ij}\right)$ be the Gram matrix of the LLL- or Minkowski-reduced lattice basis $\phi_1, \ldots, \phi_4$ of the connecting ideal I, where $g_{ij} = \langle \phi_i, \phi_j \rangle$. Using the reduced norm and that the basis is close to orthogonal, we can estimate the sizes for $i \in \{1, \ldots, 4\}$: $\|\phi_i\| = p^{\alpha_i}$ such that $\sum_{i=1}^{4} \alpha_i \approx 2$. Furthermore, we assume $\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \alpha_4$ so that $\alpha_1 \leq \frac{1}{2}$.

Let $d = p^{1/2+\epsilon}$ for some $0 < \epsilon < 1/2$. To solve Problem 6.6, the goal is to find an element $x = (x_1\ x_2\ x_3\ x_4) \in I$ such that $xGx^T = d$, i.e. $|x| = d = p^{1/2+\epsilon}$. Since $\phi_1, \ldots, \phi_4$ is a reduced basis, we can bound the sizes of the x_i as described above by

$$|x_i| \leq c \cdot p^{\frac{1}{2}+\epsilon-\alpha_i}$$

for some small constant c.

In general, solving Diophantine equations like $Q(x_1, \ldots, x_4) = d$ with four variables can be daunting. Thus, our strategy in this section is to first guess two of the variables and then solve the remaining bivariate equation using Cornacchia's algorithm. In Section 6.3, we will describe an alternative approach using Coppersmith's methods and some of its variants to solve the multivariate equations resulting from guessing two or less variables.

We make random guesses for the two variables x_3 and x_4 in Equation (6.2), which results in a quadratic bivariate equation to be solved. Note that by our labeling ϕ_3, ϕ_4 are a priori the largest vectors in the LLL-reduced basis, hence bounds on x_3 and x_4 are smaller

than the bounds on x_1 and x_2. Guessing these two variables will contribute to a factor $p^{1+2\epsilon-\alpha_3-\alpha_4}$ in the overall complexity, or a factor $p^{\frac{1}{2}+\epsilon-\frac{\alpha_3+\alpha_4}{2}}$ using Grover's quantum search algorithm [Gro96].

Assuming we guess $x_3 =: k$ and $x_4 =: l$ correctly, where $k \approx p^{\frac{1}{2}+\epsilon-\alpha_3}$ and $l \approx p^{\frac{1}{2}+\epsilon-\alpha_4}$, the remaining equation to be solved is[1]

$$\begin{aligned} f(x_1, x_2) &= Q(x_1, x_2, k, l) - d \\ &= g_{11}x_1^2 + g_{22}x_2^2 + 2g_{12}x_1x_2 && \text{(quadratic)} \\ &+ (2g_{13}k + 2g_{14}l)x_1 + (2g_{23}k + 2g_{24}l)x_2 && \text{(linear)} \\ &+ \big(2g_{34}kl + g_{33}k^2 + g_{44}l^2 - d\big). && \text{(constant)} \end{aligned}$$

Performing a change of variables similar to [SSW08] (attributed to Lagrange) then allows us to rewrite $f(x_1, x_2) = 0$ as an equation of the form

$$x^2 - Dy^2 = N \tag{6.3}$$

due to the following.

Let f_{ij} denote the coefficient of $x^i y^j$ in f. The bivariate quadratic f can, in a first step, be transformed into an equation of the form

$$Dy^2 = (Dx_2 + E)^2 + DF - E^2, \tag{6.4}$$

where the new variable y is defined as $y := 2f_{20}x_1 + f_{11}x_2 + f_{10}$ and the substitutions

$$D := f_{11}^2 - 4f_{20}f_{02},$$

[1] More precisely, f is a family of functions $f_{k,l}$ where each function depends on the specific values guessed for x_3 and x_4. To improve notation/readability, we implicitly assume that f (and the values D, E, F, x, y and N) all depend on the k and l which are considered in the context where f (and the changed variables) are used.

$$E := f_{11}f_{10} - 2f_{20}f_{01}, \text{ and}$$
$$F := f_{10}^2 - 4f_{20}f_{00}$$

are made. In a second step, the introduction of the new variable $x := Dx_2 + E$ facilitates another rearrangement from Equation (6.4) into the desired form

$$x^2 - Dy^2 = N, \tag{6.3}$$

if we define N as $N := E^2 - DF$.

Examining the coefficient values in our new quadratic equation, Equation (6.3), obtained from the change of variables leads us to several observations: Firstly, we can see that the size of N can be bounded polynomially in the absolute value of the largest entry of G (more precisely $N \in O(\max\{(g_{ij})^4\}p^{1+2\epsilon-\alpha_3-\alpha_4})$. Secondly, we note that $D = -4\left(g_{11}g_{22} - g_{12}^2\right)$ is always negative as a consequence of the symmetric and positive semi-definite nature of the Gram matrix G. Hence, Equation (6.3) has only finitely many solutions. In particular, when looking for a fixed-degree isogeny, we expect there to usually only be a unique solution. Either way, we only require a single solution to obtain the desired isogeny.

Such a solution can be found using Cornacchia's algorithm (see e.g. [Nit95, Algorithm 1]) under the condition that N does not have too many prime factors, as it requires finding (all) square roots of $D \pmod N$. Finding these square roots becomes expensive if N has too many distinct factors. More precisely, we choose to abandon a pair of guesses x_3, x_4 when factoring N reveals that N has more than $B \log\log N$ distinct prime factors for some fixed $B \in \mathbb{Z}$. To estimate the probability of this event, we use the following lemma.

Lemma 6.7. *Let N be an integer as in Equation (6.3) and let $B \in \mathbb{Z}_{>1}$. Under the heuristic assumption that the number of prime divisors of N behave as predicted by standard asymp-*

totics for sufficiently large integers, we expect N to have more than $B \log\log N$ prime factors with probability smaller than $\frac{1}{2(B-1)^2}$.

Proof. Let $\omega : \mathbb{N} \to \mathbb{N}$ be the function which maps a positive integer to its number of distinct prime divisors. Asymptotically, the distribution of $\omega(n)$ is a normal distribution around the mean $B_1 + \log\log n$, where $B_1 \approx 0.261$ is the Mertens constant, with standard deviation $\log(\log n)^{1/2}$, see e.g. [HW79, Section 22.11] or [ROW94].

Under the heuristic that N is large enough for these asymptotics to apply and that its number of prime factors behaves as predicted for a random integer of roughly the same size, we can use Chebyshev's inequality to get the bound

$$\Pr(\omega(N) - B_1 > B \log\log N) \leq \frac{1}{2B^2 \log\log N}.$$

Here, we used that the normal distribution is symmetric around $B_1 + \log\log N$ and that the standard deviation is $\log(\log N)^{1/2}$. Since for N interesting for our application $\log\log N > 1$, we can very crudely estimate our bound by

$$\Pr(\omega(N) > B \log\log N) \leq \frac{1}{2(B-1)^2}.$$

□

Note that taking a larger B to bound the number of prime factors of N, $B \log\log N$, accepted in our algorithm may increase the cost to run Cornacchia's algorithm.

Remark 6.8. *Assume that the asymptotic heuristics hold for all the N sampled by fixing x_3, x_4 for a fixed basis and assume that the correct solution is randomly distributed among these trials. Then by Lemma 6.7 taking for instance $B = 11$, we expect to find a solution in $> 99\%$ of cases after iterating through all guesses for a fixed basis. However, it may be*

possible that the correct solution $(x_1, \ldots, x_4)$ *with respect to some fixed basis gives an* N *with too many prime factors. We accept this as the* failure probability *of our algorithm.*

These observations lead us to the following theorem:

Proposition 6.9. *Assume* $\omega(N) \leq B \log\log N$*, i.e.* N *is not too smooth and has fewer than has at most* $B \log\log N$ *prime factors. One can find a solution to the equation* $f(x_1, x_2) = 0$ *in quantum polynomial time, if it exists, or determine that there is no such solution.*

Proof. The main observation is that since G is a positive definite matrix, its leading principal minors are positive. Hence we have that $g_{12}^2 - g_{11}g_{22} < 0$ which implies that $D = (2g_{12})^2 - 4g_{11}g_{22} < 0$. Therefore, it is possible to use the above change of variables to reduce solving $f(x_1, x_2) = 0$ to solving $x^2 - Dy^2 = N$ where the size of N is polynomial in the size of G (i.e. the size of the absolute value of the largest entry). One can use Shor's algorithm [Sho97] for factoring N and then apply Cornacchia's algorithm for solving $x^2 - Dy^2 = N$. Reversing the substitutions leads to a solution to $f(x_1, x_2) = 0$. Note that if a guess k, l is incorrect, then $f(x_1, x_2) = 0$ will have no solution. Fortunately, running Cornacchia's algorithm helps us detect efficiently if no solution exists by [Coh13, Section 1.5.2]. □

Theorem 6.10. *Let* $\mathcal{O}, \mathcal{O}'$ *be maximal orders in* $B_{p,\infty}$*. Let* $d \approx p^{1/2+\epsilon}$ *for some* $\epsilon > 0$ *and let* $\phi_1, \phi_2, \phi_3, \phi_4$ *be an LLL-reduced basis of the connecting ideal* I *such that* $\|\phi_i\| = p^{\alpha_i}$ *and* $\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \alpha_4$*. Then Algorithm 6 is a quantum algorithm that computes an element of reduced norm* d *in* I*, i.e. solves Problem 6.6 for the given parameters, in time* $O^*(p^{1/2+\epsilon-\alpha_3/2-\alpha_4/2})$ *or returns no solution. The algorithm fails to find an existing solution with probability smaller than* $1/2(B-1)^{-2}$ *under the heuristics of Lemma 6.7, where the probability is taken over the possible choices of LLL-reduced lattices and* $B \log\log N$ *is the number of prime factors allowed in Step 4 of Algorithm 6.*

Algorithm 6: Recovering an element of reduced norm d in connecting ideal I using Cornacchia

Input: Let $\mathcal{O}, \mathcal{O}'$ be maximal orders in $B_{p,\infty}$ and let I be their connecting ideal containing an element of reduced norm d, where $d \approx p^{1/2+\epsilon}$. Let $\phi_1, \phi_2, \phi_3, \phi_4$ be an LLL-reduced basis of I with $\|\phi_i\| = p^{\alpha_i}$ and $\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \alpha_4$. Finally, let $G = (g_{ij})$ be the corresponding Gram matrix and $B \in \mathbb{Z}_{>1}$.

