isogenies & isometries

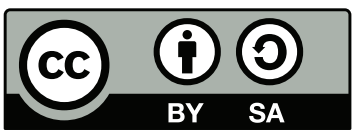
Krijn Reijnders

Isogenies & Isometries

Krijn Reijnders

https://orcid.org/0009-0002-8015-399X

Digital version available through thesis.post-quantum-crypto.com.



Isogenies & Isometries

Proefschrift ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. dr. J. M. Sanders,
volgens besluit van het college voor promoties
in het openbaar te verdedigen op

TODO: dag datum 2025
om **TODO:** tijd uur precies

door

Krijn Cornelius Johannes Maria Reijnders

geboren op 12 januari 1995
te Milheeze

Supervisor

Dr. Simona Samardjiska

Promotor

Prof. dr. Lejla Batina

Manuscriptcommissie

Prof. dr. TBD
TODO: FIX

Prof. dr. TBD

Prof. dr. TBD
TODO: FIX

Prof. dr. TBD

Prof. dr. TBD
TODO: FIX

THANKS

Hier komt een dankjewel naar iedereen! For now, this is stolen from Thom for type-setting purposes.

The book that lies before you is the culmination of a few years of work, and the end of a long time at Radboud University. I have started out as a first-year Bachelor's student, and am now leaving with a finished Ph.D. thesis. I look back on my time very fondly, which in no small part is thanks to the many people I have had the pleasure of working and interacting with throughout the years I spent at Radboud University.

Krijn Reijnders
Nijmegen, Augustus 2024

CONTENTS

INTRODUCTION TO THE THESIS

A thesis on cryptography usually starts with an introduction on cryptography. To not prolongate this thesis more than necessary, I will keep this part brief and skip the usual discussion about Romans and their ciphers.

As the world becomes more and more digital, we rely on cryptography to keep our lives secure and private. The average person will notice this mostly in the lock symbol that is shown in a web browser or the 'end-to-end encrypted' notification shown in messaging apps. And if cryptographers do their job correctly, the same person should not notice that cryptography is also used whenever they pay, make a phone call, or log in to a website.

Cryptographers hope to keep it that way, with the average user unaware of the cryptography underneath these services. This means that the cryptography we use 'in the real world' should be secure and efficient, for years or even decades to come. For example, some planes currently used in commercial flights are over 40 years old. If we want planes built today to last a similar time, we should also make sure they are digitally security for such a long lifespan. As another example, the most secret documents in the Netherlands must stay secure for 50 or even 75 years, and we must therefore ensure their encryption remains unbroken for the same amount of time.

This becomes problematic with the current development in quantum computers. These machines base their computational units on quantum bits (qubits) instead of regular bits, based on the fundamental quantum-mechanical nature of the universe. Their computational power is different to classical computers: quantum computers can solve very specific problems exponentially faster than classical computers, although they do not seem to provide a speed-up for most other problems. Unfortunately, we have most of our public-key cryptography precisely on those problems that quantum computers solve exponentially faster: the *integer factorization problem* and the *discrete logarithm problem*. Therefore, if we are able to build large enough quantum computers, we can break most public-key cryptography that is currently deployed to keep our digital worlds secure [351].

Building small quantum computers is already a daunting task, so that building large enough quantum computers to break public-key cryptography that is currently used is extremely challenging. We do not know if such quantum

computers will exist in ten years, twenty years, fifty years or perhaps never. Nevertheless, the threat exists and must be addressed already today if we want to securely build products or obscure messages that have a long lifespan, as the above examples show.

We therefore study *post-quantum cryptography:* cryptography that remains secure even if large quantum computers become reality. Current approaches usually rely on one of five mathematical or cryptographical structures: lattices, codes, isogenies, hash functions and multivariate systems. Crucially, post-quantum cryptography should still run on classical, i.e. non-quantum, devices, such as a phone or laptop. Such cryptography has been studied since the 2000s, although as a relatively minor topic until the 2010s. The last decade has seen tremendous progress in this field, and we currently believe that we have a solid understanding of quantum algorithms and cryptographical problems, so that we can start using post-quantum cryptography in the real world, albeit combined with pre-quantum cryptography. In particular, we are starting to deploy lattice-based cryptography, a specific type of post-quantum cryptography, to encapsulate keys and to digitally sign messages. Lattice-based cryptography is incredibly fast, but less versatile and larger in size than pre-quantum cryptography.

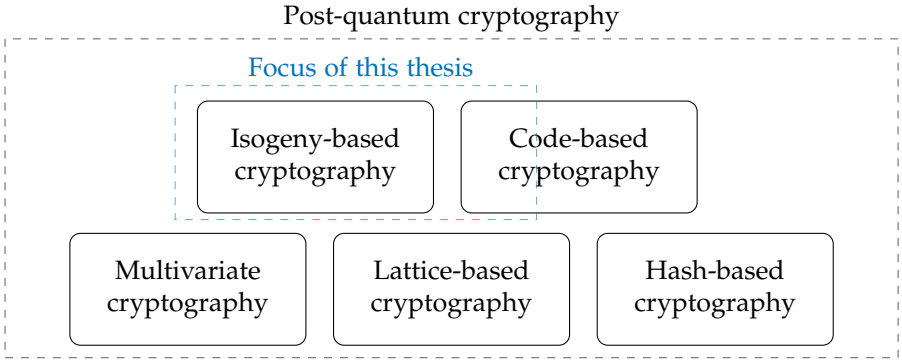


Figure 1: An overview of major topics in post-quantum cryptography.

In this thesis, we focus on isogeny-based cryptography and code-based cryptography using *isometries*. The hope is that they offer more flexability, smaller keys and signatures, and just a generally different hardness assumption than lattice-based cryptography. However, both types are also much less explored by the cryptographic community in terms of cryptanalysis and cryptographic

design. Thus, in this thesis, we analyse and improve the security and efficiency of schemes in isogeny-based and isometry-based cryptography.

There is some overlap in theory between *isogenies* and *isometries*: both are maps that preserve the essential structure between their domains and codomains, and both such maps should be hard to derive given only their domains and codomains. This makes both types of functions suitable for cryptography. Beyond these similarities, there is little overlap in the concrete theory and cryptographical application and we must be careful not to get carried away in their semblance. Nevertheless, we will see throughout this thesis that ideas in one field may inspire the other, and vice versa.

Structure of the Thesis

This thesis is divided into four parts, as visualised in Figure 2.

Part 0 lays the groundwork for the other three parts. This part is meant to introduce the reader to the most common elements of public-key cryptography, curves and their isogenies, and (linear) codes and their isometries. Experts may safely skip these sections. Similarly, advanced readers can pick and choose what parts to read, or refer back to this part when necessary while reading the other parts. If this paragraph or the following paragraphs are unreadable, we advise you to read this part.

Part 1 deals with curves and their isogenies, and focuses on CSIDH, a cryptographic group action. As CSIDH is relatively mature, this part focuses on *physical security*, both passive (Chapter IV) and active (Chapter V), and *optimizations and constant-time behaviour*, by developing and analyzing mathematical tools (Chapter VI and Chapter VIII) and studying real-world practicality (Chapter VII). The contents per subpart are presented chronologically, and each subpart can be read independently.

Part 2 also deals with curves and their isogenies, but focuses on SQIsign instead of CSIDH, a signature scheme. As SQIsign is relatively new, this part focuses mostly on mathematical tools (Chapter IX) and optimizations (Chapter X). We also explore a more esoteric approach to isogeny-based cryptography, using Kummer surfaces (Chapter XI).

Part 3 deals with codes and their isometries. Broadly, it covers the path from theoretical cryptanalysis (Chapter XII) to concrete cryptodesign (Chapter XIII) to

general techniques to improve such schemes (Chapter XIV). Thus, the chapters progress both through cryptographic maturity and chronologic development. As such, each chapter reflects our understanding at the time of writing.

The three main parts each start with an overview of the state of the art at the start of this thesis, so that the reader is able to place each chapter into the right context. At the end of each part, we briefly discuss the achieved results and some considerations for the future.

Usage of 'We'

A reader unfamiliar with mathematical or cryptographical writings might not expect most text to have been written in the first-person plural. This is not a case of majestic plural or any other grandeur. This usage of 'we' can be considered from three perspectives. First and foremost, 'we' refers to the combination of you (the reader) and I (the writer). While you read this thesis, I hope to guide you through it. Secondly, 'we' when expressing an opinion may in certain chapters reflect the opinion of my co-authors and me. None of the work in this thesis could have been done without my great co-authors. Thirdly, 'we' may sometimes reflect (a subset of) the cryptographic community. A few lines are written in the 'I' perspective. Such lines are my own personal opinion.

Contributions

This thesis is mostly a concatenation of published (or submitted) papers. Some introductory material has moved to Part o to prevent unnecessary repetitions and to increase the clarity of the thesis as a whole. Chapters IV to XIV each correspond to a paper. I briefly summarize their contents and highlight my personal contributions.

In Chapter IV, we describe a new passive side-channel attack on both CSIDH and SIKE, based on zero-value and correlation attacks. We demonstrate the feasability of these attacks on state-of-the-art implementations using simulations with various realistic noise levels. Additionally, we discuss countermeasures. This chapter is based on

> Fabio Campos, Michael Meyer, Krijn Reijnders, and Marc Stöttinger. "Patient Zero & Patient Six: Zero-Value and Correlation Attacks on

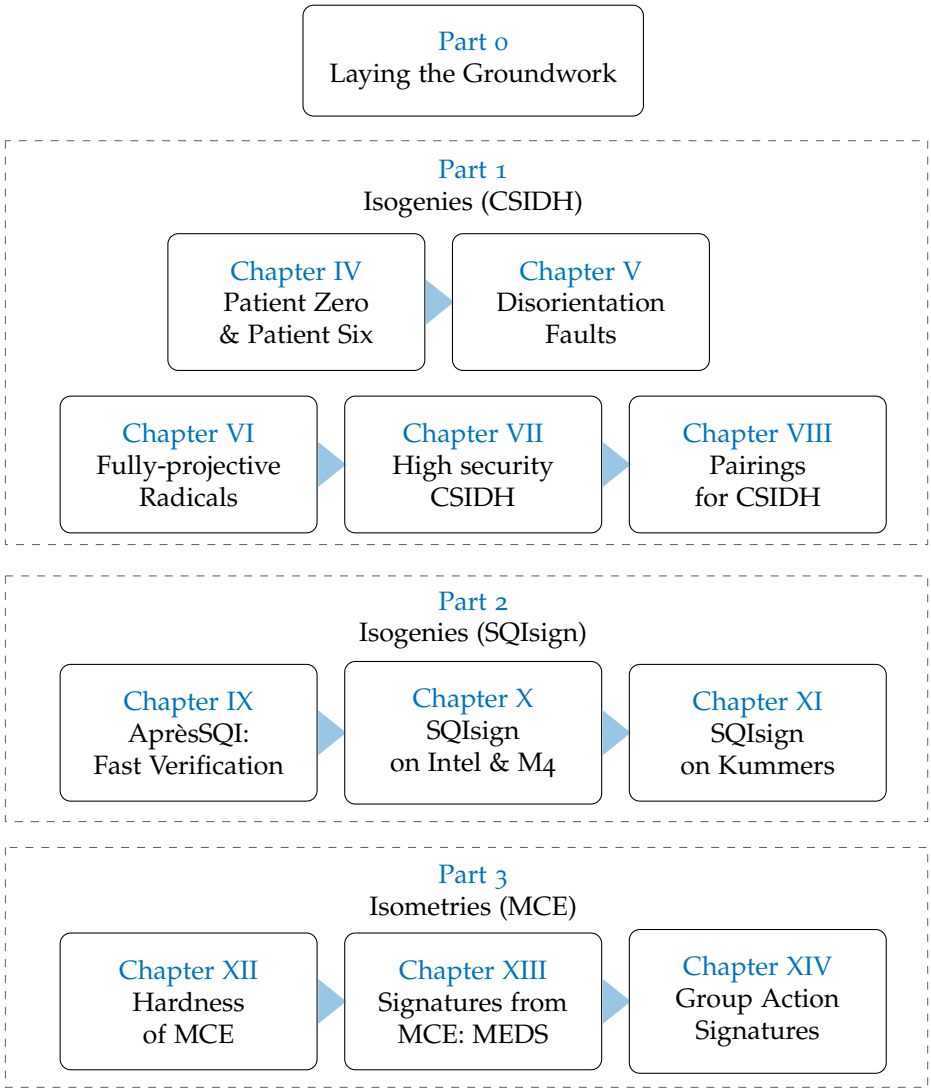


Figure 2: Organizational chart of the components of the thesis.

CSIDH and SIKE". in: *International Conference on Selected Areas in Cryptography*. Springer. 2022, pp. 234–262.

Contribution. The design of all attacks was a joint effort of all authors, as the ideas in this work were developed together in front of a blackboard. I developed the C code for the simulation of the SIKE attack. The writing was a collaborative effort between all authors.

In Chapter V, we investigate a new class of fault-injection attacks against the CSIDH family of cryptographic group actions. With several theoretical derivations, this reveals the secret key in CSIDH and CTIDH given only a relatively small number of samples. We provide a simulation and discuss lightweight countermeasures for the attack. This chapter is based on

Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska. "Disorientation faults in CSIDH". in: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 310–342.

Contribution. This work was the result of joint brainstorm sessions between the authors, where I was a major contributor to the ideas in Sections V.4, V.5, V.7, and V.8. I was also a major contributor to the writing of this paper.