Output: $x_1, x_2, x_3, x_4 \in \mathbb{Z}$ such that $Q(x_1, x_2, x_3, x_4) = \|\sum_{i=1}^{4} x_i \phi_i\| = d$.

for $(k, l) \in \{0, \pm 1, \ldots, \pm c \cdot p^{1/2+\epsilon-\alpha_3}\} \times \{0, \pm 1, \ldots, \pm c \cdot p^{1/2+\epsilon-\alpha_4}\}$ **do**
  $D \leftarrow 4(g_{12}^2 - g_{11}g_{22})$, $E \leftarrow 4(g_{12}(g_{13}k + g_{14}l) - g_{11}(g_{23}k + g_{24}l))$,
  $F \leftarrow 4((g_{13}k + g_{14}l)^2 - g_{11}(2g_{34}kl + g_{33}k^2 + g_{44}l^2 - d))$, $N \leftarrow E^2 - DF$;
  Factor N using Shor's algorithm;
  if N has more than $B \log\log N$ factors **then**
    continue
  else
    Run Cornacchia's algorithm to find solutions of $x^2 - Dy^2 = N$;
    if Cornacchia returns a solution (x, y) **then**
      $x_2 \leftarrow (x - E)D^{-1}$, $x_1 \leftarrow (2g_{11})^{-1}(y - 2(g_{12}x_2 + g_{13}k + g_{14}l))$;
      $x_3 \leftarrow k$, $x_4 \leftarrow l$;
      return x_1, x_2, x_3, x_4

Proof. Guessing two integer values for x_3 and x_4, with the restriction that $|x_i| \leq c \cdot p^{\frac{1}{2}+\epsilon-\alpha_i}$, results in (a worst-case number of) $4c^2 p^{1+2\epsilon-\alpha_3-\alpha_4}$ combinations to try.

Each of these tries consists of first computing the change of variables using the coefficients of the corresponding function f to obtain Equation (6.3). This requires a constant number of multiplications and additions of values polynomially bounded by the size of the largest entry by absolute value of the Gram matrix G. The time required for solving the resulting equation is closely related to the size and number of prime factors of N. In particular, factoring N using Shor's quantum algorithm takes $O\left((\log N)^3\right)$ and reveals the prime factors of N. When N is not too smooth for the correct values for x_3 and x_4 with respect to a fixed LLL-reduced basis of I, the algorithm recovers x_1 and x_1 correctly in quantum polynomial time by Proposition 6.9. More explicitly, for N not too smooth, the first step of Cornacchia's algorithm finds all square roots of D modulo N which can be done, for exam-

ple, using the Tonelli–Shanks algorithm for composite moduli. Each of these square roots is then reduced against N using the Euclidean algorithm, until one of these reductions yields a solution to Equation (6.3). Running the division algorithm and checking whether the result is a solution takes polynomial time, but the number of square roots to check is exponential in the number of distinct prime factors of N.

If Cornacchia's algorithm fails to provide a solution, the same procedure is repeated with new guesses for x_3 and x_4. However, if the algorithm is successful, we reverse the change of variables and obtain solutions x_1 and x_2, which together with guesses x_3 and x_4, give us an element in the connecting ideal.

If N is too smooth for the correct guesses of x_3 and x_4 with respect to the fixed LLL-reduced basis of I, the algorithm skips to run Cornacchia's algorithm and hence will fail to find the solution. Assuming that the number of factors of N for the correct solution with respect to different choices of LLL-reduced bases are independent, the probability of picking an LLL-reduced basis for which N is too smooth is less than $\frac{1}{2(B-1)^2}$ by Lemma 6.7 which gives the failure probability. We discuss the plausibility of the assumption of independence in Remark 6.11. □

Remark 6.11. *The analysis of Theorem 6.10 raises the question whether the correct guesses for x_3 and x_4 with respect to different bases leads to N with distinct prime factorisation respectively. Experimentally, we re-randomised multiple bases using unimodular matrices and indeed the resulting N corresponding to the correct guesses with respect to the respective bases were different, did in general neither share the same factors nor have the same number of distinct prime factors.*

Similarly, one could also just guess values for a different pair of x_i (instead of x_3 and x_4) to obtain a different N. For basis vectors all of size roughly $p^{1/2}$ this would not affect

the complexity of the algorithm.

Remark 6.12. *Algorithm 6 relies on quantum computations for factoring the integer N only. The remaining steps are performed using classical computation. However, to obtain the complexity statement made in Theorem 6.10 the amplitude amplification of Grover's quantum search over the guesses is used.*

6.3 Solving the norm equation with Coppersmith's methods

In this section, we describe a slightly different approach to solve Problem 6.6 of finding elements of prescribed norm in the connecting ideal of two maximal quaternion orders. The first step is the same as in Section 6.2: we compute the reduced norm form with respect to an LLL-reduced basis, and our goal is still to represent the integer d. Using the notation from the previous section we are thus looking for x_1, x_2, x_3, x_4 such that $Q(x_1, x_2, x_3, x_4) = d$ and we have the same bounds on the x_i as before.

As an alternative to solving the equation using Cornacchia's algorithm, in the following we apply several variants of Coppersmith's techniques to compute short solutions of polynomial equations. In particular, we make guesses for one or multiple of the x_i and then solve the remaining equations leading us to either discard the guess and guess anew, or to finding a representation of d. One could also directly solve the equation in four variables; the analysis of this strategy remains for future work.

We split this section into parts, first providing theoretical analyses highlighting for which isogeny degrees we expect our methods to work and later providing experimental results to support our estimates.

6.3.1 Guessing two variables

Again, we assume the same setup as in Section 6.2. Recall that $G = (g_{ij})$ is the Gram matrix of the reduced norm form of the ideal I with corresponding basis $\{\phi_i\}_{i=1}^4$. As before, we have size estimates for $i \in \{1, \dots, 4\}$: $\|\phi_i\| = p^{\alpha_i}$ such that $\sum_{i=1}^4 \alpha_i = 2$. We also assume for simplicity that $\alpha_i \approx 1/2$, which is the generic case [GPST16].

Now, we guess two variables as in the previous section, but we would like to solve the remaining bivariate quadratic equation utilising Coron's approach to Coppersmith's methods instead of Cornacchia's algorithm. We recall the quadratic polynomial that we would like to solve for guesses $k = x_3$ and $l = x_4$.

$$\begin{aligned} f(x_1, x_2) &= Q(x_1, x_2, k, l) - d \\ &= g_{11}x_1^2 + g_{22}x_2^2 + 2g_{12}x_1x_2 && \text{(quadratic)} \\ &+ (2g_{13}k + 2g_{14}l)x_1 + (2g_{23}k + 2g_{24}l)x_2 && \text{(linear)} \\ &+ \big(2g_{34}kl + g_{33}k^2 + g_{44}l^2 - d\big). && \text{(constant)} \end{aligned}$$

Also recall our size estimate for d: $d \approx p^{1/2+\epsilon}$ where $\epsilon > 0$. Since we assume that we start with a reduced basis, we have that for the correct solution $|x_i| < p^\epsilon$, hence the same is true for every guess k, l. Then Theorem 6.4 with $\delta = 2$ as in our case implies that Coron's algorithm is able to find a solution to this equation (or detect that no solution exists) if $XY < W^{1/2}$ where $|x_1| < X$,$|x_2| < Y$ and $W = \max\{|f_{ij}|X^iY^j\}$ for f_{ij} the coefficient of $x_1^i x_2^j$. Since the Gram matrix is reduced it follows from our assumptions that $g_{ij} \approx \sqrt{p}$ which in turn implies that $W \approx p^{1/2+2\epsilon}$, and therefore $X \approx Y \approx p^\epsilon$. The condition $XY < W^{1/2}$ on the bound ensuring that we can find the small integer roots of f means that we can state a

condition on ϵ to ensure the same. More explicitly, we require that

$$p^{2\epsilon} < p^{1/4+\epsilon},$$

so that Coron's algorithm will be successful for $\epsilon < 1/4$, hence for ideals with norms between $p^{1/2}$ and $p^{3/4}$.

We summarise our results in the following theorem.