In Chapter VI, we study the effectiveness of radical isogenies in constant-time implementations of CSIDH. We introduce a technique to compute projective roots in finite fields that allows us to compute projective radical isogenies almost twice as fast. Furthermore, we present a hybrid technique to optimally combine radical and traditional isogenies. We benchmark these techniques against state-of-the-art constant-time CSIDH implementations. This chapter is based on

Jesús-Javier Chi-Domínguez and Krijn Reijnders. "Fully projective radical isogenies in constant-time". In: *Cryptographers Track at the RSA Conference*. Springer. 2022, pp. 73–95.

Contribution. The work on projective radical isogenies was a joint effort between both authors. The work on the hybrid strategy was my contribution. The code and writing were a joint effort between both authors.

In Chapter VII, we optimise CSIDH in large parameter sets to analyse its practicality in real-world scenarios. To achieve this, we develop a deterministic

and dummy-free implementation which is more than twice as fast as previous results. We also consider CSIDH in network protocols, which shows too large latency to be practical. This chapter is based on

> Fabio Campos, Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Michael Meyer, Krijn Reijnders, Francisco Rodríguez-Henríquez Francisco, Peter Schwabe, and Thom Wiggers. "Optimizations and Practicality of High-Security CSIDH". in: *IACR Communications in Cryptology* 1.1 (2024).

Contribution. I am a main contributor to the design of the deterministic, dummy-free implementations in this work. Furthermore, I am the sole contributor of the VeriFast algorithm in this work. The writing was a joint effort between all authors. The attention to editorial detail in this paper is attributed to an uncredited author, who wishes to remain anonymous.

In Chapter VIII, we study the effectiveness of pairings in isogeny-based cryptography, with CSIDH as a main use case. We optimise the performance of pairings for this use case and develop several new algorithms based on pairings that are of general use in isogeny-based cryptography. We present a Rust implementation of the optimized pairings and the algorithms presented in this work. This chapter is based on

> Krijn Reijnders. "Effective Pairings in Isogeny-based Cryptography". In: *International Conference on Cryptology and Information Security in Latin America*. Springer. 2023, pp. 109–128.

Contribution. I am the sole author of the work and code in this paper.

In Chapter IX, we explore improvements to verification in SQIsign. By using extension-field signing, we are able to increase the available torsion required for faster verification. We analyse the impact of increased torsion on verification speed and provide several significant improvements for the verification algorithm. We furthermore explore size-speed trade-offs with significant practical relevance. Lastly, we benchmark all of the above in a Python implementation. This chapter is based on

> Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Krijn Reijnders. "AprèsSQI: extra fast verification for SQIsign using extension-field signing". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2024, pp. 63–93.

Contribution. I was a major contributor to the detailed analysis of verification in this work, and a main contributor of several of the core improvements. In particular, I contributed Theorem IX.4 and subsequent ideas to improve sampling of specific torsion points. I was also the main driving force to explore the performance of uncompressed signatures. Lastly, most of the writing of the code and the paper was a joint effort between all authors, except for the last days before the final submission, for which I eternally thank my co-authors.

In Chapter X, we further optimise the performance of one-dimensional SQIsign verification and provide the first optimized C implementation of the ideas presented in the previous chapter. We furthermore explore parallelisation and develop a highly-optimized library for both 32-bit and 64-bit architectures. This provides a new point of view on the relevance and significance of research into the signing procedure of one-dimensional SQIsign. This chapter is based on

> Marius A. Aardal, Gora Adj, Arwa Alblooshi, Diego F. Aranha, Isaac Canales-Martínez, Jorge Chávez-Saab, Décio Luiz Gazzoni Filho, Krijn Reijnders, and Francisco Rodríguez-Henríquez. "Optimized One-Dimensional SQIsign Verification on Intel and Cortex-M4". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025.1 (2025), pp. 1–25.

Contribution. I originated several of the core mathematical ideas in this paper, including the new public-key representation, the improved sampling techniques, the idea of shorter isogenies, and the improved hashing technique. Furthermore, I was a driving force behind exploring the parallelized SQIsign verification. I was a major contributor to the writing effort in this paper.

In Chapter XI, we explore the esoteric idea of performing SQIsign verification on Kummer surfaces, instead of elliptic curves. To do so, we must first expand the general toolbox of genus-2 cryptography. We present a detailed overview of the landscape of Kummer surfaces, develop a new technique for sampling torsion points using profiles of Tate pairings, and provide a crucial understanding of the theory connecting theta-model-based isogenies with Richelot isogenies. All of the above is implemented in both Magma and Python which enables us to perform SQIsign verification on Kummer surfaces. This chapter is based on

> Maria Corte-Real Santos and Krijn Reijnders. *Return of the Kummer: a Toolbox for Genus-2 Cryptography*. Cryptology ePrint Archive, Paper 2024/948. 2024. URL: https://eprint.iacr.org/2024/948.

Contribution. The exploration of the landscape of Kummer surfaces was a joint effort by both authors. I was the main contributor to the theory of profiles of Tate pairings and their applications, the discovery of improved maps from the Kummer to the Jacobian, and the computation of addition matrices for Kummer surfaces. The development of algorithms for Kummer-based cryptography was a joint effort. The development of the theory on Kummer isogenies was mostly done by the first author. Furthermore, I contributed a majority of the Magma code, whereas the first author contributed the majority of the Python code. Both the finetuning of the code as well as the writing of the paper was a joint effort between both authors.

In Chapter XII, we analyse the hardness of the matrix code equivalence problem. We show how this problem reduces to other cryptographical problems, and vice versa. This allows us to apply graph-based algorithms to the matrix code equivalence problem. This chapter is based on

> Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. "Hardness estimates of the code equivalence problem in the rank metric". In: *Designs, Codes and Cryptography* 92.3 (2024), pp. 833–862.

Contribution. This paper is a collaborative effort by all authors, as the ideas in this work were mostly developed in front of a whiteboard with all three authors present.

In Chapter XIII, we design a signature scheme based on the matrix code eqivalence problem, MEDS. We extend the cryptanalysis of the previous chapter and use this to provide an initial set of parameters for MEDS, along with a reference implementation in C. This chapter is based on

> Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. "Take your MEDS: digital signatures from matrix code equivalence". In: *International conference on cryptology in Africa*. Springer. 2023, pp. 28–52.

Contribution. I was a major contributor to the design of MEDS and the writing of the paper. Furthermore, I contributed to the cryptanalysis of MEDS.

In Chapter XIV, we present a unified taconomy of well-known and novel techniques to improve the performance of signature schemes based on Sigma protocols. This provides a concrete overview of the impact of these techniques, and we apply these to several group action-based schemes, such as MEDS.

Giacomo Borin, Edoardo Persichetti, Federico Pintore, Krijn Reijnders, and Paolo Santini. "A Guide to the Design of Digital Signatures based on Cryptographic Group Actions". In: *Journal of Cryptology* **TODO:** fix.**TODO:** fix (2024), **TODO:** fix.

Contribution. This work is a reformulation of an earlier work by the first, second, and fifth author. I contributed to an overall rewrite of that earlier work to unify the transformations and contributed to the analysis of combinations of techniques and the analysis of existing schemes.

Research Data

This thesis research has been carried out under the research data management policy of the Institute for Computing and Information Science of Radboud University, The Netherlands. As such, all data or software used in this thesis, arranged per chapter, can be found at

Krijn Reijnders. *Isogenies & Isometries*. 2024. DOI: 10.5281/zenodo.13293882. URL: https://doi.org/10.5281/zenodo.13293882.

Part 0

LAYING THE GROUNDWORK

LAYING THE GROUNDWORK

We start with an outline of public key cryptography (Chapter I). We continue with the theoretical background on curves and their isogenies (Chapter II) and then present codes and their isometries (Chapter III)

NOTATION. Random sampling from a set S is denoted by $a \xleftarrow{\$} S$. We use the notation $\tilde{\mathcal{O}}(f(n))$ to denote $\mathcal{O}(f(n)\log(f(n)))$ whenever we want to omit polynomial factors from the complexity expression. We use the notation $f = \Theta(g)$ whenever f is bounded from below and from above by g asymptotically. For a computational problem P, if we want to emphasize a list of parameters p defining the size of the inputs and the input set S, we use the notation $\mathsf{P}(p, S)$.

The finite field with p elements, with p prime, is denoted $\mathbb{F}_p$. The finite field with q elements, with $q = p^k$ for $k > 0$, is denoted $\mathbb{F}_q$. Whenever $p = 3 \mod 4$, the field $\mathbb{F}_{p^2}$ is realized as $\mathbb{F}_p(i)$ with $i^2 = -1$. The algebraic closure $\mathbb{F}_q$ is denoted $\bar{\mathbb{F}}_q$, and its polynomial ring $\mathbb{F}_q[x]$. The space of matrices, i.e. 2-tensors, over $\mathbb{F}_q$ of size $m \times n$ is denoted $\mathbb{F}_q^{m\times n}$. The set of k-subsets of $\mathbb{F}_q^{m\times n}$, i.e. 3-tensors, is denoted by $\mathbb{F}_q^{m\times n\times k}$. The general linear group and the general affine group of degree n over $\mathbb{F}_q$ are denoted by $\mathrm{GL}_n(q)$ and $\mathrm{AGL}_n(q)$, respectively.

For the chapters in code-based cryptography, we use bold letters to denote vectors $\mathbf{a}, \mathbf{c}, \mathbf{x}, \ldots$, and matrices $\mathbf{A}, \mathbf{B}, \ldots$. The identity matrix in $\mathrm{GL}_n(q)$ is denoted $\mathbf{I}_n$. The entries of a vector $\mathbf{a}$ are denoted by a_i, and we write $\mathbf{a} = (a_1, \ldots, a_n)$ for a (row) vector of dimension n over some field and $\mathbf{a}^\top = (a_1, \ldots, a_n)^\top$ for its transpose. Similarly, the entries of a matrix $\mathbf{A}$ are denoted by A_{ij}. A matrix $\mathbf{A}$ is called symmetric if $\mathbf{A}^\top = \mathbf{A}$ and skew-symmetric if $\mathbf{A}^\top = -\mathbf{A}$. The notation $|\mathbf{M}|$ denotes the determinant of the matrix $\mathbf{M}$. For two matrices $\mathbf{A}$ and $\mathbf{B}$, we denote the Kronecker product by $\mathbf{A} \otimes \mathbf{B}$.

In chapters on isogeny-based cryptography, the boldface letters $\mathbf{M}$, $\mathbf{S}$, $\mathbf{a}$, $\mathbf{I}$, and $\mathbf{E}$ should not be confused with matrices. Instead they denote the finite-field operations multiplication, squaring, addition, inverstion, and exponentiation, respectively. Orders are usually denoted by $\mathcal{O}$, and their class groups by $\mathcal{Cl}(\mathcal{O})$. The group of primitive r-th roots is denoted μ_r. Whenever $r \mid q - 1$, we may write $\mu_r \subseteq \mathbb{F}_q$ and assume *some* identification. The identification is made explicit when required.

I

GENERAL CRYPTOGRAPHY

§1 Public Key Cryptography

We start by outlining the general concepts of public key cryptography that will be used throughout the work such as cryptographic primitives and security notions. A classic reference is Katz and Lindell [235], a more thorough introduction for public-key cryptogaphy is the textbook by Galbraith [188], which inspired this section. Although cryptography has many spectacular and 'fancy' applications, we restrict ourselves to three basic constructions in public key cryptography:

1. *non-interactive key exchange protocols* (NIKEs) which exchange a key between two parties without any interaction between these parties,

2. *key-encapsulation mechanisms* (KEMs), which simply allow two parties to agree on a shared value (a key), but require interaction, and

3. *digital signatures*, which allow anyone with access to some public key to verify that a message or piece of data was signed by the owner of the associated private key.

These three core primitives are crucial building blocks for an extensive list of advanced primitives, protocols and applications, and are used daily in any digital devices connected to other devices. Hence, improving the cryptanalysis or performance of particular NIKEs, KEMs or signatures is essential to ensure safety, security and speed in our day-to-day lives. As we will see later, code-based and isogeny-based primitives both face challenges in terms of practicality.

1.1 NON-INTERACTIVE KEY EXCHANGE

Informally, a non-interactive key exchange allows two parties $\mathcal{A}$ and $\mathcal{B}$ to agree on some shared value K, with $\mathcal{A}$ only needing to know some public key pk_B of $\mathcal{B}$ and vice versa. A secret key sk is associated to such a public key pk, with the secret key being known only to one party, whereas pk may be known to anyone. Such a pair $(\mathsf{sk}, \mathsf{pk})$ is generated by an algorithm known as Keygen which takes as input the security parameter λ. Deriving the shared value K requires an algorithm Derive, which returns the same value for both parties. Altogether, we get the following definition.

Definition 1. A *non-interactive key exchange (NIKE)* is a collection of two algorithms Keygen and Derive, such that

- Keygen is a probabilistic algorithm that on input 1^λ, with λ the security parameter, outputs a *keypair* $(\mathsf{sk}, \mathsf{pk})$, and
- Derive is a deterministic algorithm that on input a public key pk and a secret key sk outputs a *shared key* K.

A NIKE is *correct* if for every two pairs $(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A}) \leftarrow \mathsf{Keygen}(1^\lambda)$ and $(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B}) \leftarrow \mathsf{Keygen}(1^\lambda)$ it holds that $\mathsf{Derive}(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}) = \mathsf{Derive}(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{A})$.

Non-interactive refers to the fact that, given the public information pk of some party $\mathcal{X}$, any party $\mathcal{Y}$ with private key sk' can derive a shared key $\mathsf{Derive}(\mathsf{sk}', \mathsf{pk})$ without any interaction with $\mathcal{X}$.

PRE-QUANTUM KEY EXCHANGE. The quintessential example of a NIKE is the *Diffie-Hellman-Merkle key exchange* [158, 272], usually given for finite fields or elliptic curves, and more generally applicable to any cyclic group G. As a scheme parameter, it requires a generator g of order q for G. It relies on the core equality

$$(g^a)^b = g^{ab} = (g^b)^a$$

for positive integers a and b, which allows two parties $\mathcal{A}$ and $\mathcal{B}$ to compute g^{ab}, where $\mathcal{A}$ only needs to know her (secret) value $\mathsf{sk}_\mathsf{A} := a$ and the public value $\mathsf{pk}_\mathsf{B} := g^b$, and similarly, $\mathcal{B}$ can compute g^{ab} given $\mathsf{pk}_\mathsf{A} := g^a$ and $\mathsf{sk}_\mathsf{B} := b$. In this example, Keygen simply samples a random positive integer $a \in \mathbb{Z}_q$ and returns the pair (a, g^a), whereas Derive computes g^{ab} given g^b and a. We summarize this protocol with Figure 3.

It is easy to see that this cryptographic scheme would be completely broken if one could derive either a from g and g^a or g^{ab} from g, g^a and g^b, as this

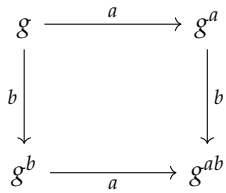


Figure 3: Diffie-Hellman-Merkle key exchange.

would allow an adversary without any of the required secret knowledge to derive the secret value K that should only be accessible to $\mathcal{A}$ and $\mathcal{B}$. We capture this in *hardness assumptions*, or equivalently, the diffuclty of mathematical problems. For Diffie-Hellman-Merkle key exchange, we want the *computational Diffie-Hellman problem* (CDH) to be a hard problem.

Problem 1 (Computational Diffie-Hellman problem)**.** Let G be a group. The *computational Diffie-Hellman problem* is:

$$\text{Given } g, g^a, g^b \in G, \quad \text{compute } g^{ab}.$$

Sometimes, hardness of the decisional variant is required: in this case, one is given g, g^a, g^b and some z, which is either a random value g^c or the value g^{ab}. The goal is then to decide if z is random or $z = g^{ab}$. The other problem, deriving a from g and g^a is the *discrete logarithm problem*.

Problem 2 (Discrete logarithm problem)**.** Let G be a group. The *discrete logarithm problem* is:

$$\text{Given } g, h \in G, \quad \text{find } a \in \mathbb{N} \text{ such that } h = g^a.$$

Solving the discrete logarithm problem implies that you can solve CDH too, which we call a *reduction*: we say that CDH reduces to the discrete logarithm problem.

For Diffie-Hellman-Merkle key exchange, we usually require a (sub)group G of prime order q, and have to choose this group G and the generator g specifically so that Problem 2 is cryptographically hard, i.e. there exist no polynomial time algorithms to find a given g and h.

If the group G is selected carefully, for example as the group of points on a specific elliptic curve E over a finite field $\mathbb{F}_q$, this problem remains secure for classical adversaries. Unfortunately, Shor [351] shows that an adversary with access to a quantum computer can break the discrete logarithm problem

in polynomial time, and therefore, by reduction, also break CDH and any cryptography built on top.

POST-QUANTUM KEY EXCHANGE. In post-quantum cryptography, we require our problems to remain cryptographically hard, even with access to a quantum computer. As of the moment of writing, there are two classes of post-quantum NIKEs available:

1. CSIDH [96], and generalizations [143], which is the subject of Part 1 of this thesis, and
2. Swoosh [183], a lattice-based key exchange originally considered "folklore", with a first proper attempt given by de Kock [152].

Both NIKEs are, at the moment of writing, practically computable, yet far from practical: The performance and security of CSIDH will be analyzed in Part 1, whereas Swoosh suffers from large public keys.

1.2 KEY-ENCAPSULATION MECHANISMS

A key-encapsulation mechanism (KEM) achieves a similar goal to a NIKE: sharing some key between two parties. The main difference is that a KEM requires interaction between these two parties. More precisely, in key-encapsulation we again generate a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Keygen}(1^\lambda)$ for each party, but the public key pk is now used to encapsulate some key K in a ciphertext ct using an algorithm Encaps. The associated sk is required to compute K from ct. Thus, any party that knows pk can get a K from Encaps (and keeps it secret) and knows that only the owner of sk can compute the same K too, using Decaps. Altogether, we get the following definition.

Definition 2. A *key-encapsulation mechanism (KEM)* is a collection of three algorithms Keygen, Encaps, and Decaps, such that

- Keygen is a probabilistic algorithm that on input 1^λ, with λ the security parameter, returns a keypair $(\mathsf{sk}, \mathsf{pk})$, and
- Encaps is a probabilistic algorithm that on input a public key pk returns returns a ciphertext ct and a key K, and
- Decaps is a deterministic algorithm that on input a secret key sk and a ciphertext ct returns a key K.

A KEM is *correct* if for every pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Keygen}(1^\lambda)$ and $(\mathsf{ct}, \mathsf{K}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$, it holds that $\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}) = \mathsf{K}$.

DIFFERENCE WITH NIKE. Although both NIKEs and KEMs are used to exchange a key K, we can be more flexible with a NIKE as it is non-interactive, which explains the abundance of applications using the Diffie-Hellman-Merkle NIKE in pre-quantum cryptography. In a post-quantum setting, the limitations of current NIKEs prevent them from being used in most settings. In general, the usage of NIKEs and KEMs in protocols falls into three categories:

1. Some protocols require an interaction by nature, where using a NIKE instead of a KEM is possible but does not offer any additional benefit. A traditional example is the ephemeral key exchange in TLS 1.3, where the current usage of Diffie-Hellman-Merkle can migrate to post-quantum KEMs [64, 340].

2. Some protocols require a NIKE by nature, which cannot be replaced by a KEM. An example is Signal's X3DH protocol [267], which still needs to work when participants are offline and can therefore not interact.

3. Some protocols can use KEMs but at some additional cost, usually an extra round of communication or at the loss of some security property. In such instances, the comparison between post-quantum NIKEs or KEMs is especially interesting. Chapter VII discusses this topic in more detail.

SECURITY. Freire, Hofheinz, Kiltz, and Paterson [181] describe precise security notions for NIKEs and Cramer and Shoup [140, § 7] for KEMs. For our purposes, we need to know the notion of *indistinguishability* for encryption, relative to different models of strength for an adversary.

Definition 3. A public-key encryption scheme has the security property *indistinguishability* (IND) when any polynomial-time adversary $\mathcal{A}$ cannot distinguish the encryption of any two messages of the same length, even if $\mathcal{A}$ chooses the messages.

Formalised by a game-based description, let $\mathcal{A}$ be a randomised polynomial-time algorithm, given a public key pk by some challenger. First, $\mathcal{A}$ outputs two messages msg_0 and msg_1 of equal length. The challenger randomly encrypts either msg_0 or msg_1 and returns the encryption c. The adversary is successful if it outputs the bit b such that the encryption of msg_b is c. The encryption scheme has the indistinguishability property if there are no polynomial-time adversaries with non-negligible success probabilities for this game.

In the above example, the adversary receives only the public key pk and the ciphertext c before it has to decide on $b = 0$ or $b = 1$. This attack model is called

chosen-plaintext attack (CPA) and we may then say that the encryption scheme is *IND-CPA* secure: it has the indistinguishability property when considering the attack model CPA. We may also give the adversary more power: the *adaptive chosen-ciphertext attack* (CCA) allows the adversary access to an oracle that provides decryption of any ciphertext $c' \neq c$, which the adversary may query before and after receiving c, before deciding on $b = 0$ or $b = 1$. IND-CCA security is the strongest security notion of encryption schemes.

For KEMs, the natural analogue of IND-CCA security replaces encryption and decryption by encapsulation and decapsulation of keys. For this security notion, a polynomial-time adversary receives a public key pk, can query a decapsulation oracle, and then receives a ciphertext ct, together with K', where K' set to $\mathsf{K}' \leftarrow \mathsf{K}$ from an actual encapsulation $(\mathsf{K}, \mathsf{ct}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$ or to a random bitstring of the same length, chosen with probability $1/2$. The adversary can again query the decapsulation oracle, with any ciphertext except ct, and has to decide whether K' is actually the encapsulated key or a random string. Thus, KEMs are IND-CCA secure when no polynomial-time algorithms exists which have a non-negligible succes probability at distinguishing real encapsulated keys K from random strings of the same length [140].

It is possible to convert a NIKE into a KEM, assuming some technical requirements [181, § 5.1]. The analogue of indistinguishability then asks a polynomial-time adversary to distinguish between a properly derived shared secret and a random key. Thus, when applied to the the Diffie-Hellman-Merkle NIKE, this is the decisional Diffie-Hellman problem.

1.3 DIGITAL SIGNATURES

A digital signature allows some party $\mathcal{A}$ to compute a signature σ for some piece of data, usually a message msg, using its secret key sk with an algorithm called Sign. Any party $\mathcal{X}$ with the associated public key pk can verify this signature σ with an algorithm called Verif, which assures $\mathcal{X}$ that $\mathcal{A}$ computed σ for msg. A successful verification of a message msg is usually understood as *authenticating* $\mathcal{A}$ as the author of msg, as noone else should be able to create a signature that verifies under pk. It furthermore implies that the message msg is not altered, thus the signature provides *integrity* of the message. Altogether, we get the following definition.

Definition 4. A *signature scheme* is a collection of three algorithms Keygen, Sign, and Verif, such that

- Keygen is a probabilistic algorithm that on input 1^λ, with λ the security parameter, returns a keypair $(\mathsf{sk}, \mathsf{pk})$, and
- Sign is a probabilistic algorithm that on input a message msg and a secret key sk returns a signature σ, and
- Verif is a deterministic algorithm that on input a public key pk, a message msg and a signature σ returns either "valid" or "invalid".

A signature scheme is *correct* if for every pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Keygen}(1^\lambda)$ and signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, \mathsf{sk})$, it holds that $\mathsf{Verif}(\mathsf{msg}, \sigma, \mathsf{pk})$ returns "valid".

SECURITY. Similar to the security notions for encryption schemes, KEMs and NIKEs, signature schemes have security strengths defined relative to the attacker model. We define these security notions in terms of attack goals, that is, we define what *should not* happen for a secure signature scheme.

A signature scheme has a

- *total break*, when an adversary can obtain the secret key associated to a given public key,
- *selective forgery*, when an adversary can generate valid signatures for any message for a given public key,
- *existential forgery*, when an adversary can generate a valid signature for at least one message for a given public key.

The strongest security notion for signature schemes is thus the last one: the inexistence of existential forgeries.

Definition 5. A signature scheme has the security property *existential unforgeability* (EUF) when any polynomial-time adversary $\mathcal{A}$ cannot create existential forgeries for a given public key other than with negligible succes probability.

When the adversary is only given the public key, we call the attack model *passive attack*. The standard notion for security, however, allows an adversary access to a signing oracle that generates signatures for any message for the given public key, before forging a signature on a message not previously queried. This model is known as the *adaptive chosen-message attack* (CMA), and thus, we call signature schemes with the existential unforgeability notion under this model *EUF-CMA* secure. This is the notion for security that we will mostly use to consider a signature scheme secure.

PRE-QUANTUM SIGNATURES. The most elegant yet simple and natural example of pre-quantum signatures is the *Schnorr signature scheme* [337], which is based on the discrete logarithm problem (Problem 2), described by the following *interactive* protocol.

Protocol 1. Schnorr identification protocol

$\mathcal{A}$, knows sk_A, pk_A		$\mathcal{B}$, knows pk_A
$r \xleftarrow{\$} \mathbb{Z}_q^*,\ R \leftarrow g^r$		
$\mathsf{cmt} \leftarrow R$		
	$\xrightarrow{\mathsf{cmt}}$	
		$c \xleftarrow{\$} \mathbb{Z}_q$
		$\mathsf{ch} \leftarrow c$
	$\xleftarrow{\mathsf{ch}}$	
$s \leftarrow r - c \cdot \mathsf{sk}_\mathsf{A}$		
$\mathsf{resp} \leftarrow s$		
	$\xrightarrow{\mathsf{resp}}$	
		$R \stackrel{?}{=} g^s \cdot \mathsf{pk}_\mathsf{A}{}^c$

The core idea is that $\mathcal{A}$ is the only one that can compute s as it requires knowledge of sk_A, and so, by confirming that the commitment R equals $g^s \cdot \mathsf{pk}_\mathsf{A}{}^c$, party $\mathcal{B}$ is convinced that $\mathcal{A}$ really does know sk_A.

To turn Protocol 1 into a non-interactive signature scheme, both $\mathcal{A}$ and $\mathcal{B}$ agree on a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$, and we replace the challenge $c \leftarrow \mathbb{Z}_q$ by the hash of R (as a bit string) concatenated with msg to get $c \leftarrow \mathsf{H}(R \,||\, \mathsf{msg})$. The resulting $s \leftarrow r - c \cdot \mathsf{sk}_\mathsf{A} \mod q$ together with c becomes the signature $\sigma \leftarrow (s, c)$. To verify, we recompute $R \leftarrow g^s \cdot \mathsf{pk}_\mathsf{A}{}^c$ and ensure that $\mathsf{H}(R \,||\, \mathsf{msg})$ equals c.

SIGMA PROTOCOLS AND THE FIAT-SHAMIR TRANSFORM. The method of the previous example to turn an identification protocol into a secure[1] signature scheme is a well-known method named the *Fiat-Shamir transform* [178] and applies to a broad range of identification protocols. In this thesis, we restrict ourselves to Sigma protocols [139] which are identification protocols based on the commitment-challenge-response structure we saw in Protocol 1. Such Sigma protocols assume two parties: a "prover" $\mathcal{A}$ and a "verifier" $\mathcal{B}$, with $\mathcal{A}$

1 In the (Q)ROM [162, 314].

proving knowledge of some public statement x tied to some secret knowledge w, often called the *witness*. For example, in Protocol 1, $\mathcal{A}$ convinces $\mathcal{B}$ that she knows the witness $w = \mathsf{sk}_A$ such that $x = \mathsf{pk}_A = g^{\mathsf{sk}_A}$. We use the following definition for Sigma protocols.