Theorem 6.13. *Let $\mathcal{O}, \mathcal{O}'$ be maximal orders in $B_{p,\infty}$. Let $d \approx p^{1/2+\epsilon}$ for some $0 < \epsilon < 1/4$. Further, let $\phi_1, \phi_2, \phi_3, \phi_4$ be an LLL-reduced basis of the ideal I connecting $\mathcal{O}$ and $\mathcal{O}'$ such that $\deg(\phi_i) = p^{\alpha_i}$ and $\alpha_i \approx 1/2$ for alls i. Then there exists a quantum algorithm that computes an element of reduced norm d in I in time $O(p^{1/4})$ or determines that no such element exists.*

The cost of the entire algorithm is exactly the same as in the previous section as it is dominated by guessing two of the variables and the endomorphism ring computations. The advantage of this approach using Coron's algorithm is that in comparison to the method described in Theorem 6.10, it has no failure probability and thus does not rely on non-standard heuristics (other than that the shortest element in $\mathrm{Hom}(E, E')$ has degree approximately $\sqrt{p}$).

6.3.2 Guessing one variable

Let us finally consider the case of guessing only a single variable $l = x_4$. We now sketch for which sizes of degrees $d = p^{1/2+\epsilon}$ we expect this method to produce a valid solution to Problem 6.6.

Using our previous notation, we have $W = Q(x_1, x_2, x_3, x_4) = d = p^{1/2+\epsilon}$, where Q is again considered with respect to a reduced basis. In this case, we have $x_i \approx d/\|\max\{g_{ij}\}\| \approx$

p^ϵ, as for a generic basis all of the vectors will be roughly of size $p^{1/2}$.

Due to the symmetry in the set of monomials appearing in the norm equation, we focus on sets S that are invariant under permutations of variables; see Table 6.1 for examples of S. Using the notation of [BJ07] we introduced in Section 6.1.1, we find that $s_x = s_y = s_z$ for these S.

Neglecting the LLL approximation factors, which depend on the parameters used in the computation of the LLL basis and asymptotically only contribute a small constant, Theorem 6.5 then implies that we can find a solution of the trivariate polynomial $Q(x_1, x_2, x_3, k) = d$ as long as

$$X^{3s_x} < W^s. \tag{6.5}$$

Using the estimate $|x_i| \approx p^\epsilon$ to give a bound X and $W = p^{1/2+\epsilon}$, we find that Equation (6.5) is equivalent to

$$3s_x\epsilon < (1/2 + \epsilon)s.$$

Therefore, we expect the Bauer–Joux algorithm to provide us with a solution whenever

$$\epsilon < \frac{s}{2(3s_x - s)}. \tag{6.6}$$

Table 6.1 provides values for s, $s_x = s_y = s_z$ and $\frac{s}{2(3s_x-s)}$ for some a priori plausible symmetric sets S. The first row already provides a suitable set S, containing monomials of total degree up to $D = 1$. We then consider two examples of sets S for $D = 2$ and in the last row give formulae and estimates for sets S for increasing D. We remark that while increasing D ostensibly improves the estimate for the bound (6.6) on ϵ, the matrix M_1 defining the lattice L_1 grows significantly resulting in much slower LLL reduction. For efficiency resasons, despite the algorithm still technically being polynomial-time for any fixed D, we keep $D = 1$ in practice.

Symmetric set S	s	$s_x = s_y = s_z$	$\frac{s}{2(3s_x - s)}$
$\{1, x, y, z\}$	4	14	0.05263
$\{1, xy, xz, yz\}$	4	26	0.02703
$\{1, x, y, z, xy, xz, yz\}$	7	28	0.04545
monomials with total degree $\leq D$	$\sum_{i=0}^{D} \binom{i+2}{2}$	$\sum_{i=0}^{D+1}(D+2-i)\binom{i+1}{1} + \sum_{i=0}^{D}(D+1-i)\binom{i+1}{1}$	$\to 0.1$

Table 6.1: Values for plausible symmetric sets.

This approach requires us to run Coppersmith's algorithm once per guess $l = x_4$. Depending on the given parameters, a trade-off between broadening the range of applicability by adding more monomials, i.e. increasing D, and the rise in complexity stemming from a larger set S can be considered.

6.3.3 Experimental results

In our experiments of solving the norm equation we used MAGMA [MAGMA] to generate maximal orders and connecting ideals containing an element with increasing reduced norm from random walks. We then transformed them into the corresponding quadratic forms. This method immediately yields the solution to our problem, knowledge of which we utilise to avoid guessing when working with large parameters to reduce computation times. Instead, we pick one variable (resp. two variables) we consider known, i.e. correctly guessed, and then use our implementations of Coppersmith's methods to solve the form for the remaining three (resp. two) variables. We can then compare the thus obtained solution with the known solution to check correctness. Note that once one of the algorithms has found enough additional polynomials, we try to obtain the unknown root by computing resultants which simultaneously checks for algebraic dependence. As with many lattice reduction applications, we observe that the approaches seem to work better in practice than in theory.

Our experimental results for the bivariate case are presented in Tables 6.2 to 6.4. For

each of the three primes we generated using SageMath [SAGE], we observe that Coron's bivariate approach works for ideal norms until approximately $p^{0.8}$, i.e. approximately $2^{0.8\ell}$, where ℓ is the bit length of p. Note that the we only include ideal norms in the tables which allow us to observe when the rate of success starts to decrease from 100% and at which point it drops close to zero.

Ideal norm	**Instances run**	**number of successes**
2^{208}	100	100
2^{209}	100	76
2^{210}	100	29
2^{211}	100	18
2^{212}	100	9
2^{213}	100	4

Table 6.2: Experiments for a 253-bit prime p.

Ideal norm	**Instances run**	**number of successes**
2^{245}	100	100
2^{246}	100	77
2^{247}	100	34
2^{248}	100	18
2^{249}	100	15
2^{250}	100	0

Table 6.3: Experiments for a 300-bit prime p.

Our SageMath implementations of the trivariate Bauer–Joux approach and the trivariate Coron approach find connecting ideals between two maximal orders $\mathcal{O}$ and $\mathcal{O}'$ containing an element of reduced norm up to approximately $2^{0.67\ell}$ where ℓ is the bit-length of the prime p. We tested our implementations on large primes ranging from 253 to 503 bit-length. The primes include the two 254-bit primes p_{6983} and p_{3923} from [DKLPW20; DLLW23] as well as the first two primes used in SIKE [JACCDHJKLLNRSU17] which have bit-lengths of 434 and 503 respectively.

Ideal norm	**Instances run**	**number of successes**
2^{204}	100	100
2^{405}	100	99
2^{406}	100	72
2^{407}	100	32
2^{408}	100	5
2^{409}	100	0

Table 6.4: Experiments for a 500-bit prime p.

As already discussed in Section 6.1.1, we avoid a Gröbner basis computation and further LLL reduction by using a slightly altered version of the original Bauer–Joux algorithm. This alteration can for example be found in [BVZ12], and entails using other LLL-reduced and orthogonalised vectors $(b_1^\star, \ldots, b_{r-1}^\star)$ in reverse order, in addition to $b_r^\star$. In particular we check if any of the $b_i^\star$ already yields another polynomial P_2 which annihilates the desired root and is algebraically independent from P and the polynomial P_1 obtained from $b_r^\star$. As with $b_r^\star$, this is guaranteed if $\|s_0\| < \|b_i^\star\|$ but can happen even if the condition is not fulfilled. If we are successful, we immediately proceed to extracting the root and skip the second, costly step of the Bauer–Joux algorithm entirely.

Our experimental results for the trivariate case are presented in Tables 6.5 to 6.8, and show the successes of both implementations among a hundred randomly generated instances of quadratic forms and their roots. For each of the four selected primes, we observe that both trivariate approaches work for ideal norms until approximately $p^{0.66}$, i.e. approximately $2^{0.66\ell}$ where ℓ is the bit length of p. Again, we only display meaningful experimental results in the tables by only including the range of ideal norms which shows relevant changes to the rate of success for either algorithmic approach.

Ideal norm	Instances run	Coron success	Bauer–Joux success
2^{169}	100	100	100
2^{170}	100	100	96
2^{171}	100	84	0
2^{172}	100	27	0
2^{173}	100	2	0
2^{174}	100	0	0

Table 6.5: Experiments for p_{3923} from SQISign.

Ideal norm	Instances run	Coron success	Bauer–Joux success
2^{170}	100	100	100
2^{171}	100	88	100
2^{172}	100	26	0
2^{173}	100	4	0
2^{174}	100	2	0
2^{165}	100	0	0

Table 6.6: Experiments for p_{6983} from SQISign.

6.4 The order embedding problem

Though the main focus of this chapter lies on improving isogeny finding algorithms via speed-ups for solving Problem 6.6, our methods are naturally applicable to a slightly different algorithmic problem where one wants to find an element of prescribed trace and norm inside a maximal order $\mathcal{O}$, i.e. the following problem.

Problem 6.14 (Order embedding problem)**.** *Let $\mathcal{O}$ be a maximal order in $B_{p,\infty}$ for some prime p and let D be a quadratic order. Decide whether D embeds into $\mathcal{O}$ and find this embedding if it exists.*

There are several reasons why this problem is interesting. On the one hand there is a natural connection between this problem and finding connecting ideals of a given norm. For

Ideal norm	Instances run	Coron success	Bauer–Joux success
2^{288}	100	100	100
2^{289}	100	100	82
2^{290}	100	84	0
2^{291}	100	26	0
2^{292}	100	3	0
2^{293}	100	0	0

Table 6.7: Experiments for SIKEp434.

Ideal norm	Instances run	Coron success	Bauer–Joux success
2^{333}	100	100	100
2^{334}	100	100	15
2^{335}	100	71	0
2^{336}	100	22	0
2^{337}	100	1	0
2^{338}	100	0	0

Table 6.8: Experiments for SIKEp503.

example, it is easy to see that finding a connecting ideal to the endomorphism ring of the curve $E : y^2 = x^3 + x$ of norm d is closely related to finding an embedding of the quadratic order $Z[d\iota]$ where ι is the non-trivial automorphism of the curve E with $j(E) = 1728$. On the other hand, the order embedding problem plays an important role in the reductions of [Wes22a]. Namely, Problem 6.14 is the missing link in relating the *Uber isogeny problem* [DDFKLPSW21, Problem 5.1] and the *endomorphism ring problem* [Wes22a, Problem 6].