Definition 6. A *Sigma protocol* is a three-pass interactive protocol between two parties, a prover $\mathcal{A}$ and a verifier $\mathcal{B}$, following a commitment-challenge-response structure, using a commitment algorithm P_{cmt}, a response algorithm P_{resp}, and a verification algorithm V_{verif} which outputs **true** or **false**. Furthermore, $\mathcal{B}$ can sample uniformly random from a predefined set S, called the *challenge space*. To prove knowledge of a witness w for the statement x, prover $\mathcal{A}$ and verifier $\mathcal{B}$ execute Protocol 2.

Protocol 2. Sigma protocol

$\mathcal{A}$, knows w, x		$\mathcal{B}$, knows x
$\mathsf{cmt} \leftarrow P_{\mathsf{cmt}}$		
	$\xrightarrow{\mathsf{cmt}}$	
		$\mathsf{ch} \xleftarrow{\$} S$
	$\xleftarrow{\mathsf{ch}}$	
$\mathsf{resp} \leftarrow P_{\mathsf{resp}}(x, w, \mathsf{cmt}, \mathsf{ch})$		
	$\xrightarrow{\mathsf{resp}}$	
		$V_{\mathsf{verif}}(x, \mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$

If the *transcript* $(\mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$ verifies x, i.e. $V_{\mathsf{verif}}(x, \mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$ outputs **true**, we call the transcript *valid* for x.

A Sigma protocol will only be useful if a prover $\mathcal{A}$ can actually convince $\mathcal{B}$, that is, if only $\mathcal{A}$ who knows a witness w for x can generate a valid transcript for x, instead of some adversary[2]. Furthermore, we want that such a transcript can *always* be verified by $\mathcal{B}$. These are three distinct properties of a Sigma protocol for which we provide an intuitive explanation. Formal definitions are abundant in the literature [139, 236].

- A Sigma protocol is **complete** if a verifier that actually knows the secret w for x will be able to generate a valid transcript. More precisely, any honestly generated transcript $(\mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$ for x is valid: $V_{\mathsf{verif}}(x, \mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$ passes.

2 With overwhelming probability.

- A Sigma protocol is **honest-verifier**[3] **zero-knowledge** if the verifier $\mathcal{B}$ gains no knowledge on the secret w from the transcript $(\mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$. Zero-knowledge comes in different gradations:
  - the zero-knowledge is *perfect* if it is mathematically impossible to get information from the transcript,
  - the zero-knowledge is *statistical* if the distribution of valid transcripts is indistinguishable from a random transcript, and
  - the zero-knowledge is *computational* if differentiating between a valid transcript and a random transcript is a computationally hard problem.
- A Sigma protocol is **special sound** if, given two transcripts for the same cmt but different ch, it is possible to derive w in polynomial time. This implies that someone without knowledge of w can only convince the verifier with probability $1/|S|$, essentially by guessing the right response resp.

We call a single execution of the Sigma protocol a *round*. Repeating the protocol a number of rounds before applying the Fiat-Shamir transform increases the soundness: it becomes much harder for an adversary to cheat every round.

SECURITY. When we apply the Fiat-Shamir transform to a Sigma protocol that is complete, honest-verifier zero-knowledge, and special sound, we obtain a signature scheme that is EUF-CMA secure, given that the challenge space is large enough [236]. Thus, we may repeat Protocol 2 for t rounds, with t so that $|S|^t > 2^\lambda$ to obtain a λ-bit EUF-CMA secure signature scheme.

§2 Cryptographic Group Actions

Cryptographic group actions are a useful framework to unify several different approaches to designing signature schemes, based around the concept of a *group action*,

$$\star : G \times X \to X, \quad (g, x) \mapsto g \star x,$$

where $\star$ behaves well with regard to the group structure. We first discuss several properties of group actions, and what makes a group action *cryptographic*. We then apply cryptographic group actions to design cryptographic protocols,

3 That is, we assume that verifier $\mathcal{B}$ follows the protocol.

whose security reduces to the hardness of recovering the group element that acted. This approach has led to an abundance of schemes, and we discuss these in Part 1 and Part 3.

2.1 PROPERTIES OF CRYPTOGRAPHIC GROUP ACTIONS

Let G be a group and X a set. Denote by $\star$ a group action of G on X, that is, a function

$$\star : G \times X \to X, \quad (g, x) \mapsto g \star x,$$

such that $e \star x = x$ for all $x \in X$, with e the neutral element of G, and $g_1 \star (g_2 \star x) = (g_1 \cdot g_2) \star x$ for all $x \in X$ and all $g_1, g_2 \in G$. Several properties of group actions are particularly relevant when used in a cryptographic context.

Definition 7. Let G act on X. The group action is said to be

- *transitive*, if for every $x, y \in X$ there is a $g \in G$ such that $y = g \star x$,
- *faithful*, if no $g \in G$, except the neutral element, acts trivially on X, i.e. $g \star x = x$ for all $x \in X$,
- *free*, if no $g \in G$, except the neutral element, acts trivially on any element of X, i.e. $g \star x = x$ for one $x \in X$ implies $g = e$,
- *regular*, if the group action is both transitive and free, which implies there is a unique $g \in G$ for any two $x, y \in X$, such that $y = g \star x$.

Beyond these mathematical properties, we require several cryptographic properties for a group action to be useful in cryptography.

Definition 8. Let G act on X. The group action is *cryptographic* if

- *vectorisation* is hard, that is, given $y = g \star x$ and x, find g
- *parallellisation* is hard, that is, given x, $g_1 \star x$ and $g_2 \star x$, find $(g_1 \cdot g_2) \star x$.

Vectorisation is sometimes called the *Group Action Inverse Problem*, or GAIP, and parallellisation is sometimes called the *computational Group Action Diffie-Hellman Problem*, because of the following example.

Example 1. Let the action of G on X be a regular, commutative group action. Then Figure 4 defines a Diffie-Hellman-like non-interactive key exchange, with $g_1, g_2 \in G$ secret keys, x_0 a system parameter, $x_1, x_2 \in X$ public keys and $x_{12} \in X$ the shared secret.

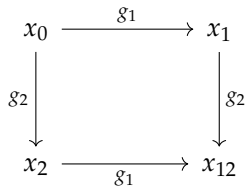


Figure 4: A group action-based NIKE.

The similarity with Figure 3 is not a coincidence as any Diffie-Hellman-Merkle key exchange in a cyclic group G of order q can similarly be defined as a group action-based key exchange, by the group action $\mathbb{Z}_q^* \times G \to G$ given by $(a, g) \mapsto g^a$.

A *cryptographic group action* is secure to use for protocol design, but may still not be very useful. This depends on yet a third set of properties, defined in [7], adapted here to our context.

Definition 9. Let G act on X, and let both be finite. The group action is *efficient* if the following procedures have efficient (probabilistic polynomial time) algorithms.
For the group G,

- *membership testing*, that is, to decide if some element is a valid element of G,
- *equality testing*, that is, to decide if two elements represent the same element of G,
- *sampling*, that is, to sample (statistically close) to uniformly random from G,
- *multiplication*, that is, to compute $g_1 \cdot g_2$ given any $g_1, g_2 \in G$,
- *inversion*, that is, to compute g^{-1} given any $g \in G$.

For the set X,

- *membership testing*, that is, to decide if some element is a valid element of X,
- *uniqueness of representation*, that is, to give a unique representation x for any arbitrary $x \in X$.

Furthermore, most crucially, *evaluation* must be efficient, that is, to compute $g \star x$ given any $g \in G$ and any $x \in X$.

Beyond these properties, commutativity of the group action is especially nice to have. As shown in Example 1, a commutative group action allows a simple construction of a NIKE. Part 1 explores the practicality of precisely such a NIKE, based on the only known post-quantum commutative cryptographic group action.

If the group action is not commutative, we can still construction signature schemess. We explore this approach for a specific code-based cryptographic group action in Part 3. The general construction of such signature schemes from group actions is the topic of Chapter XIV.

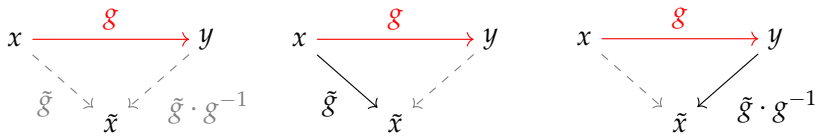
2.2 SIGMA PROTOCOLS FROM CRYPTOGRAPHIC GROUP ACTIONS

Given a cryptographic group action, it is easy to define a Sigma protocol and thus, after applying the Fiat-Shamir transform, a digital signature scheme [71, 196]. The central idea is to prove knowledge of a secret element $g \in G$, with the pair $(x, y = g \star x) \in X \times X$ as the public key. The Sigma protocol is then given by Protocol 3.

Protocol 3. Sigma protocol from group action.

$\mathcal{A}$, knows g, (x, y)		$\mathcal{B}$, knows (x, y)
$\tilde{g} \xleftarrow{\$} G,\ \tilde{x} \leftarrow \tilde{g} \star x$ $\mathsf{cmt} \leftarrow \tilde{x}$		
	$\xrightarrow{\mathsf{cmt}}$	
		$\mathsf{ch} \xleftarrow{\$} \{0,1\}$
	$\xleftarrow{\mathsf{ch}}$	
If $\mathsf{ch} = 0 : \mathsf{resp} \leftarrow \tilde{g}$ If $\mathsf{ch} = 1 : \mathsf{resp} \leftarrow \tilde{g} \cdot g^{-1}$		
	$\xrightarrow{\mathsf{resp}}$	
		If $\mathsf{ch} = 0 : \mathsf{cmt} \stackrel{?}{=} \mathsf{resp} \star x$ If $\mathsf{ch} = 1 : \mathsf{cmt} \stackrel{?}{=} \mathsf{resp} \star y$

The crucial idea is that only $\mathcal{A}$ can compute both paths $x \to \tilde{x}$ and $y \to \tilde{x}$ as visualised in Figure 5.



(a) The commitment. **(b)** If ch $= 0$, send $\tilde{g}$. **(c)** If ch $= 1$, send $\tilde{g}g^{-1}$.

Figure 5: Simple one-round Sigma protocol based on group actions.

We call a single execution of this protocol a *round*. After $t = \lambda$ succesful rounds, where $\mathcal{B}$ may each time ask for either of these paths, party $\mathcal{B}$ should be convinced that $\mathcal{A}$ knows g, as an adversary that does not know g would only have $1/2$ chance to cheat per round. This is the starting Sigma protocol, which we denote Π_{basic}, on which many advanced Sigma protocols are based. More concretely, in Π_{basic}, we commit to t values $\tilde{x}_i = \tilde{g}_i \star x$, sample a challenge ch from $\{0,1\}^t$, and respond with a vector resp where $\mathsf{resp}_i = \tilde{g}_i \cdot g^{-\mathsf{ch}_i}$.

One can easily show that Π_{basic} is complete, honest-verifier zero-knowledge, and 2-special sound. Hence, the security reduces to the vectorisation problem with x and $y = g \star x$. An easy calculation shows that, to reach λ-bit security by repeating $t = \lambda$ rounds, we need to communicate at least t group elements, and a verifier needs to perform at least t group action evaluations. This may make the scheme ineffective, as the signature could become too large, or signing and verifying signatures might become very slow.

An early construction of signature schemes derived from Sigma protocols using cryptographic group actions was given in Stolbunov's thesis [359]. A general framework based on group actions was explored in more detail by [7], allowing for the design of several primitives. Chapter XIV systematically analyses more advanced Sigma protocols starting from Π_{basic} to reduce signature sizes and improve the efficiency of signing and verification.

II

CURVES & ISOGENIES

Part 1 and Part 2 of this thesis concern themselves with two subareas of isogeny-based cryptography, i.e. cryptography based on isogenies between abelian varieties. Luckily, most of this thesis concerns itself with isogenies between elliptic curves, which are well-known objects to any cryptographer, with (isogenies between) hyperelliptic curves[1] playing a much smaller role[2]. This chapter introduces the main players: curves, their isogenies and pairings on curves. We assume basic familiarity with elliptic curves, such as an understanding of Silverman [355] or Galbraith [188], and a basic understanding of quaternion algebras, for which Voight [375] is our main reference.

§1 Curves

All curves in this thesis are smooth, projective varieties of dimension 1. We denote the base field by k. For all practical purposes in this thesis, we may assume $k = \mathbb{F}_q$.

Definition 1. An *elliptic curve* $(E, \mathcal{O})$ is a smooth projective curve E of genus 1 together with a rational point $\mathcal{O} \in E$. We say an elliptic curve is defined over k whenever E is defined over k as a curve and $\mathcal{O} \in E(k)$.

Elliptic curves are pleasant to work with: we do not have to consider the more abstract algebraic geometry, discussing polarizations, duals and Jacobians,

1 There is discussion whether or not hyperelliptic curves include elliptic curves or not. Both Hartshorne [209] and Galbraith [188] exclude elliptic curves, and we follow suit. Silverman [354] unfortunately disagrees.

2 However, the field of isogeny-based cryptography is quickly moving to higher-dimensional isogenies, as described in Section 6. We thus require a deeper understanding of isogenies between abelian varieties beyond elliptic curves for parts of this thesis.

as each elliptic curve comes with a canonical principal polarization, and $E(k)$ is furthermore isomorphic to the Jacobian of E, ensuring a group structure on this set of points. Even more pleasantly, the fields we are working over never have characteristic 2 or 3, which ensures that every elliptic curve has an isomorphic curve E in *short Weierstrass form* defined by the affine equation

$$E : y^2 = x^3 + ax + b, \qquad a, b \in k.$$

Such a curve is smooth, i.e. non-singular, whenever $\Delta(E) = -16(4a^3 + 27b^2)$ is non-zero. Although E is given in affine form, we assume the reader understands that, properly speaking, we imply the projective closure with $\mathcal{O} = (0 : 1 : 0)$ the *point at infinity* not corresponding to any affine point. Thus, the set of points $E(k)$ contains precisely those points $P = (x, y)$ with $x, y \in k$ such that the equation holds, plus $\mathcal{O}$.

In practice, isogeny-based cryptography uses different curve models too, as the cost of crucial operations highly depends on the choice of model. A popular model was given by Montgomery [280].

Definition 2. A *Montgomery curve* E over k, with $\operatorname{char} k \neq 2$, is an elliptic curve E with affine equation

$$E : By^2 = x^3 + Ax^2 + x, \qquad A, B \in k$$

with $\Delta(E) = B(A^2 - 4)$ non-zero.

Whenever we work with Montgomery curves in this thesis, we can choose $B = 1$, and we refer to $A \in k$ as the *Montgomery coefficient* of the curve. This reduces smoothness to $A \neq \pm 2$. Furthermore, in such cases, or whenever it is clear, E_A refers to the Montgomery curve with Montgomery coefficient A. In particular, E_0 refers to the Montgomery curve with the affine equation

$$E_0 : y^2 = x^3 + x.$$

This curve has many 'nice' properties, hence E_0 appears frequently in isogeny-based cryptography as a system parameter.

1.1 CURVE ARITHMETIC USING ONLY THE x-COORDINATE

Even in the very first articles introducing elliptic curves in cryptography [240, 278], Miller and Koblitz independently noted that one could perform most operations required for elliptic curve cryptography using only the x-coordinate

of points $P \in E(k)$. In particular for curves in Montgomery form, such x-only approaches are significantly faster. This holds not only for classical ECC, but also for current-day isogeny-based cryptography.

As the x-coordinate of a point is equal for both P and $-P$, mathematically working with x-only arithmetic means an identification of these two. That is, we work on E with equivalence between P and $-P$. This turns out to be an algebraic variety $\mathcal{K}_E = E/\langle \pm 1 \rangle$ isomorphic to the projective line $\mathbb{P}^1$, named the *Kummer line* of E. The Kummer line $\mathcal{K}_E$ does not inherit the group structure of E due to the identification of P and $-P$. Nevertheless, $\mathcal{K}_E$ is a *pseudo-group*, which means we can still perform scalar multiplication and differential addition. This turns out to be enough for most cryptographic applications, such as a Diffie-Hellman key exchange based on the group $E(k)$, as it only requires the computation of $P \mapsto [n]P$ for $n \in \mathbb{Z}$.

§2 Hyperelliptic Curves

Beyond elliptic curves, we require an understanding of hyperelliptic curves. Such objects are well-studied in cryptography for their use in hyperelliptic curve cryptography (HECC) since Koblitz [241]. Hyperelliptic curves are curves of genus $g > 1$ with a ramified double cover of the projective line. As we assume we are working over fields with $\operatorname{char} k \neq 2$, such hyperelliptic curves are isomorphic to curves with a nice affine model.

Lemma 3. Any hyperelliptic curve over k, with $\operatorname{char} k \neq 2$, is isomorphic over k to a curve $\mathcal{C}$ given by the affine model

$$\mathcal{C} : y^2 = f(x), \qquad f(x) \in k[x]$$

with $\deg f = 2g + 1$ or $\deg f = 2g + 2$. The curve $\mathcal{C}$ is smooth if $\gcd(f, f') = 0$, e.g. f has no repeated roots.

Remark 1. The projective closures of the affine models given by Lemma 3 have a singular point at infinity. However, a birational non-singular model for the smooth completion exists and is therefore always implicitely implied.

The polynomial $f(x)$ may have roots $w_i \in k$, which implies points $(w_i, 0) \in \mathcal{C}(k)$. These points are the *Weierstrass points* of $\mathcal{C}$. In cryptographic applications, we again often require a special curve model suitable for our purposes. In the case of hyperelliptic curves of genus 2, a particularly interesting curve model is given by Rosenhain [334].

Definition 4. A hyperelliptic curve $\mathcal{C}$ over k of genus 2 is in *Rosenhain form* if we have

$$\mathcal{C} : y^2 = x(x-1)(x-\lambda)(x-\mu)(x-\nu), \qquad \lambda, \mu, \nu \in k.$$

The coefficients λ, μ, ν are called the *Rosenhain invariants*.

Hyperelliptic curves are isomorphic over k to a curve in Rosenhain form when $f(x)$ splits in $k[x]$, or equivalently, all Weierstrass points are rational. In this situation, by a Möbius transform κ, one can choose three Weierstrass points and map these to $\mathcal{O}$, $(0,0)$ and $(1,0)$. This determines an isomorphism κ, which determines $(\lambda, 0)$, $(\mu, 0)$ and $(\nu, 0)$ as the images of the other three Weierstrass points. We get a total of 720 possible choices for such a map κ, and therefore 720 possible Rosenhain forms for any hyperelliptic curve.

2.1 JACOBIANS OF HYPERELLIPTIC CURVES

Unfortunately, the set of points $\mathcal{C}(k)$ no longer have a group structure, as they did in the case of elliptic curves. Fortunately, to every hyperelliptic curve we can associate an abelian variety, the *Jacobian* $\mathcal{J}_\mathcal{C}$ of $\mathcal{C}$, which does carry the structure of the group[3]. Formally, a description of the curve provides us with a canonical principal polarization, and so we can understand $\mathcal{J}_\mathcal{C}$ as the *Picard variety of degree* 0, denoted $\mathrm{Pic}^0_k(\mathcal{C})$, which is the group of degree zero divisors quotiented out by principal divisors. For the purpose of this thesis, these formalities are of lesser importance, and we refer the reader to Cassels and Flynn [84] for an elaborate introduction. We mainly need to understand how elements of $\mathcal{J}_\mathcal{C}$ arise from points on $\mathcal{C}$, how we represent such elements, and how these form a group. We keep our introduction slightly informal, for brevity and clarity.

Any element $P \in \mathcal{J}_\mathcal{C}$ can be represented as a divisor D_P of the form

$$D_P = (P_1) + (P_2) + \ldots + (P_n)$$

with each $P_i \in \mathcal{C}$. When $n \leq g$, this representation is unique, and we call the representation *reduced*. A practical representation of divisors D_P is given by Mumford [285].

3 The set of points of an elliptic curve E form a group precisely because E is isomorphic to its Jacobian $\mathcal{J}_E$.

Definition 5. The *Mumford representation* of an element $D_P = \sum_{i=1}^{n}(P_i)$ with $P_i = (x_i, y_i) \in \mathcal{C}(\bar{k})$ is given by a pair of polynomials $a(x), b(x) \in \bar{k}[x]$, uniquely defined by the properties

$$a(x) = \prod x - x_i, \qquad b(x_i) = y_i$$

with $\deg a = n$ and $\deg b < \deg a$. The representation is denoted as $\langle a(x), b(x) \rangle$.

It can be shown that each reduced divisor $D_P \in \mathcal{J}_\mathcal{C}$ has a unique Mumford representation $\langle a(x), b(x) \rangle$. Furthermore, a divisor is defined over k whenever both $a(x)$ and $b(x)$ are defined over k. Note that this does not imply that each $P_i \in \mathcal{C}(k)$! For example, D_P can be rational whenever this divisor contains each Galois conjugate of $P_i \in \mathcal{C}(K)$ for some finite field extension K/k.

An efficient algorithm to add elements D_P and D_Q is given by Cantor [82], which takes the Mumford representations of D_P and D_Q and returns the Mumford representation of $D_P + D_Q$. This algorithm is a higher-dimensional generalization of the chord-tangent construction for the group operation on elliptic curves.

2.2 KUMMER SURFACES

Similar to the elliptic curve situation, arithmetic on the Jacobian $\mathcal{J}_\mathcal{C}$ using Cantor's algorithm is rather slow and not suitable for cryptography. Where for an elliptic curve E we could simply use the x-coordinate[4] which keeps enough structure to do cryptography, for hyperelliptic curves of genus 2 we can similarly work on a Kummer surface $\mathcal{K}_\mathcal{C} = \mathcal{J}_\mathcal{C}/\langle \pm \rangle$, for which models exist as algebraic varieties in $\mathbb{P}^3$. Several different models of Kummer surfaces are in use in cryptography. A more detailed introduction is given in Chapter XI.

§3 ISOGENIES

As the name suggests, isogenies are the main object of interest in isogeny-based cryptography. Shor's algorithm [351] breaks most curve-based cryptography that is based on the hardness of the discrete logarithm problem. However, so far, no polynomial-time quantum algorithms have been found for hard problems in isogeny-based cryptography. As cryptographic primitives in this domain provide key exchanges and signatures with key or signature sizes far smaller

4 More abstractly, we use $\mathbb{P}^1$ as a representation of the Kummer line $\mathcal{K}_E = E/\langle \pm 1 \rangle$

than other areas of post-quantum cryptography, isogeny-based cryptography has seen an explosion of interest in recent years. We formally define an isogeny only between elliptic curves, although they can be defined more generally for abelian varieties and even algebraic groups.

Definition 6. An isogeny between elliptic curves E and E' over k is a non-zero morphism

$$\varphi : E \to E',$$

such that $\varphi(\mathcal{O}_E) = \mathcal{O}_{E'}$. The isogeny φ induces a group homomorphism $E(k) \to E'(k)$.

Any isogeny is a rational map, and we say that φ is defined over k whenever φ can be written as a map

$$(x, y) \mapsto \left(\frac{f_1(x,y)}{f_2(x,y)}, \frac{g_1(x,y)}{g_2(x,y)} \right),$$

where f_i, g_i are polynomials in $k[x]$. Whenever an isogeny exists between two curves E and E', they are called *isogenous*. Surprisingly, it is easy to decide if two curves over a finite field are isogenous, due to Tate's theorem [364]: with $k = \mathbb{F}_q$, the curves E and E' are isogenous over $\mathbb{F}_q$ if and only if $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$. Tate's theorem holds in more generality, and in particular also for Jacobians of hyperelliptic curves. On the other hand, finding such an isogeny $E \to E'$ given only E and E' is presumably hard.

Problem 1. Given two isogenous elliptic curves E and E' over $k = \mathbb{F}_q$, find an isogeny $\varphi : E \to E'$.

Any isogeny φ furthermore induces a map $\varphi^* : k(E') \to k(E)$ between the function fields of E and E' by pre-composition, and the degree of φ is defined as the degree of the induced field extension $[k(E) : \varphi^* k(E')]$. The degree is multiplicative: $\deg(\varphi \circ \psi) = \deg \varphi \cdot \deg \psi$.

An isogeny is said to be *separable* if the field extension $\varphi^* k(E')/k(E)$ is separable. In particular, whenever the degree of an isogeny is coprime to $\operatorname{char} k$, the isogeny is separable.

Separable isogenies are pleasant to work with. It can be shown that the degree of a separable isogeny over a finite field is precisely the cardinality of its kernel.

Example 1. Let $k = \mathbb{F}_{11}$ and let $E : y^2 = x^3 + x$ and $E' : y^2 = x^3 + 5$. then

$$\varphi : (x, y) \mapsto \left(\frac{x^3 + x^2 + x + 2}{(x-5)^2}, y \cdot \frac{x^3 - 4x^2 + 2}{(x-5)^3} \right)$$

is a separable isogeny $E \to E'$ of degree 3. The kernel of φ are the three points $\mathcal{O}$, $(5,3)$ and $(5,-3)$.

Example 2. For any curve E over k and any n such that $\operatorname{char} k \nmid n$, the map $[n] : P \mapsto [n]P$ is a separable isogeny. As $E[n] := \ker[n] \cong \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$, the isogeny $[n]$ is of degree n^2.

Definition 7. Let E over k with $\operatorname{char} k = p$. The isogeny $\pi : (x,y) \mapsto (x^p, y^p)$ is the *Frobenius isogeny*. The Frobenius isogeny is a (purely) inseparable isogeny.

All inseparable isogenies essentially sprout from this single source, as it can be shown that any isogeny φ can be uniquely decomposed as

$$\varphi = \varphi_{\text{sep}} \circ \pi^k,$$

where φ_{sep} is a separable isogeny, up to composition with isomorphisms.

Any isogeny $\varphi : E \to E'$ naturally has a counterpart $\hat{\varphi} : E' \to E$ of the same degree called the *dual* of φ, with the property that $\hat{\varphi} \circ \varphi$ acts as multiplication by $\deg \varphi$ on E, and vice-versa that $\varphi \circ \hat{\varphi}$ acts as multiplication by $\deg \varphi$ on E'. This implies that φ is the dual of $\hat{\varphi}$, and furthermore that the isogeny $[n]$ is self-dual.

The set of isogenies between two elliptic curves E and E', together with the zero morphism[5] $[0] : E \to E'$, $P \mapsto \mathcal{O}$, is denoted $\operatorname{Hom}(E,E')$, with subsets $\operatorname{Hom}_k(E,E')$ for those isogenies defined over k. By pointwise addition $(\varphi + \psi)(P) := \varphi(P) + \psi(P)$ for $\varphi, \psi \in \operatorname{Hom}(E,E')$, we find an abelian group structure.

Computing isogenies.

The mathematical description of isogenies tells us nothing about how to find or compute such isogenies. It turns out that for separable isogenies there is a one-to-one correspondence between finite subgroups $G \leq E$ and separable isogenies $\varphi : E \to E'$, up to post-composition with an isomorphism. We use the notation E/G to denote the codomain of the isogeny φ_G associated to $G \leq E$. Velu's formulas [373] give an explicit method to compute φ given G. These formulas have time complexity $\mathcal{O}(\#G)$ and allow us to compute the codomain E/G, as well as compute $\varphi(Q) \in E/G$ for any $Q \in E$.