Informally, the *Uber isogeny problem* is the following. One is given two D-oriented curves and one has to find a connecting ideal class between them. The key recovery problem in many isogeny-based schemes can be reduced to this problem [DDFKLPSW21]. A particular example of this is the key recovery in CSIDH [CLMPR18] and its relation to the general isogeny problem: If the discriminant of the quadratic order is large enough, we expect D to be embedded in every maximal order. Hence, finding the desired ideal class would solve the

pure isogeny problem of finding any isogeny between the two given curves.

For simplicity, we will assume that the element we are looking for has trace zero, i.e. we would like to embed $\mathbb{Z}[\sqrt{-d}]$ into $\mathcal{O}$. First, one can compute the $\mathbb{Z}$-lattice of trace-0 elements which is known to be a rank-3 lattice of determinant p^2. If $d < p^{2/3}$ one can usually find this element by computing the shortest element in the lattice. This approach is no longer useful when d is substantially bigger than $p^{2/3}$. However, since we are working with a rank-3 lattice, the trivariate approaches described in Section 6.3.2 can be applied. For efficient computations, we are only interested in polynomial-time algorithms and will hence refrain from investigating the complexity of first guessing one variable and then applying one of our bivariate approaches; deducing a running time should nevertheless be straightforward. The results presented below are heuristic but more rigorous bounds could potentially be achieved using the Bauer–Joux approach.

Experimental results

For our experiments, we generated problem instances in the following way. First, we computed a random maximal order in $B_{p,\infty}$. This can be accomplished by starting from a standard maximal order and taking a random walk of length $\log p$. Then we computed a basis for the trace-0 part of the order and found a reduced basis of this lattice. From this basis we generated the corresponding quadratic form. We then chose random values for x_1, x_2 and x_3 of bounded size and checked whether the Coron or Bauer–Joux algorithms from Section 6.3 were able to recompute the (known) solution. We used this approach in our experiments.

Remark 6.15. *An alternative procedure to the one above is as follows: We fix some order $\mathbb{Z}[\sqrt{-d}] =: \mathcal{O}_0$ and find a maximal order $\mathcal{O}$ containing it. This can be accomplished by embedding $\mathcal{O}_0$ into the quaternion algebra $B_{p,\infty}$ through finding rational solutions (x, y, z) to the equation $x^2 + py^2 + pz^2 = d$. Given d in terms of its prime factors, we can use the*

algorithm from [Sim05] which is conveniently implemented in PARI/GP [PARI/GP] to solve the equation over $\mathbb{Q}$. It remains to compute a maximal order containing this element. Thus, we have constructed a maximal order which we know is oriented by D.

We experimented with 3 different primes of varying sizes and we ran multiple instances for each different discriminant size of the orders. In Tables 6.9 to 6.11 we present our findings on when and how often we succeeded in computing the embeddings.

Discriminant size	**Instances run**	**Number of successes**
2^{188}	100	100
2^{189}	100	58
2^{190}	100	7
2^{191}	100	1
2^{192}	100	0

Table 6.9: Experiments for a 256-bit prime p.

Discriminant size	**Instances run**	**Number of successes**
2^{316}	100	100
2^{317}	100	100
2^{318}	100	99
2^{319}	100	43
2^{320}	100	11
2^{321}	100	4
2^{322}	100	1
2^{323}	100	0

Table 6.10: Experiments for a 434-bit prime p.

Based on our experiments, we conjecture that the approach outlined in this section works for discriminants of size $p^{0.8}$.

Discriminant size	Instances run	Number of successes
2^{485}	100	100
2^{487}	100	100
2^{489}	100	95
2^{491}	100	23
2^{493}	100	4
2^{495}	100	3
2^{497}	100	0

Table 6.11: Experiments for a 610-bit prime p.

6.5 Solving the degree-d isogeny problem for supersingular elliptic curves

Let E and E' be two supersingular elliptic curves defined over the field $\mathbb{F}_{p^2}$ connected by an unknown isogeny of degree d. We now briefly give a strategy for how finding an element of reduced norm d in the ideal connecting the endomorphisms rings of E and E', $\mathcal{O}$ and $\mathcal{O}'$, leads to solving Problem 6.1 by finding a degree-d isogeny $E \to E'$.

To recover the unknown isogeny starting from knowledge of just the two curves and the specified degree d, we need to first compute the endomorphism rings $\mathcal{O}$ and $\mathcal{O}'$. This can be done classically in time $O^*(p^{1/2})$ or on a quantum computer in $O^*(p^{1/4})$ using the algorithm by Eisenträger et al. [EHLMP20] and its quantum version, or alternatively the Delfs–Galbraith [DG16] algorithm. After the endomorphism rings have been recovered, we compute any connecting ideal I for $\mathcal{O}$ and $\mathcal{O}'$. This can be done efficiently, for instance using the algorithm of Kirschmer and Voight [KV10].

Given the connecting ideal I, we next have to recover an element of reduced norm d in I. This means solving Problem 6.6 and we described methods for finding solutions in the previous sections. More explicitly, in Section 6.2 we described an algorithm which solves

Problem 6.6 for $d \approx p^{1/2+\epsilon}$, $\epsilon > 0$ in roughly $O^*(p^{1/4}+\epsilon)$ with high probability on a quantum computer or returns no solution; see Theorem 6.10. As an alternative, we presented another algorithm based on a method by Coppersmith's for bivariate equations in Section 6.3. This solves Problem 6.6 for $d \approx p^{1/2+\epsilon}$, $0 < \epsilon < 1/4$ in roughly $O^*(p^{1/4})$ on a quantum computer; see Theorem 6.13.

Once the element of reduced norm d in I has been recovered, we can compute a connecting ideal of $\mathcal{O}$ and $\mathcal{O}'$ of norm d by [KLPT14, Lemma 5]. Under Deuring's correspondence [Deu41], this ideal then corresponds to an isogeny $\varphi : E \to E'$ of degree d. To make this correspondence effective, it remains to compute the corresponding isogeny from the ideal. That is, to convert the quaternion representation of the isogeny into a more usable form. For example, this could mean giving a kernel representation of the corresponding isogeny φ, i.e. writing down $\ker(\varphi)$ explicitly, or providing explicit rational maps defining the isogeny. Depending on smoothness of d and its relation to p, it might not be possible to represent the isogeny in these common forms efficiently, e.g. as the kernel might only be defined over a large field extension. If d is sufficiently smooth to allow for an efficient kernel representation of φ, this kernel representation can be recovered efficiently using tools described for example in [DKLPW20; DLLW23] or [EPSV23]. In case d is not smooth, say a large prime, our approach allows for an *efficient isogeny representation* in the sense that one can still evaluate an isogeny of degree d on any point $P \in E$ and compute $\varphi(P) \in E'$ using [FKMT22, Algorithm 1] or strategies from [Ler23]. [Rob22] also provides a strategy for evaluating isogenies of large prime-degrees where the isogeny is embedded into a higher-dimensional analogue to make allow for explicit computations.

The costs of using our various techniques to find fixed-degree isogenies are summarised in the following table.

Method	**Cost (classical)**	**Cost (quantum)**	**Condition on size**
Cornacchia (Section 6.2)	$\max\{\frac{1}{2}, 2\epsilon\}$	$\max\{\frac{1}{4}, \epsilon\}$	-
Coppersmith bivariate (Section 6.3.1)	$\max\{\frac{1}{2}, 2\epsilon\}$	$\max\{\frac{1}{4}, \epsilon\}$	$\epsilon < 1/4$
Coppersmith trivariate (Section 6.3.2)	$\max\{\frac{1}{2}, \epsilon\}$	$\max\{\frac{1}{4}, \frac{\epsilon}{2}\}$	$\epsilon < 0.16$
State-of-the-art (general d)	$\frac{1}{2} + \epsilon$	$\frac{1}{4} + \frac{\epsilon}{2}$	-
State-of-the-art (general d)	$\frac{1}{2}$	$\frac{1}{4}$	$\epsilon > 5/2$
State-of-the-art (smooth d)	$\frac{1}{4} + \frac{\epsilon}{2}$	$\frac{1}{4} + \frac{\epsilon}{2}$	-

Table 6.12: Summary of cost for finding isogenies via our different approaches and (empirical) conditions for the algorithms to work. The isogeny degree is given as $d = p^{1/2+\epsilon}$, and costs are provided as logarithms in base p.

In summary, if one is given two d-isogenous supersingular elliptic curves, the quaternion representation of the d-isogeny can be recovered in $O^*(\max\{p^{1/4}, p^{\epsilon}\})$, which can be transformed to a kernel representation whenever d is smooth enough. Previously, the best known algorithm to solve the problem of recovering a d-isogeny took $O^*(p^{1/4+\epsilon/2})$.

6.6 Improvements and outlook

In this chapter, we provided new algorithms to compute an isogeny of specified degree d between two given supersingular elliptic curves defined over a field of prime characteristic p, provided that such an isogeny exists.