Any separable isogeny φ of degree $n = \prod \ell_i^{e_i}$ with ℓ_i prime can be decomposed into a sequence of isogenies of degrees ℓ_i, which allow us to compute φ

5 Some authors, notably Silverman [354], *do* consider $[0]$ an isogeny.

in e_i steps of complexity $\mathcal{O}(\ell_i)$. The $\sqrt{\text{élu}}$ algorithm [43] provides a (close to) square-root speedup over Velu's formulas to compute ℓ-isogenies in $\tilde{\mathcal{O}}(\sqrt{\ell})$.

3.1 ENDOMORPHISMS AND THE ENDOMORPHISM RING

Isogenies from E to itself are called *endomorphisms* and they turn out to be more crucial and interesting than one would at first glance surmise. We have seen two examples of endomorphisms already: the multiplication-by-n map $[n] : E \to E$ and the Frobenius $\pi : (x, y) \mapsto (x^p, y^p)$.

Definition 8. For an elliptic curve E over k, the set of endomorphisms of E is denoted $\mathrm{End}(E) := \mathrm{Hom}(E, E)$. With addition by $(\varphi + \psi)(P) = \varphi(P) + \psi(P)$ and multiplication $(\varphi \cdot \psi)(P)$ by composition $\varphi \circ \psi(P)$, we obtain the structure of a ring on $\mathrm{End}(E)$, called the *endomorphism ring* of E. The subring of k-rational endomorphisms is denoted $\mathrm{End}_k(E)$, or $\mathrm{End}_q(E)$ when $k = \mathbb{F}_q$.

The maps $[n] \in \mathrm{End}(E)$ for $n \in \mathbb{Z}$ define an embedding of $\mathbb{Z}$ in $\mathrm{End}(E)$, and similarly $\pi \in \mathrm{End}(E)$ then implies a subring $\mathbb{Z}[\pi] \subseteq \mathrm{End}(E)$ for any curve E over a field of non-zero characteristic. A more natural way to study the endomorphism ring is using the *endomorphism algebra* $\mathrm{End}^0(E) := \mathrm{End}(E) \otimes_{\mathbb{Z}} \mathbb{Q}$, with $\mathrm{End}(E)$ an order in $\mathrm{End}^0(E)$. The rank of $\mathrm{End}(E)$ neatly characterizes elliptic curves over finite fields.

Theorem 9. Let E be an elliptic curve over $k = \mathbb{F}_q$. Then the rank of $\mathrm{End}(E)$ as a $\mathbb{Z}$-module is either

- two, in which case we call E *ordinary*, and $\mathrm{End}(E)$ is a quadratic order in the quadratic field $\mathrm{End}^0(E)$,
- or four, in which case we call E *supersingular*, and $\mathrm{End}(E)$ is a maximal order in the quaternion algebra $\mathrm{End}^0(E)$.

Supersingular curves lie at the core of isogeny-based cryptography: we only consider supersingular curves in Parts 1 and 2. The main reason is that we have a fine control on the number of points on a supersingular curve. For example, assuming $p > 3$, any supersingular curve E over $\mathbb{F}_p$ has $p + 1$ points. As we can choose p freely, we can ensure E has points of order $\ell \mid p + 1$ with ℓ prime. This turns out to be useful in many isogeny-based protocols, due to the correspondence between subgroups and isogenies.

§4 The Action of the Class Group

Assume now a curve E, either ordinary or supersingular, with an endomorphism $\alpha : E \to E$ such that α^2 acts as $[-D] : P \mapsto [-D]P$ for some $D \in \mathbb{N}$. Thus, we may identify a subring of $\mathrm{End}(E)$ with $\mathcal{O} = \mathbb{Z} + \sqrt{-D}\mathbb{Z}$, by $\iota : \sqrt{-D} \mapsto \alpha$. Let K denote a imaginary quadratic field such that $\mathcal{O}$ is a quadratic order in K, then ι is an *$\mathcal{O}$-orientation* of E if it can be extended to an embedding $K \to \mathrm{End}^0(E)$ such that $\iota(\mathcal{O}) \subseteq \mathrm{End}(E)$. Such an orientation is called *primitive* if $\iota(\mathcal{O}) = \mathrm{End}^0(E) \cap \iota(K)$. If such a (primitive) $\mathcal{O}$-orientation exists, E is called an $\mathcal{O}$-oriented elliptic curve. Note that multiple $\mathcal{O}$-orientations may exist for the same curve E and order $\mathcal{O}$. In this section, we slightly abuse notation by not explicitly writing the embedding ι.

We quickly get an action of any non-zero ideal $\mathfrak{a} \subseteq \mathcal{O}$ on such curves: We define a kernel on E associated to $\mathfrak{a}$ by

$$E[\mathfrak{a}] = \bigcap_{\alpha \in \mathfrak{a}} \ker \alpha,$$

and use Velu's formulas to translate the kernel $E[\mathfrak{a}]$ into an isogeny $\varphi_{\mathfrak{a}} : E \to E/\mathfrak{a}$. As before, this is only efficient if the degree of $\varphi_{\mathfrak{a}}$ is smooth enough and $E[\mathfrak{a}]$ is either rational or defined over only a small field extension. Whenever $\mathfrak{a}$ is a principal ideal (ω), the kernel $E[\mathfrak{a}]$ is simply $\ker \omega$, and so $\omega : E \to E$ is an endomorphism. However, one can show that any isogeny $\varphi_{\mathfrak{a}}$ induced by a non-principal (fractional) ideal $\mathfrak{a}$ is never an endomorphism, and two fractional ideals $\mathfrak{a}$ and $\mathfrak{b}$ have isomorphic codomains only when $\mathfrak{a} = (\omega)\mathfrak{b}$ for some principal ideal (ω). In other words, the class group $\mathcal{Cl}(\mathcal{O})$ of fractional ideals quotiented out by principal ideals, acts (commutatively) on the set of $\mathcal{O}$-oriented curves E over k, which we denote $\mathcal{Ell}_k(\mathcal{O})$. Onuki [299] shows that this action $\mathcal{Cl}(\mathcal{O}) \times \mathcal{Ell}_k(\mathcal{O}) \to \mathcal{Ell}_k(\mathcal{O})$ is free and transitive, up to technical details that are not important in this thesis.

In theory, assuming vectorization and parallelization is hard for such a group action, we immediatly get the Diffie-Hellman-like non-interactive key exchange from Example I.1 given a starting curve E, Alice and Bob both sample their private keys as elements $\mathfrak{a}, \mathfrak{b} \in \mathcal{Cl}(\mathcal{O})$, their public keys are respectively $E/\mathfrak{a}$ and $E/\mathfrak{b}$, and they compute shared secrets by the action of $\mathfrak{a}$ on $E/\mathfrak{b}$ resp. $\mathfrak{b}$ on $E/\mathfrak{a}$ so that both derive $E/\mathfrak{ab}$, summarized in the following diagram.

However, the difficulty of using such a group action cryptographically lies in the efficiency of the group action: it is difficult to compute the action of randomly sampled elements of $\mathcal{Cl}(\mathcal{O})$ on random curves in $\mathcal{Ell}_k(\mathcal{O})$.

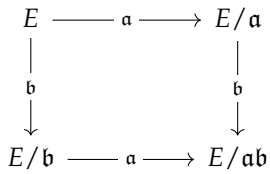


Figure 6: The class group action NIKE.

One soltion is to find an order $\mathcal{O}$ in a quadratic imaginary field K such that many small odd primes ℓ split in $\mathcal{O}$. CSIDH [96] finds a beautiful solution: by choosing primes of the form

$$p = 4 \cdot \ell_1 \cdot \ldots \cdot \ell_n - 1,$$

where the ℓ_i are small odd primes, then in the order $\mathcal{O} = \mathbb{Z} + \pi\mathbb{Z}$, with $\pi = \sqrt{-p}$ in the imaginary quadratic field $K = \mathbb{Q}(\sqrt{-p})$, each of these ℓ_i splits as

$$(\ell_i) = \mathfrak{l} \cdot \bar{\mathfrak{l}}, \qquad \mathfrak{l} = (\ell, \pi - 1), \quad \bar{\mathfrak{l}} = (\ell, \pi + 1).$$

Supersingular curves now have $\#E(\mathbb{F}_p) = p + 1$, and the Frobenius endomorphism π on E, defined over $\mathbb{F}_p$, has precisely the property that $\pi^2 = [-p]$. The set $\mathcal{E}\ell\ell_p(\mathcal{O})$ then consists precisely of those curves E with $\mathrm{End}_p(E) \xrightarrow{\sim} \mathbb{Z} + \pi\mathbb{Z}$. Furthermore, for any such curve E, we can easily compute $E[\mathfrak{l}]$ or $E[\bar{\mathfrak{l}}]$, because by definition

$$E[\mathfrak{l}] = \ker(\ell) \cap \ker(\pi - 1),$$

thus we need both $\pi P = P$ and $\ell P = \mathcal{O}_E$, which means $E[\mathfrak{l}]$ is precisely the set of $\mathbb{F}_p$-rational points of order ℓ on E. As $\ell \mid p + 1$ and supersingular curves have $\#E(\mathbb{F}_p) = p + 1$, we can easily find and compute with rational points of order ℓ, and thus compute the action of $\mathfrak{l}$, resp. $\bar{\mathfrak{l}}$ on $E \in \mathcal{E}\ell\ell_p(E)$.

We still face the difficulty that computing the action of randomly sampled elements of $\mathcal{C}\ell(\mathcal{O})$ is rather inefficient. Thus, CSIDH uses the heuristic assumption that $\mathcal{C}\ell(\mathcal{O})$ is generated by these ideals $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$, and instead samples random elements by

$$\mathfrak{a} = \prod \mathfrak{l}_i^{e_i}, \qquad e_i \in \mathbb{Z},$$

where $\mathfrak{l}_i^{-1}$ denotes $\bar{\mathfrak{l}}_i$, for large enough integers e_i. The action of $\mathfrak{a}$ can then be computed per $\mathfrak{l}_i$, that is, we decompose $\varphi_{\mathfrak{a}} : E \to E/\mathfrak{a}$ into its prime-degree components given by $\mathfrak{l}_i$.

The last problem that needs to be solved to get the efficient NIKE described above, is the efficiency of the membership test: upon receiving a curve E, how are we ensured that $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ before we act with a secret key? Precisely in the CSIDH situation, this simply requires us to verify that E is supersingular, which is again efficient. The practicality of CSIDH is the main topic of Part 1.

With respect to the properties defined in Section 2, we must stress that CSIDH is *not* an effective group action, as we cannot evaluate the action of every (randomly sampled) ideal. CSI-FiSh [54] solved this for one instantiation, by computing the structure of the class group $\mathcal{C}\ell(\mathcal{O})$ for the order $\mathcal{O}$ used in CSIDH-512. Recently, Clapoti(s) [304] solved this issue for *all* orders, by leveraging the power of higher-dimensional isogenies to compute the class group action in polynomial time.

4.1 ARITHMETIC OF CSIDH

The CSIDH scheme is based on the group action as described above: We apply each of the n different $\mathfrak{l}_i^{\pm 1}$ a number of times to a given curve E, and we denote this number by e_i. Hence, the secret key is some vector of n integers $(e_1, \ldots, e_n)$ defining an element $\mathfrak{a} = \prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$ which we can apply to supersingular curves E_A over $\mathbb{F}_p$. There is some variation between different proposals on where e_i is chosen from: The original proposal of CSIDH-512 picks $e_i \in \{-m, \ldots, m\}$ with $m = 5$, but one can also define individual bounds $m_i \in \mathbb{Z}$ per e_i. The key space is of size $\prod(2m_i + 1)$. For the original CSIDH-512 proposal with $m_i = 5$ and $n = 74$, this gives roughly size 2^{256}.

4.2 THE SECURITY OF CSIDH

While classical security imposes a restriction on the minimum key space size, quantum security usually poses more restrictive requirements. However, it is argued in [103] that for reasonable key spaces (that is, spaces large enough to achieve classical security), the quantum security of CSIDH relies only on the size of the prime p, regardless of the size of the actual key space being used. This is due to the fact that the most efficient quantum attack, Kuperberg's algorithm [247, 248], requires working with a group structure. Since the entire group representing all possible isogenies is of size roughly $\sqrt{p}$, this attack needs to search a space much larger than the key space itself, which only depends on n and the exponent bounds m_i. It is expected that a handful of $\mathfrak{l}_i$ already generate the entire group of size roughly $\sqrt{p}$. In a nutshell, classical

security is determined by the size of the key space, whereas quantum security is determined by the size of p, as long as the key space is not chosen particularly badly, e.g., as a small subgroup of the full class group.

While the original CSIDH proposal considered a 512-bit prime p sufficient for NIST Security Level I [96], its exact quantum security is debated [49, 60, 102, 307]. For instance, [102] claims that 4096-bit primes are required for NIST Security Level I. At larger prime sizes, the number n of small primes ℓ_i grows, and therefore it becomes natural to pick secret key vectors from $\{-1, 0, 1\}^n$ resp. $\{-1, 1\}^n$ for primes sizes of at least 1792 resp. 2048 bits. This allows for a large enough key space for classical security, while increasing p for sufficient quantum security.

4.3 CONSTANT-TIME IMPLEMENTATIONS

CSIDH is difficult to implement in constant time, as this requires that the timing of the execution is independent of the secret key $(e_1, \ldots, e_n)$. However, picking a secret key vector $(e_1, \ldots, e_n)$ translates to the computation of $|e_i|$ isogenies of degree ℓ_i, which directly affects the timing of the group action evaluation.

One way to mitigate this timing leakage is by using dummy isogenies: We can keep the total number of isogenies per degree constant by computing m_i isogenies of degree ℓ_i, but discarding the results of $m_i - |e_i|$ of these, effectively making them dummy computations [274, 276]. Several optimizations and different techniques have been proposed in the literature [98, 107, 301].

A noteworthy variant of constant-time implementations of CSIDH is CTIDH [20]. In contrast to sampling private key vectors such that $e_i \in \{-m_i, \ldots, m_i\}$, CTIDH uses a different key space that exploits the approach of batching the primes ℓ_i. We define *batches* $B_1, \ldots, B_N$ of consecutive primes of lengths $n_1, \ldots, n_N$, i.e., $B_1 = (\ell_{1,1}, \ldots, \ell_{1,n_1}) = (\ell_1, \ldots, \ell_{n_1})$, $B_2 = (\ell_{2,1}, \ldots, \ell_{2,n_2}) = (\ell_{n_1+1}, \ldots, \ell_{n_1+n_2})$, et cetera. We write $e_{i,j}$ for the (secret) coefficient associated to $\ell_{i,j}$. Instead of defining bounds m_i for each individual ℓ_i so that $|e_i| \leq m_i$, CTIDH uses bounds M_i for the batch B_i, i.e., we compute at most M_i isogenies of those degrees that are contained in B_i. That is, the key sampling requires $|e_{i,1}| + \cdots + |e_{i,n_i}| \leq M_i$. CTIDH then adapts the CSIDH algorithm such that the distribution of the M_i isogenies among degrees of batch B_i does not leak through the timing channel. Among other techniques, this involves Matryoshka isogenies, first introduced in [49], that perform the exact same sequence of instructions independent of its isogeny degree $\ell_{i,j} \in B_i$.

The main advantage of CTIDH is the ambiguity of the isogeny computations: From a time-channel perspective, a Matryoshka isogeny for B_i could be an

$\ell_{i,j}$-isogeny for any $\ell_{i,j} \in B_i$. Thus, in comparison to the previous CSIDH algorithms, CTIDH covers the same key-space size in fewer isogenies.

§5 The Deuring Correspondence

In the previous section, we have seen that $\mathrm{End}(E)$ for supersingular curves is isomorphic to some maximal order in a quaternion algebra $\mathcal{B}_{p,\infty}$. In more generality, the world of supersingular elliptic curves and their isogenies have many properties in common with the world of quaternion algebras and their ideals. The *Deuring correspondence*, after Max Deuring [157], provides us with the general bridge between these two worlds, mainly based on the functorial equivalence given by

$$\begin{aligned} \{\text{ss. curves over } \mathbb{F}_{p^2}\}/\sim &\to \{\text{max. orders in } \mathcal{B}_{p,\infty}\}/\sim \\ E &\mapsto \mathrm{End}(E) \end{aligned}$$

where $\sim$ on the left side is equivalence up to isomorphism and Galois conjugacy, and on the right side up to isomorphism. This thesis does not require an in-depth understanding of quaternion algebras, or the equivalence of categories. However, we sketch the situation informally so that we can understand SQIsign, an signature scheme whose complex signing procedure relies mainly on this correspondence. Part 2 of this thesis will focus only on the verification part of SQIsign, which can be understood in terms of elliptic curves and isogenies only. A reader can therefore safely skip the remainder of this section if they are only interested in SQIsign verification.

Similar to the previous section, the Deuring correspondence allows us to see isogenies $\varphi : E \to E'$ as ideals I_φ, but this time of a maximal order instead of a quadratic order. Again, the association relies on the kernel: all endomorphisms ω that vanish on $\ker \varphi$ form a left ideal I_φ of $\mathrm{End}(E)$, and similarly any non-zero left ideal I of $\mathrm{End}(E)$ gives a subgroup $E[I] \subseteq E$ by the intersection of the kernel of all generators of I, which gives a unique isogeny $\varphi_I : E \to E/I$, up to post-composition with an isomorphism. The *norm* of such an ideal I, the greatest common divisor of the norms of its elements, is then equal to the degree of φ_I.

The codomain associated to such a left ideal I of $\mathcal{O}$ should then be some 'natural' maximal order with I as a right ideal. This natural object is the *right order*[6]

$$\mathcal{O}_r(I) := \{\beta \in \mathcal{B}_{p,\infty} \mid I\beta \subseteq I\},$$

which is isomorphic to the endomorphism ring of the elliptic curve E/I. In such a situation, we say that I is a *connecting ideal* of two maximal orders $\mathcal{O}_1$ and $\mathcal{O}_2$, that is, when I is a left ideal of $\mathcal{O}_1$ whose right order is isomorphic to $\mathcal{O}_2$, or vice versa.

The KLPT algorithm.

The main algorithmic building block in SQIsign is a polynomial time algorithm by Kohel, Lauter, Petit, and Tignol [242], the KLPT algorithm. The idea is simple: we may often encounter isogenies $\varphi : E \to E'$ of some generic large degree d. Most likely, d is neither smooth, nor is the d-torsion rational or defined over a small extension field. If we could somehow, given φ construct another isogeny $\varphi' : E \to E'$ whose degree d' is very smooth, such as $d' = \ell^k$ for some small prime ℓ, we could easily compute φ' instead which could alleviate our burdens.

In the world of supersingular elliptic curves, this problem is presumably hard: finding such a φ' implies we find a non-trivial endomorphism $\hat{\varphi}' \circ \varphi$, which is the hard problem all isogeny-based cryptography is based on. However, if we know $\mathrm{End}(E)$, and can therefore use the Deuring correspondence to move this problem to the world of quaternion algebras by $\varphi \mapsto I_\varphi$, our problems are magically solved by the KLPT algorithm, as long as the domain of φ is of a special form. We leave out the details of this special form, and simply refer to *special* maximal orders $\mathcal{O}$. Informally, we can state the KLPT algorithm as follows.

Theorem 10 (The KLPT algorithm)**.** Let I be a left ideal of a special maximal order $\mathcal{O}$ in $\mathcal{B}_{p,\infty}$. There exists a heuristic polynomial-time algorithm that returns an ideal J, equivalent to I, of norm ℓ^k for some $k \in \mathbb{N}$.

The result is wonderful: instead of working with the isogeny/ideal of ghastly degree/norm d, we can apply KLPT to work with the ideal J of smooth norm ℓ^k. By translating such an ideal back to an isogeny[7], the isogeny $\varphi_J : E \to E'$ can be efficiently computed in polynomial time.

6 No pun intended.
7 A highly non-trivial and slow task, which we will not discuss in this thesis.

So far, we have ignored two main problems: first, are we ensured the maximal orders we will encounter 'in the wild' are of the special form, and second, how large is k in the output of KLPT? As an answer to the first question, it turns out that the special curve $E_0 : y^2 = x^3 + x$ has an endomorphism ring of precisely the special form we need, and this is all we require for the applications in SQIsign. For the second question, the norm of the output ideal can under some heuristic assumptions be estimated as $p^{7/2}$, and with the improvement by Petit and Smith [312] to p^3. In practice, this means that φ_J is a rather long isogeny, but not unpractically long.

The SQIsign signature scheme.

SQIsign [100, 148, 149] is a signature scheme that cleverly uses the Deuring correspondence and the KLPT algorithm to construct an identification scheme using a Sigma protocol. Although there are some significant details between subsequent versions of the scheme, the main idea is as follows.

We assume the starting elliptic curve E_0, whose endomorphism ring $\mathcal{O}_0 = \mathrm{End}(E_0)$, a special maximal order, is publicly known. Alice, the prover, samples a secret isogeny $\varphi_A : E_0 \to E_A$, and provides E_A as her public key. By knowing φ_A, she ensures she, and only she, knows $\mathrm{End}(E_A)$. Her goal is to prove this knowledge.

The Sigma protocol then starts with Alice committing to some curve E_1 which is the codomain of another random isogeny $\varphi_{\mathrm{com}} : E_0 \to E_1$. She sends E_1 to Bob, the verifier. Again, only Alice can compute $\mathrm{End}(E_1)$ as she keeps φ_{com} hidden. Bob samples a random isogeny $\varphi_{\mathrm{chall}} : E_1 \to E_2$ and communicates φ_{chall} to Alice, which allows her to compute $\mathrm{End}(E_2)$ using her knowledge of $\mathrm{End}(E_1)$. Alice thus knows $\mathrm{End}(E_A)$ and $\mathrm{End}(E_2)$, and furthermore can compute the ideals I_{secret}, I_{com}, I_{chall}. Thus, she can compute an ideal I_{resp} as the product $I_{\mathrm{chall}} \cdot I_{\mathrm{com}} \cdot \bar{I}_{\mathrm{secret}}$ to which she can apply (a generalization of) the KLPT algorithm to obtain a smooth equivalent ideal J. She then computes the associated isogeny $\varphi_J : E_A \to E_2$, which is her response $\varphi_{\mathrm{resp}} := \varphi_J$. Bob verifies the response by recomputing $\varphi_{\mathrm{resp}} : E_A \to E_2$ and confirms that the codomain is indeed (isomorphic to) the challenged E_2. The protocol is summarized by the Figure 19.

The SQIsign works ingeniously solve many obstacles that are not visible in the above description to allow Alice to sign, e.g. compute φ_{resp}, in practice, and to ensure the security of the scheme. However, with our main focus in Part 2 being SQIsign verification, we do not go in-depth on the signing procedure. Instead, we focus on the verification procedure.

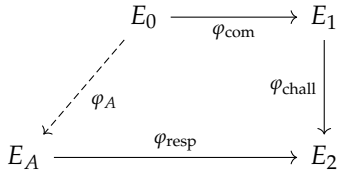


Figure 7: The SQIsign protocol as a diagram.

The ideal J associated with φ_{resp} is an output of the KLPT algorithm and therefore, as noted before, of rather large norm ℓ^k. In practice, we find $\deg \varphi_{\text{resp}} \approx p^{15/4}$. For NIST Level I security, where $\log p \approx 2\lambda = 256$ bits, the current parameters give $\deg \varphi_{\text{resp}} = 2^e$ with $e \approx 1000$. The maximal available rational $2^\bullet$-torsion on supersingular curves is determined by the largest f such that $2^f \mid p + 1$. For the given prime for NIST Level I security, this gives $f = 75$, and thus, in verification, we recompute φ_{resp} as a composition of $n := \lceil \frac{e}{f} \rceil$ isogenies $\varphi_i : E^{(i)} \to E^{(i+1)}$ of degree 2^f, such that $\varphi_{\text{resp}} = \varphi_n \circ \ldots \varphi_1$. In Part 2, in particular in Chapter IX and Chapter X, we analyze the performance of SQIsign verification in detail.

§6 Moving to Higher Dimensions

During the years that spanned the work in this thesis, SIDH/SIKE [177, 221] was spectacularly broken by higher-dimensional isogenies [86, 266, 329] using a (generalization of) Kani's lemma [231] combined with Zahrin's trick. From the ashes of SIDH arose a phoenix: higher-dimensional isogenies allow for an incredible constructive tool that increased the general flexibility for cryptodesign. This has sprouted numerous new schemes and will continue to provide significant improvements in isogeny-based cryptography as this paradigm shift to higher-dimensional isogenies is explored.

For this thesis, we sketch a quick understanding of higher-dimensional isogenies and their applications in variants of SQIsign. Although a detailed understanding is not required, a high-level understanding is useful to grasp the current landscape of isogeny-based cryptography and the precise role Chapter IX, X and XI play in our exploration of this landscape.

6.1 HIGHER-DIMENSIONAL ISOGENIES

By higher-dimensional isogenies, we refer to isogenies between Abelian varieties of dimension larger than 1. In practice, we usually encounter 2-dimensional Abelian varieties over some field k, and these are easily classified. Let A be a 2-dimensional Abelian variety, then A is isomorphic to a) the Jacobian of some hyperelliptic curve $\mathcal{C}$ of genus 2 over k, or b) the product of two elliptic curves E and E' over k, or c) the Weil restriction of an elliptic curve E over K, where K/k is a degree-2 field extension.

We specifically focus on isogenies $f : A \to B$ whose kernel as a subgroup is isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$, and furthermore maximally isotropic[8]. Such isogenies as maps between hyperelliptic curves are precisely the *Richelot isogenies*, which have appeared in hyperelliptic curve cryptography literature. In such cases, we say that f is a $(2,2)$-isogeny. A concatenation of n such $(2,2)$-isogenies, modulo some requirements on cyclicity, is then a $(2^n, 2^n)$-isogeny.

Kani's Lemma.

Kani's lemma [231] now allows for a very powerful tool: given the right setup, two 1-dimensional isogenies φ and ψ between elliptic curves can be associated with a specific 2-dimensional isogeny Φ. Computing Φ, as a 2-dimensional isogeny[9], will allow us to derive information on φ and ψ that was practically unavailable using only 1-dimensional isogenies. The right setup in this situation is an *isogeny diamond*.

Definition 11. Let $\psi : E_1 \to E_2$ and $\varphi : E_1 \to E_3$ be two isogenies of coprime degrees. An *isogeny diamond* is a square of isogenies

$$\begin{array}{ccc} E_1 & \xrightarrow{\varphi} & E_3 \\ \downarrow{\scriptstyle \psi} & & \downarrow{\scriptstyle \psi'} \\ E_2 & \xrightarrow{\varphi'} & E_4 \end{array}$$

such that $\varphi \circ \psi' = \psi \circ \varphi'$, with $\deg \varphi = \deg \varphi'$ and $\deg \psi = \deg \psi'$.