Our strategy consisted of computing the curves' endomorphism rings and a connecting ideal thereof first, and then finding a representation of the integer d in terms of the

norm form of $\mathrm{Hom}(E, E')$. In particular, we improved upon previously known quantum algorithms in the setting where the isogeny degree $d \approx p^{1/2+\epsilon}$ satisfies $0 < \epsilon < 1/2$ by reducing the complexity from $O^*(p^{1/4+\epsilon/2})$ to $O^*(\max\{p^{1/4}, p^{\epsilon}\})$. To achieve this speed-up, we examined several methods of solving the norm equation. While using Cornacchia's algorithm as in Section 6.2 means accepting a small failure probability, the other techniques based on Coppersmith methods Section 6.3 work whenever the ideal norm is not too large. The approach where we guess two variables works whenever $0 < \epsilon < 1/4$ and the approach where one variable is guessed until $\epsilon < 1/6$. The performance of the latter approaches for different sizes of p are compared in Table 6.12. As we described how to reduce finding a fixed-degree isogeny to the problem of finding an element of prescribed reduced norm in an ideal connecting two maximal quaternion orders, we also examined the related problem of embedding an arbitrary quadratic order into a given maximal order, if such an embedding exists. Based on our experiments, we observe that the previously developed tools also succeed in finding such embeddings provided the size of the discriminant does not exceed $p^{0.8}$.

There are several avenues for follow-up research. Currently the trivariate Coppersmith methods do not provide an overall speed-up, just a better reduction from finding a degree-d isogeny to endomorphism ring computations. In order to achieve improvements over the bivariate approaches one could potentially combine the approach with guessing the most significant bits of the other three variables which should immediately improve the range of our results. Furthermore, one might also consider combining this approach with guessing part of the secret isogeny. This might lead to an improved algorithm (i.e. better than the bivariate approaches we use) for isogeny degrees slightly larger than $p^{3/4}$. We leave a precise investigation of these ideas to future research.

As we only provide an advancement when using quantum resources, it remains an important task in cryptanalysing isogeny-based schemes to examine and improve on the current classical complexity of solving the fixed-degree isogeny finding problem when no

further information is provided. Developing more efficient algorithms to solving multivariate, specifically four-variable, quadratic equations with relatively small integer solutions could further lower the quantum complexity of the problems in question.

Part III

Conclusion

CHAPTER 7

Conclusion

In this thesis, we demonstrated several cryptanalytic results regarding the security of SIDH, an SIDH-based NIKE due to Jao and Urbanik and the Genus-2 SIDH protocol, and further addressed the (quantum) hardness of the problem of computing general prescribed-degree isogenies. In particular, we first presented an attack framework which allows one to invert a certain type of one-way function in subexponential quantum time which can be instantiated to recover secret keys for overstretched and unbalanced SIDH. We additionally presented adaptive attacks on two SIDH variants, showing specifically that not only protocols can be adapted for using (isogenies of) higher-dimensional abelian varieties, but so can cryptanalytic methods. Our exploration of the G2SIDH protocol also included an improvement to the secret sampling algorithm brought about by a classification of kernel subgroups appearing in the procedure. Finally, we examined one of the fundamental isogeny problems, that of finding an isogeny of a specific degree between two supersingular elliptic curves, and provided new methods of solving this problem which utilise known quantum algorithms as subroutines.

Impact in light of the SIDH attacks and future directions Though the attacks presented in this thesis are now surpassed by the brilliant SIDH-attacks by Castryck–Decru and others, they advanced the understanding of (cryptanalysis of) SIDH-based schemes. Many of the ideas and techniques which emerged from the former attempts such as GPST or Petit's torsion point attacks and the recent full break have inspired further research, both into developing protocols and cryptanalysis as they provide useful tools and guidance when building new or assessing the security of other schemes.

For example, techniques from Chapters 4 and 5 can be utilised to examine the security claims of other isogeny-based schemes. The M-SIDH and MD-SIDH protocols respectively masking torsion point information or isogeny degrees in SIDH were proposed in response to the SIDH-attacks and though trust in SIDH-based but sufficiently altered schemes has not been restored, these protocols could benefit from an assessment of their active security. Furthermore, the very recently proposed *Fast Encryption from Supersingular Torsion Attacks* (FESTA) protocol [BMP23] should be inspected as it could show vulnerabilities to adaptive attacks via sending repeated malformed public information (when the OAEP transform is not applied).

The malleability oracle framework from Chapter 3 can be utilised as a building block for further cryptanalytic efforts. A further development of the SIDH group action we constructed was recently shown in [CIIKLP23] where the authors perform a quantum polynomial-time attack on the key exchange protocol pSIDH [Ler23]. It is possible that the hidden shift reduction framework and work building on it can be helpful in assessing security of other schemes.

Exploring isogeny-based cryptography with higher-dimensional abelian varieties also leaves many avenues for further research. While there have not yet been any proposals of schemes which directly originate from the properties of higher-dimensional isogenies, it also

remains an ongoing project to translate schemes more elaborate than SIDH and CGL to the two-dimensional setting. Multiple schemes would lend themselves to this exercise. So far, there has not yet been an adaptation of the CSIDH group action for endomorphism rings of Jacobians of genus-2 curves. In order to build a version of CSIDH working with general principally polarised abelian surfaces, more insight into higher-dimensional isogeny graphs, isogeny computation and other subroutines of the CSIDH protocol is required. A better understanding of the genus-2 isogeny graphs and properties of abelian surfaces will most likely not only advance the CSIDH-centred attempts of generalising protocols. For example, rigorously and explicitly expressing the Deuring correspondence for two-dimensional abelian varieties or designing an algorithm akin to KLPT in dimension two would be great accomplishments. Regarding cryptographic primitives, especially in the light of the (planned) submission of SQISign to NIST's signature standardisation project [NIST23] and the proposal of SQISignHD [DLRW23], it will be of interest to build more signature schemes from higher-dimensional isogenies.

For cryptanalytic purposes, speeding up the algorithms presented in this thesis by, for example, improving runtime of certain subroutines or coming up with alternative strategies, especially for those in Chapters 3 and 6, would be of interest. It is possible that some of the quantum subroutines could be avoided with classical computations of similar (or lower) complexity. It remains of utmost importance to isogeny-based cryptography to keep examining whether there exist better, i.e. faster or low(er)-memory, ways to compute the endomorphism ring of an arbitrary supersingular elliptic curve, compute arbitrary isogenies as well as isogenies satisfying certain characteristics.

References

[ACLLNSS23] Sarah Arpin, Catalina Camacho-Navarro, Kristin Lauter, Joelle Lim, Kristina Nelson, Travis Scholl, and Jana Sotáková. "Adventures in supersingularland". In: *Experimental Mathematics* vol. 32, no. 2 (2023), pp. 241–268. Taylor & Francis.

[AJL18] Reza Azarderakhsh, David Jao, and Christopher Leonardi. "Post-quantum static-static key agreement using multiple protocol instances". In: *Selected Areas in Cryptography — SAC 2017*. Ed. by Carlisle Adams and Jan Camenisch. Lecture Notes in Computer Science vol. 10719. Springer. 2018, pp. 45–63.

[BD99] Dan Boneh and Glenn Durfee. "Cryptanalysis of RSA with private key d less than $N^{0.292}$". In: *Advances in Cryptology — EUROCRYPT 1999*. Ed. by Jacques Stern. Lecture Notes in Computer Science vol. 1592. Springer. 1999, pp. 1–11.

[BDH99] Dan Boneh, Glenn Durfee, and Nick Howgrave-Graham. "Factoring $N = p^r q$ for large r". In: *Advances in Cryptology — CRYPTO'99*. Ed. by Michael Wiener. Lecture Notes in Computer Science vol. 1666. Springer. 1999, pp. 326–337.

[BF23] Andrea Basso and Tako Boris Fouotsa. "New SIDH Countermeasures for a More Efficient Key Exchange". In: *Cryptology ePrint Archive* (2023). ePrint no. 2023/791.

[BFT14] Nils Bruin, E Victor Flynn, and Damiano Testa. "Descent via $(3, 3)$-isogeny on Jacobians of genus 2 curves". In: *Acta Arithmetica* vol. 165, no. 3 (2014), pp. 201–223. Institute of Mathematics of the Polish Academy of Sciences.

[BJ07] Aurélie Bauer and Antoine Joux. "Toward a Rigorous Variation of Coppersmith's Algorithm on Three Variables". In: *Advances in Cryptology — EUROCRYPT 2007*. Ed. by Moni Naor. Lecture Notes in Computer Science vol. 4515. Springer, 2007, pp. 361–378.

[BJS14] Jean-François Biasse, David Jao, and Anirudh Sankar. "A quantum algorithm for computing isogenies between supersingular elliptic curves". In: *Progress in Cryptology — INDOCRYPT 2014*. Ed. by Willi Meier and Debdeep Mukhopadhyay. Lecture Notes in Computer Science vol. 8885. Springer. 2014, pp. 428–442.

[BKMPW20] Andrea Basso, Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Charlotte Weitkämper. "On adaptive attacks against Jao–Urbanik's isogeny-based protocol". In: *Progress in Cryptology — AFRICACRYPT 2020*. Ed. by Abderrahmane Nitaj and Amr M. Youssef. Lecture Notes in Computer Science vol. 12174. Springer. 2020, pp. 195–213.

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. "CSI-FiSh: efficient isogeny based signatures through class group computations". In: *Advances in Cryptology — ASI-*

ACRYPT 2019. Ed. by Steven D. Galbraith and Shiho Moriai. Lecture Notes in Computer Science vol. 11921. Springer. 2019, pp. 227–247.

[BLMP19] Daniel J Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. "Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies". In: *Advances in Cryptology — EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Lecture Notes in Computer Science vol. 11477. Springer. 2019, pp. 409–441.

[BMP23] Andrea Basso, Luciano Maino, and Giacomo Pope. "FESTA: Fast Encryption from Supersingular Torsion Attacks". In: *Cryptology ePrint Archive* (2023). ePrint no. 2023/660.

[BS20] Xavier Bonnetain and André Schrottenloher. "Quantum security analysis of CSIDH". In: *Advances in Cryptology — EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Lecture Notes in Computer Science vol. 12106. Springer. 2020, pp. 493–522.

[BVZ12] Aurélie Bauer, Damien Vergnaud, and Jean-Christophe Zapalowicz. "Inferring Sequences Produced by Nonlinear Pseudorandom Number Generators Using Coppersmith's Methods". In: *Public Key Cryptography — PKC 2012*. Ed. by Marc Fischlin, Johannes Buchmann, and Mark Manulis. Lecture Notes in Computer Science vol. 7293. Springer, 2012, pp. 609–626.

[CCS22] Maria Corte-Real Santos, Craig Costello, and Jia Shi. "Accelerating the Delfs–Galbraith Algorithm with Fast Subfield Root Detection". In: *Advances in Cryptology — CRYPTO*

2022. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Lecture Notes in Computer Science vol. 13509. Springer. 2022, pp. 285–314.

[CD23] Wouter Castryck and Thomas Decru. "An efficient key recovery attack on SIDH". In: *Advances in Cryptology — EUROCRYPT 2023*. Ed. by Carmit Hazay and Martijn Stam. Lecture Notes in Computer Science vol. 14008. Springer, 2023, pp. 423–447.

[CDS20] Wouter Castryck, Thomas Decru, and Benjamin Smith. "Hash functions from superspecial genus-2 curves using Richelot isogenies". In: *Journal of Mathematical Cryptology* vol. 14, no. 1 (2020), pp. 268–292. De Gruyter.

[CF96] J. W. S. Cassels and E. V. Flynn. *Prolegomena to a Middlebrow Arithmetic of Curves of Genus 2*. London Mathematical Society Lecture Note Series vol. 230. Cambridge University Press, 1996.

[CGL09] Denis X Charles, Eyal Z Goren, and Kristin E Lauter. "Families of Ramanujan graphs and quaternion algebras". In: *Groups and Symmetries: From Neolithic Scots to John McKay*. Ed. by J.P. Harnad and P. Winternitz. CRM proceedings & lecture notes vol. 47. American Mathematical Society, 2009, pp. 53–63.

[CIIKLP23] Mingjie Chen, Muhammad Imran, Gábor Ivanyos, Péter Kutas, Antonin Leroux, and Christophe Petit. "Hidden Stabilizers, the Isogeny to Endomorphism Ring Problem and the Cryptanalysis of pSIDH". In: *Cryptology ePrint Archive*

(2023). ePrint no. 2023/779 (to appear at ASIACRYPT 2023).

[CJS14] Andrew Childs, David Jao, and Vladimir Soukharev. "Constructing elliptic curve isogenies in quantum subexponential time". In: *Journal of Mathematical Cryptology* vol. 8, no. 1 (2014), pp. 1–29. De Gruyter.

[CK20] Leonardo Colò and David Kohel. "Orienting supersingular isogeny graphs". In: *Journal of Mathematical Cryptology* vol. 14, no. 1 (2020), pp. 414–437. De Gruyter.

[CLG09] Denis X Charles, Kristin E Lauter, and Eyal Z Goren. "Cryptographic hash functions from expander graphs". In: *Journal of Cryptology* vol. 22, no. 1 (2009), pp. 93–113. Springer.

[CLMPR18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. "CSIDH: An Efficient Post-Quantum Commutative Group Action". In: *Advances in Cryptology — ASIACRYPT 2018*. Ed. by Thomas Peyrin and Steven Galbraith. Lecture Notes in Computer Science vol. 11274. Springer. 2018, pp. 395–427.

[CLNRV20] C. Costello, P. Longa, M. Naehrig, J. Renes, and Fernando Virdia. "Improved Classical Cryptanalysis of SIKE in Practice". In: *Public Key Cryptography — PKC 2020*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Lecture Notes in Computer Science vol. 12111. Springer, 2020, pp. 505–534.

[Coh13] Henri Cohen. *A course in computational algebraic number theory*. Graduate Texts in Mathematics vol. 138. Springer, 2013.

[Cop96a] Don Coppersmith. "Finding a small root of a bivariate integer equation; factoring with high bits known". In: *Advances in Cryptology — EUROCRYPT 1996*. Ed. by Ueli Maurer. Lecture Notes in Computer Science vol. 1070. Springer. 1996, pp. 178–189.

[Cop96b] Don Coppersmith. "Finding a small root of a univariate modular equation". In: *Advances in Cryptology — EUROCRYPT 1996*. Ed. by Ueli Maurer. Lecture Notes in Computer Science vol. 1070. Springer. 1996, pp. 155–165.

[Cop97] Don Coppersmith. "Small solutions to polynomial equations, and low exponent RSA vulnerabilities". In: *Journal of Cryptology* vol. 10, no. 4 (1997), pp. 233–260. Springer.

[Cor07] Jean-Sébastien Coron. "Finding small roots of bivariate integer polynomial equations: A direct approach". In: *Advances in Cryptology — CRYPTO 2007*. Ed. by Alfred Menezes. Lecture Notes in Computer Science vol. 4622. Springer. 2007, pp. 379–394.

[Cor08] Giuseppe Cornacchia. "Su di un metodo per la risoluzione in numeri interi dell'equazione $\sum_{h=0}^{n} c_h x^{n-h} y^h = p$". In: *Giornale di Matematiche di Battaglini* vol. 46, (1908), pp. 33–90.

[Cou97] Jean-Marc Couveignes. "Hard homogeneous spaces". In: *Cryptology ePrint Archive* (1997). ePrint no. 2006/291.

[CR15] Romain Cosset and Damien Robert. "Computing (ℓ, ℓ)-isogenies in polynomial time on Jacobians of genus 2 curves". In: *Mathematics of Computation* vol. 84, no. 294 (2015), pp. 1953–1975. American Mathematical Society.

[CS20] Craig Costello and Benjamin Smith. "The supersingular isogeny problem in genus 2 and beyond". In: *Post-Quantum Cryptography — PQCrypto 2020*. Ed. by Jintai Ding and Jean-Pierre Tillich. Lecture Notes in Computer Science vol. 12100. Springer. 2020, pp. 151–168.

[DDFKLPSW21] Luca De Feo, Cyprien Delpech de Saint Guilhem, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Christophe Petit, Javier Silva, and Benjamin Wesolowski. "SÉTA: Supersingular encryption from torsion attacks". In: *Advances in Cryptology — ASIACRYPT 2021*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Lecture Notes in Computer Science vol. 13093. Springer. 2021, pp. 249–278.

[DDGZ23] Luca De Feo, Samuel Dobson, Steven D Galbraith, and Lukas Zobernig. "SIDH proof of knowledge". In: *Advances in Cryptology — ASIACRYPT 2022*. Ed. by Shweta Agrawal and Dongdai Lin. Lecture Notes in Computer Science vol. 13792. Springer. 2023, pp. 310–339.

[Den03] Alexander W Dent. "A designer's guide to KEMs". In: *Cryptography and Coding 2003*. Ed. by Kenneth G Paterson. Lecture Notes in Computer Science vol. 2898. Springer. 2003, pp. 133–151.

[Deu41] Max Deuring. "Die Typen der Multiplikatorenringe elliptischer Funktionenkörper: G. Herglotz zum 60. Geburtstag gewidmet". In: *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*. Vol. 14. Springer. 1941, pp. 197–272.

[DG16] Christina Delfs and Steven D Galbraith. "Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$". In: *Designs, Codes and Cryptography* vol. 78, no. 2 (2016), pp. 425–440. Springer.

[DG19] Luca De Feo and Steven D Galbraith. "SeaSign: compact isogeny signatures from class group actions". In: *Advances in Cryptology — EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Lecture Notes in Computer Science vol. 11478. Springer. 2019, pp. 759–789.

[DGLTZ20] Samuel Dobson, Steven D Galbraith, Jason LeGrow, Yan Bo Ti, and Lukas Zobernig. "An adaptive attack on 2-SIDH". In: *International Journal of Computer Mathematics: Computer Systems Theory* vol. 5, no. 4 (2020), pp. 282–299. Taylor & Francis.

[DH76] W. Diffie and M. Hellman. "New directions in cryptography". In: *IEEE Transactions on Information Theory* vol. 22, no. 6 (1976), pp. 644–654.

[DK23] Thomas Decru and Sabrina Kunzweiler. "Efficient computation of $(3^n, 3^n)$-isogenies". In: *Cryptology ePrint Archive* (2023). ePrint no. 2023/376 (to appear at AFRICACRYPT 2023).

[DKLPW20] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. "SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies". In: *Advances in Cryptology — ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Lecture Notes in Computer Science vol. 12491. Springer, 2020, pp. 64–93.

[DLLW23] Luca De Feo, Antonin Leroux, Patrick Longa, and Benjamin Wesolowski. "New Algorithms for the Deuring Correspondence". In: *Advances in Cryptology — EUROCRYPT 2023*. Ed. by Carmit Hazay and Martijn Stam. Lecture Notes in Computer Science vol. 14008. Springer. 2023, pp. 659–690.

[DLRW23] Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. "SQISignHD: New dimensions in cryptography". In: *Cryptology ePrint Archive* (2023). ePrint no. 2023/436.

[DM20] Luca De Feo and Michael Meyer. "Threshold schemes from isogeny assumptions". In: *Public Key Cryptography — PKC 2020*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Lecture Notes in Computer Science vol. 12111. Springer. 2020, pp. 187–212.

[DMPS19] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. "Verifiable delay functions from supersingular isogenies and pairings". In: *Advances in Cryptology — ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Lecture Notes in Computer Science vol. 11921. Springer. 2019, pp. 248–277.

[EHLMP18] Kirsten Eisenträger, Sean Hallgren, Kristin E. Lauter, Travis Morrison, and Christophe Petit. "Supersingular Isogeny Graphs and Endomorphism Rings: Reductions and Solutions". In: *Advances in Cryptology — EUROCRYPT 2018*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Lecture Notes in Computer Science vol. 10822. Springer. 2018, pp. 329–368.

[EHLMP20] Kirsten Eisenträger, Sean Hallgren, Chris Leonardi, Travis Morrison, and Jennifer Park. "Computing endomorphism rings of supersingular elliptic curves and connections to path-finding in isogeny graphs". In: *Open Book Series* vol. 4, no. 1 (2020), pp. 215–232. Mathematical Sciences Publishers.

[EPSV23] Jonathan Komada Eriksen, Lorenz Panny, Jana Sotáková, and Mattia Veroni. "Deuring for the People: Supersingular Elliptic Curves with Prescribed Endomorphism Ring in General Characteristic". In: *Cryptology ePrint Archive* (2023). ePrint no. 2023/106 (accepted at LuCaNT 2023).

[FKMT21] Tako Boris Fouotsa, Péter Kutas, Simon-Philipp Merz, and Yan Bo Ti. "On the Isogeny Problem with Torsion Point Information". In: *Cryptology ePrint Archive* (2021). ePrint no. 2021/153.

[FKMT22] Tako Boris Fouotsa, Péter Kutas, Simon-Philipp Merz, and Yan Bo Ti. "On the isogeny problem with torsion point information". In: *Public Key Cryptography — PKC 2022*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Lecture Notes in Computer Science vol. 13177. Springer. 2022, pp. 142–161.

[Fly15] E Victor Flynn. "Descent via $(5, 5)$-isogeny on Jacobians of genus 2 curves". In: *Journal of Number Theory* vol. 153, (2015), pp. 270–282. Elsevier.

[FMP23] Tako Boris Fouotsa, Tomoki Moriya, and Christophe Petit. "M-SIDH and MD-SIDH: countering SIDH attacks by masking information". In: *Advances in Cryptology — EU-*

ROCRYPT 2023. Ed. by Carmit Hazay and Martijn Stam. Lecture Notes in Computer Science vol. 14008. Springer, 2023, pp. 282–309.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure integration of asymmetric and symmetric encryption schemes". In: *Advances in Cryptology — CRYPTO'99*. Ed. by Michael Wiener. Lecture Notes in Computer Science vol. 1666. Springer. 1999, pp. 537–554.

[Fou22] Tako Boris Fouotsa. "SIDH with masked torsion point images". In: *Cryptology ePrint Archive* (2022). ePrint no. 2022/1054.

[FSMS09] Katalin Friedl, Miklos Santha, Frédéric Magniez, and Pranab Sen. "Quantum testers for hidden group properties". In: *Fundamenta Informaticae* vol. 91, no. 2 (2009), pp. 325–340. IOS Press.

[FT19] E Victor Flynn and Yan Bo Ti. "Genus two isogeny cryptography". In: *Post-Quantum Cryptography — PQCrypto 2019*. Ed. by Jintai Ding and Rainer Steinwandt. Lecture Notes in Computer Science vol. 11505. Springer. 2019, pp. 286–306.

[GPS20] Steven D Galbraith, Christophe Petit, and Javier Silva. "Identification protocols and signature schemes based on supersingular isogeny problems". In: *Journal of Cryptology* vol. 33, no. 1 (2020), pp. 130–175. Springer.

[GPST16] Steven D Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. "On the security of supersingular isogeny cryptosystems". In: *Advances in Cryptology — ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Lec-

ture Notes in Computer Science vol. 10031. Springer. 2016, pp. 63–91.

[Gro96] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *ACM Symposium on Theory of Computing — STOC 1996*. Ed. by Gary L. Miller. Association for Computing Machinery, 1996, pp. 212–219.

[GTTD07] Pierrick Gaudry, Emmanuel Thomé, Nicolas Thériault, and Claus Diem. "A double large prime variation for small genus hyperelliptic index calculus". In: *Mathematics of Computation* vol. 76, no. 257 (2007), pp. 475–492. American Mathematical Society.

[GTV90] Marc Girault, Philippe Toffin, and Brigitte Vallée. "Computation of approximate L-th roots modulo n and application to cryptography". In: *Advances in Cryptology — CRYPTO'88*. Ed. by Shafi Goldwasser. Lecture Notes in Computer Science vol. 403. Springer. 1990, pp. 100–117.

[Has86] Johan Hastad. "On using RSA with low exponent in a public key network". In: *Advances in Cryptology — CRYPTO'85 Proceedings*. Ed. by Hugh C. Williams. Lecture Notes in Computer Science vol. 218. Springer. 1986, pp. 403–408.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A modular analysis of the Fujisaki–Okamoto transformation". In: *Theory of Cryptography Conference — TCC 2017*. Ed. by Yael Kalai and Leonid Reyzin. Lecture Notes in Computer Science vol. 10677. Springer. 2017, pp. 341–371.

[How97] Nicholas Howgrave-Graham. "Finding small roots of univariate modular equations revisited". In: *Crytography and*

Coding 1997. Ed. by Michael Darnell. Lecture Notes in Computer Science vol. 1355. Springer. 1997, pp. 131–142.

[HW79] Godfrey Harold Hardy and Edward Maitland Wright. *An introduction to the theory of numbers*. Oxford University Press, 1979.

[Igu60] Jun-Ichi Igusa. "Arithmetic variety of moduli for genus two". In: *Annals of Mathematics* vol. 72, no. 3 (1960), pp. 612–649.

[JACCDHHJKKLLNPRSU22] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Aaron Hutchinson, Amir Jalali, Koray Karabina, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Geovandro Pereira, Joost Renes, Vladimir Soukharev, and David Urbanik. "Supersingular isogeny key encapsulation". In: *Update for Round 3 of NIST Post-Quantum Standardization Project* (2022). http://sike.org/.

[JACCDHJKLLNRSU17] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. "SIKE: Supersingular isogeny key encapsulation". In: *Submission to the NIST Post-Quantum Standardization project* (2017). http://sike.org/.

[JD11] David Jao and Luca De Feo. "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies". In: *Post-Quantum Cryptography — PQCrypto 2011*. Ed. by

Bo-Yin Yang. Lecture Notes in Computer Science vol. 7071. Springer. 2011, pp. 19–34.

[JS19] Samuel Jaques and John M Schanck. "Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE". In: *Advances in Cryptology — CRYPTO 2019*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Lecture Notes in Computer Science vol. 11692. Springer. 2019, pp. 32–61.

[Jut98] Charanjit S Jutla. "On finding small solutions of modular multivariate polynomial equations". In: *Advances in Cryptology — EUROCRYPT 1998*. Ed. by Kaisa Nyberg. Lecture Notes in Computer Science vol. 1403. Springer, 1998, pp. 158–170.

[Kan97] Ernst Kani. "The number of curves of genus two with elliptic differentials". In: *Journal für die reine und angewandte Mathematik* vol. 485, (1997), pp. 93–122. De Gruyter.

[KLPT14] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. "On the quaternion ℓ-isogeny path problem". In: *LMS Journal of Computation and Mathematics* vol. 17, no. A (2014), pp. 418–432. London Mathematical Society.

[KMPW21] Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Charlotte Weitkämper. "One-way functions and malleability oracles: Hidden shift attacks on isogeny-based protocols". In: *Advances in Cryptology — EUROCRYPT 2021*. Ed. by Anne Canteaut and François-Xavier Standaert. Lecture Notes in Computer Science vol. 12696. Springer. 2021, pp. 242–271.

[Kob89] Neal Koblitz. "Hyperelliptic cryptosystems". In: *Journal of Cryptology* vol. 1, no. 3 (1989), pp. 139–150. Springer.

[Koh96] David Russell Kohel. "Endomorphism rings of elliptic curves over finite fields". PhD thesis. University of California, Berkeley, 1996. https://www.i2m.univ-amu.fr/perso/david.kohel/pub/thesis.pdf.

[KTW21] Sabrina Kunzweiler, Yan Bo Ti, and Charlotte Weitkämper. "Secret keys in Genus-2 SIDH". In: *Selected Areas in Cryptography — SAC 2021*. Ed. by Riham AlTawy and Andreas Hülsing. Lecture Notes in Computer Science vol. 13203. Springer. 2021, pp. 483–507.

[Kun22] Sabrina Kunzweiler. "Efficient Computation of $(2^n, 2^n)$-Isogenies". In: *Cryptology ePrint Archive* (2022). ePrint no. 2022/990.

[Kup05] Greg Kuperberg. "A subexponential-time quantum algorithm for the dihedral hidden subgroup problem". In: *SIAM Journal on Computing* vol. 35, no. 1 (2005), pp. 170–188. SIAM.

[Kup13] Greg Kuperberg. "Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem". In: *Theory of Quantum Computation, Communication and Cryptography — TQC 2013*. Ed. by Simone Severini and Fernando Brandao. vol. 22. Leibniz-Zentrum für Informatik, 2013, pp. 20–34.

[KV10] Markus Kirschmer and John Voight. "Algorithmic enumeration of ideal classes for quaternion orders". In: *SIAM Jour-*

nal on Computing vol. 39, no. 5 (2010), pp. 1714–1747. SIAM.

[Lan08] Edmund Landau. "Über die Einteilung der positiven ganzen Zahlen in vier Klassen nach der Mindestzahl der zu ihrer additiven Zusammensetzung erforderlichen Quadrate". In: *Archiv der Mathematik und Physik* vol. 13, no. 3 (1908), pp. 305–312. Teubner.

[Ler23] Antonin Leroux. "A new isogeny representation and applications to cryptography". In: *Advances in Cryptology — ASIACRYPT 2022*. Ed. by Shweta Agrawal and Dongdai Lin. Lecture Notes in Computer Science vol. 13792. Springer. 2023, pp. 3–35.

[LLL82] Arjen K. Lenstra, Hendrik Willem Lenstra, and László Lovász. "Factoring polynomials with rational coefficients". In: *Mathematische Annalen* vol. 261, (1982), pp. 515–534. Springer.

[MAGMA] Wieb Bosma, John Cannon, and Catherine Playoust. "The Magma algebra system I. The user language". In: *Journal of Symbolic Computation* vol. 24, no. 3–4 (1997), pp. 235–265.

[Mil86] J. S. Milne. "Abelian Varieties". In: *Arithmetic Geometry*. Ed. by Gary Cornell and Joseph H. Silverman. Springer, 1986, pp. 103–150.

[MM22] Luciano Maino and Chloe Martindale. "An attack on SIDH with arbitrary starting curve". In: *Cryptology ePrint Archive* (2022). ePrint no. 2022/1026.

[MMPPW23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. "A direct key recovery

attack on SIDH". In: *Advances in Cryptology — EUROCRYPT 2023*. Ed. by Carmit Hazay and Martijn Stam. Lecture Notes in Computer Science vol. 14008. Springer, 2023, pp. 448–471.

[Mor22] Tomoki Moriya. "Masked-degree SIDH". In: *Cryptology ePrint Archive* (2022). ePrint no. 2022/1019.

[Mos18] Michele Mosca. "Cybersecurity in an era with quantum computers: will we be ready?" In: *IEEE Security & Privacy* vol. 16, no. 5 (2018), pp. 38–41. IEEE.

[MP22] Michele Mosca and Marco Piani. *2022 Quantum threat timeline report*. https://globalriskinstitute.org/publication/2022-quantum-threat-timeline-report/. 2022.

[Mum70] David Mumford. *Abelian Varieties*. Oxford University Press, 1970.

[NIST16] National Institute for Standards and Technology (NIST). *NIST Post-Quantum Cryptography Project*. http://csrc.nist.gov/groups/ST/post-quantum-crypto/. 2016.

[NIST23] National Institute for Standards and Technology (NIST). *NIST Post-Quantum Cryptography Project: Digital Signature Schemes*. https://csrc.nist.gov/projects/pqc-dig-sig. 2023.

[Nit95] Abderrahmane Nitaj. "L'algorithme de Cornacchia". In: *Exposition. Math.* vol. 13, no. 4 (1995), pp. 358–365.

[PARI/GP] The PARI Group. *PARI/GP*. https://pari.math.u-bordeaux.fr. Université de Bordeaux, 2000–2022.

[Pei20] Chris Peikert. "He gives C-sieves on the CSIDH". In: *Advances in Cryptology — EUROCRYPT 2020*. Ed. by Anne

Canteaut and Yuval Ishai. Lecture Notes in Computer Science vol. 12106. Springer. 2020, pp. 463–492.

[Pet17] Christophe Petit. "Faster algorithms for isogeny problems using torsion point images". In: *Advances in Cryptology — ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Lecture Notes in Computer Science vol. 10625. Springer. 2017, pp. 330–353.

[PL17] Christophe Petit and Kristin Lauter. "Hard and easy problems for supersingular isogeny graphs". In: *Cryptology ePrint Archive* (2017). ePrint no. 2017/962.

[PS18] Christophe Petit and Spike Smith. *An improvement to the quaternion analogue of the l-isogeny problem*. Presentation at MathCrypt. 2018. https://crypto.iacr.org/2018/affevents/mathcrypt/medias/08-50_3.pdf.

[QKLMPPS21] Victoria de Quehen, Péter Kutas, Chris Leonardi, Chloe Martindale, Lorenz Panny, Christophe Petit, and Katherine E Stange. "Improved torsion-point attacks on SIDH variants". In: *Advances in Cryptology — CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Lecture Notes in Computer Science vol. 12827. Springer. 2021, pp. 432–470.

[Ram13] Srinivasa Ramanujan. "First letter to G.H. Hardy". In: *Ramanujan: Letters and Commentary*. Ed. by Bruce C. Berndt and Robert A. Rankin. History of Mathematics vol. 9. American Mathematical Society, 1913, pp. 21–30.

[Reg04] Oded Regev. "A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space". In: *arXiv preprint quant-ph/0406151* (2004).

[Rob22] Damien Robert. "Evaluating isogenies in polylogarithmic time". In: *Cryptology ePrint Archive* (2022). ePrint no. 2022/1068.

[Rob23] Damien Robert. "Breaking SIDH in polynomial time". In: *Advances in Cryptology — EUROCRYPT 2023*. Ed. by Carmit Hazay and Martijn Stam. Lecture Notes in Computer Science vol. 14008. Springer, 2023, pp. 472–503.

[ROW94] Hans Riesel, J Oesterlé, and A Weinstein. *Prime numbers and computer methods for factorization*. Progress in Mathematics vol. 126. Springer, 1994.

[RS06] Alexander Rostovtsev and Anton Stolbunov. "Public-key cryptosystem based on isogenies". In: *Cryptology ePrint Archive* (2006). ePrint no. 2006/145.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the ACM* vol. 21, no. 2 (1978), pp. 120–126.

[SAGE] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.0)*. https://www.sagemath.org. 2023.

[Sch87] René Schoof. "Nonsingular plane cubic curves over finite fields". In: *Journal of Combinatorial Theory, Series A* vol. 46, no. 2 (1987), pp. 183–211. Elsevier.

[Sho94] Peter W Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th annual symposium on foundations of computer science*. IEEE. 1994, pp. 124–134.

[Sho97] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* vol. 26, no. 5 (1997), pp. 1484–1509. SIAM.

[Sil09] Joseph H Silverman. *The arithmetic of elliptic curves*. Graduate Texts in Mathematics vol. 106. Springer, 2009.

[Sim05] Denis Simon. *Quadratic equations in dimensions 4, 5 and more*. Preprint. https://simond.users.lmno.cnrs.fr/maths/Dim4.pdf. 2005.

[SSW08] Reginald E Sawilla, Alan K Silvester, and Hugh C Williams. "A new look at an old equation". In: *Algorithmic Number Theory — ANTS-VIII*. Ed. by Alfred J. van der Poorten and Andreas Stein. Lecture Notes in Computer Science vol. 5011. Springer. 2008, pp. 37–59.

[Tak18] Katsuyuki Takashima. "Efficient algorithms for isogeny sequences and their cryptographic applications". In: *Mathematical Modelling for Next-Generation Cryptography*. Ed. by Tsuyoshi Takagi, Masato Wakayama, Keisuke Tanaka, Noboru Kunihiro, Kazufumi Kimoto, and Dung Hoang Duong. Mathematics for Industry 29. Springer, 2018, pp. 97–114.

[Tan09] Seiichiro Tani. "Claw finding algorithms using quantum walk". In: *Theoretical Computer Science* vol. 410, no. 50 (2009), pp. 5285–5297. Elsevier.

[Tat66] John Tate. "Endomorphisms of abelian varieties over finite fields". In: *Inventiones mathematicae* vol. 2, no. 2 (1966), pp. 134–144. Springer.

[Ti19] Yan Bo Ti. "Isogenies of abelian varieties in cryptography (Ph.D. thesis)". PhD thesis. University of Auckland, 2019. https://www.math.auckland.ac.nz/~sgal018/Yan-Bo-Ti-Thesis.pdf.

[UJ20] David Urbanik and David Jao. "New techniques for SIDH-based NIKE". In: *Journal of Mathematical Cryptology* vol. 14, no. 1 (2020), pp. 120–128. De Gruyter.

[Vél71] Jacques Vélu. "Isogénies entre courbes elliptiques". In: *Comptes-Rendus de l'Académie des Sciences* vol. 273, (1971), pp. 238–241.

[Voi18] John Voight. *Quaternion algebras*. Preprint. https://math.dartmouth.edu/~jvoight/quat-book.pdf, 2018.

[Wat69] William C Waterhouse. "Abelian varieties over finite fields". In: *Annales scientifiques de l'École Normale Supérieure* vol. 2, no. 4 (1969), pp. 521–560.

[Wei57] André Weil. "Zum Beweis des Torellischen Satzes". In: *Nachr. Akad. Wiss. Göttingen, Math.-Phys. Kl.* (1957), pp. 33–53. Vandenhoeck & Ruprecht.

[Wes22a] Benjamin Wesolowski. "Orientations and the supersingular endomorphism ring problem". In: *Advances in Cryptology — EUROCRYPT 2022*. Ed. by Orr Dunkelman and Stefan Dziembowski. Lecture Notes in Computer Science vol. 13277. Springer. 2022, pp. 345–371.

[Wes22b] Benjamin Wesolowski. "The supersingular isogeny path and endomorphism ring problems are equivalent". In: *62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 1100–1111.