8 A technical requirement that we will not explore in more detail.

9 The reason we only consider $(2^n, 2^n)$-isogeny is precisely that we can only efficiently compute 2-dimensional isogenies of this degree, although recent progress also explores $(3,3)$-isogenies and beyond [118].

Kani's lemma provides the crucial 2-dimensional isogeny that arises from an isogeny diamond.

Lemma 12 (Kani's lemma)**.** Consider an isogeny diamond given by φ, φ', ψ and ψ', and let $N = \deg\varphi + \deg\psi$. then the map

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

given in matrix form as

$$\begin{pmatrix} \hat{\psi} & \hat{\varphi} \\ -\varphi' & \psi \end{pmatrix},$$

is an (N, N)-isogeny of abelian surfaces with kernel

$$\ker\Phi = \{(\varphi(P), \psi(P)) \mid P \in E_1[N]\}.$$

The main idea is now simple: Although φ and ψ individually might be difficult to compute, with the right set-up Φ might be efficient to compute, and can provide the same information.

Embedding isogenies.

A practical example of the above idea is given by Robert [330] to *embed an isogeny* φ of dimension 1 into a higher-dimensional isogeny Φ, so that evaluation of Φ allows us to evaluate φ.

Consider as a first example an isogeny $\varphi : E \to E'$, such that $2^n - \deg\varphi = x^2$ for some $n \in \mathbb{N}$ and $x \in \mathbb{Z}$. In such a case, we construct the isogeny diamond

$$\begin{array}{ccc} E & \xrightarrow{\varphi} & E' \\ \downarrow{\scriptstyle [x]} & & \downarrow{\scriptstyle [x]} \\ E & \xrightarrow{\varphi} & E' \end{array}$$

where $[x]$ denotes the simple-to-compute multiplication-by-x map $P \mapsto [x]P$. Kani's lemma now implies that $\Phi : E \times E' \to E \times E'$ is an (N, N)-isogeny with $N = \deg\varphi + \deg[x] = 2^n$, and so

$$\ker\Phi = \{(\varphi(P), [x]P) \mid P \in E[2^n]\}.$$

Thus, if we know $\varphi(P)$ for $P \in E[2^n]$ and we simply compute $[x]P$, we can compute $\varphi(Q)$ for any other $Q \in E$ using Φ, which we can efficiently compute as a $(2^n, 2^n)$-isogeny. Thus, we can 'embed' the 1-dimensional isogeny φ into the 2-dimensional isogeny Φ, if $\deg \varphi$ differs from a power of 2 by precisely a square.

This situation is highly unlikely to happen for any random isogeny φ, as only very few integers differ from a power of 2 by precisely a square. However, by Legendre's four-square theorem, any natural number can be represented as the sum of four squares. This inspired Robert [330] to generalize the result by Kani to a result between 8-dimensional abelian varieties $\Phi : E^4 \times E'^4 \to E^4 \times E'^4$. Then, instead of hoping $\deg \varphi$ precisely fits $2^n - x^2$ for some $x \in \mathbb{Z}$, we find four integers $x_1, x_2, x_3, x_4 \in \mathbb{Z}$ such that $2^n - \deg \varphi = x_1^2 + x_2^2 + x_3^2 + x_4^2$. A similar setup as before now *always* allows us to embed the 1-dimensional isogeny φ into an 8-dimensional isogeny Φ. Using several technical tricks, one can often already achieve a similar result with a 4-dimensional embedding.

6.2 HIGHER-DIMENSIONAL SQISIGN

The technique to embed 1-dimensional isogenies into higher dimensional isogenies is powerful and versatile. For example, whenever the 1-dimensional isogeny φ is of large non-smooth degree but embeddable in a smooth higher-dimensional isogeny Φ, we can simply evaluate φ on points using Φ, which is efficient to compute.

This has led to a revolution in many areas of isogeny-based cryptography, and most notably in SQIsign: instead of having to use KLPT to obtain a smooth response isogeny (of fairly large degree), we may embed the response isogeny into a smooth higher-dimensional isogeny Φ. Verification is then reduced to the efficient recomputation of Φ. To achieve this requires solving several technical difficulties which was done by Dartois, Leroux, Robert, and Wesolowski [141]. The resulting signature scheme, using 4- or 8-dimensional isogenies is named SQIsignHD and allows significantly faster signing than the original SQIsign. A drawback is slow verification, as computing 4- or 8-dimensional isogenies is much more complex than 1-dimensional isogenies.

Using an algorithmic building block by Nakagawa and Onuki [287], three works independently from each other were able to improve SQIsignHD to achieve verification using 2-dimensional isogenies only. These works, SQIsign2D-West [34], SQIsign2D-East [288] and SQIPrime [166], achieve both fast signing and verification, ushering in an era of practical higher-dimensional isogeny-based cryptography.

In summary, SQIsign2D is currently the most practical approach to isogeny-based signatures, with relatively fast signing and verification times on par with 1-dimensional SQIsign. However, 1-dimensional SQIsign is still an interesting topic for research, as many questions remain unanswered when comparing the real-world practicality of these two approaches. Chapter X deals with some of these questions.

§7 Pairings

Like dogs to humans, pairings are great companions for all of isogeny-based cryptography. Pairings, bilinear maps between Abelian groups, appear naturally in many versatile use cases due to their interesting properties, in particular in relation to isogenies of abelian varieties. This thesis contains many of these applications of pairings, mostly using the Tate pairing[10], though in general, the Weil pairing is more commonly used. Both are usually used in very specific and concrete situations, yet we will introduce these in their general forms, following Bruin [74], Garefalakis [191], and Silverman [353]. This allows us to provide more intuition to the reader in what these pairings are fundamentally doing. More properties, details and computational aspects are widely described in the literature, see [115, 184, 187]. Inspiration is also drawn from [93, 333].

7.1 THE WEIL PAIRING

Let $\varphi : E \to E'$ be an isogeny with $\operatorname{char} k \nmid \deg \varphi$. To φ, we can associate the generalized φ-Weil pairing

$$e_\varphi : \ker \varphi \times \ker \hat{\varphi} \to \overline{k}^*,$$

which is a bilinear, non-degenerate pairing invariant under the action of $\mathrm{Gal}(\overline{k}/k)$, taking as values m-th roots in $\overline{k}^*$, for m such that $\ker \varphi \subset E[m]$. We denote the cyclic group of m-th roots by μ_m. Several other desirable properties are described in [93, 187].

Taking $\varphi = [m] : E \to E$, we recover the more traditional Weil pairing

$$e_m : E[m] \times E[m] \to \mu_m.$$

10 More properly named the Tate-Lichtenbaum pairing, or also the Frey-Rück pairing, most authors stick to *Tate pairing*, perhaps simply because it is easiest to write.

Computing $e_m(P, Q)$ for two points $P, Q \in E[m]$ can be done using two Miller functions $f_{m,P}, f_{m,Q} \in \bar{k}(E)$, defined by the property that

$$\operatorname{div} f_{m,P} = m(P) - m(\mathcal{O}), \quad \operatorname{div} f_{m,Q} = m(Q) - m(\mathcal{O}).$$

We want to evaluate $f_{m,P}$ on the divisor $(Q) - (\mathcal{O})$, and vice versa. However, as $\mathcal{O}$ is a root of $f_{m,P}$, we must first find a divisor D_Q equivalent to $(Q) - (\mathcal{O})$, with support disjoint from $P, \mathcal{O}$, and similar for D_P. Then

$$e_m(P, Q) = f_{m,P}(D_Q)/f_{m,Q}(D_P).$$

Computing the Miller functions $f_{m,P}$ and $f_{m,Q}$ can be done using Miller's algorithm [277] in $\mathcal{O}(\log m)$.

7.2 THE TATE PAIRING

We focus on the case $k = \mathbb{F}_q$. Let $\varphi : E \to E'$ be an isogeny with $\ker \varphi \subset E[m] \subset E[q-1]$. To φ, we can associate the generalized φ-Tate pairing

$$T_\varphi : \ker \varphi(\mathbb{F}_q) \times \operatorname{coker} \hat{\varphi}(\mathbb{F}_q) \to \mathbb{F}_q^*/\mathbb{F}_q^{*m},$$

which is a bilinear, non-degenerate pairing invariant under the action of $\operatorname{Gal}(\bar{k}/k)$. The values are equivalence classes modulo m-th powers. Sometimes this is enough, but in practice, we often raise the value to the power $(q-1)/m$ to obtain specific m-th roots. We then call the Tate pairing *reduced*, and denote by t_φ the composition of T_φ with the exponentiation $z \mapsto z^{(q-1)/m}$, which is an isomorphism $\mathbb{F}_q^*/\mathbb{F}_q^{*m} \to \mu_m(\mathbb{F}_q)$. Several other desirable properties are described in [93, 187].

Computing $t_\varphi(P, Q)$ for $P \in \ker \varphi$, $Q \in E'$ can be done using the Miller function $f_{m,P}$ evaluated on a divisor D_Q equivalent to $(Q) - (\mathcal{O})$, with disjoint support, where again we can compute $f_{m,P}$ using Miller's algorithm. Taking $\varphi = [m]$ recovers the more traditional Tate pairing

$$t_m : E[m] \times E(\mathbb{F}_q)/[m]E(\mathbb{F}_q) \to \mu_m.$$

The case $\varphi = [m]$ also shows an immediate use case of the Tate pairing: as it is non-degenerate, we can identify the points $Q \in [m]E(\mathbb{F}_q)$ precisely as those points where $t_m(P, Q) = 1$ for all $P \in E[m]$. Conversely, this implies that $t_m(P, Q) \neq 1$ for some $P \in E[m]$ and $Q \in E(\mathbb{F}_q)$ immediately implies that $Q \notin [m]E(\mathbb{F}_q)$. Even more specific for $[m] = [2]$, for any Montgomery curve E_A,

which has $(0,0) \in E[2]$, this implies that if $t_2((0,0), Q) \neq 1$, then $Q \notin [2]E$. The value $t_2((0,0), Q)$ can be computed to be precisely the quadratic residue of x_Q. For supersingular curves E_A with $2^f \mid p+1$, this thus implies that the order of Q is divisible by 2^f, whenever x_Q is non-square. Thus, a classical result on the image of E under doubling [212, Thm. 4.1] can concisely be described using the Tate pairing. More generally speaking, we can use the φ-Tate pairing together with $\ker \varphi$ to determine the coset in $\operatorname{coker} \hat{\varphi}(\mathbb{F}_q)$ in which a point $Q \in E$ lies.

III

CODES & ISOMETRIES

Part 3 of this thesis concerns itself with code-based cryptography: our fundamental hardness assumption is a presumably difficult problem in coding theory. Contrary to classical code-based cryptography, the underlying hardness assumption on which much of the work in this part is based is not syndrome decoding, but code equivalence. We first describe codes, and then their isometries, to give an overview of the ingredients required for Part 3.

§1 Codes

This thesis only concerns itself with *linear codes*, e.g. subspaces $\mathcal{C}$ of a vector space V over a finite field $\mathbb{F}_q$, equipped with some metric d. Let n denote the dimension of V, and k the dimension of $\mathcal{C}$, then such codes are called $[n,k]$-codes. Any metric d defines a weight function w on V by $w(v) = d(v,0)$ where $0 \in V$ is the zero vector.

Most well-known are *Hamming metric* codes, where $V = \mathbb{F}_q^n$ and $d(v,v')$ simply counts the number of different coefficients $v_i \neq v'_i$, and the weight function wt is simply the *Hamming weight*.

This thesis, however, focuses on a different class of codes, namely matrix codes: subspaces $\mathcal{C}$ of the vector space $V = \mathbb{F}_q^{m\times n}$ of matrices over a finite field. A more thorough introduction on matrix codes is given by Gorla [198]. The usual choice for measuring the distance between matrices over a finite field is the so-called *rank metric*, defined as follows.

Definition 1. Let $\mathrm{Rank}(\mathbf{M})$ denote the rank of a matrix $\mathbf{M} \in \mathbb{F}_q^{m\times n}$. The *rank distance* between two $m \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$ over $\mathbb{F}_q$ is defined as

$$d(\mathbf{A},\mathbf{B}) = \mathrm{Rank}(\mathbf{A}-\mathbf{B}).$$

As a consequence, the weight function $w(\mathbf{A}) = d(\mathbf{A}, 0)$ is simply the rank of a matrix, and we therefore usually use 'rank' to refer to the 'weight' of codewords in this thesis. By symmetry, without loss of generality, we may assume $n \geqslant m$.

Definition 2. A *matrix code* is a subspace $\mathcal{C}$ of $m \times n$ matrices over $\mathbb{F}_q$ endowed with the rank metric. We let k denote the dimension of $\mathcal{C}$ as a subspace of $\mathbb{F}_q^{m\times n}$ and we denote the basis by $\langle \mathbf{C}_1, \ldots, \mathbf{C}_k \rangle$, with $\mathbf{C}_i \in \mathbb{F}_q^{m\times n}$ linearly independent.

VECTORIZATION. For a matrix $\mathbf{A} \in \mathbb{F}_q^{m\times n}$, let $\mathrm{Vec} : \mathbb{F}_q^{m\times n} \to \mathbb{F}_q^{mn}$ be the mapping that sends a matrix $\mathbf{A}$ to the vector $\mathrm{Vec}(\mathbf{A}) \in \mathbb{F}_q^{mn}$ obtained by 'flattening' $\mathbf{A}$:

$$\mathrm{Vec} : \begin{pmatrix} a_{1,1} & \ldots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \ldots & a_{m,n} \end{pmatrix} \mapsto (a_{1,1}, \ldots, a_{1,n}, \ldots, a_{m,1}, \ldots, a_{m,n}).$$

We denote the inverse operation by Mat, so that $\mathrm{Mat}(\mathrm{Vec}(\mathbf{A})) = \mathbf{A}$. Using the map Vec, an $[m \times n, k]$ matrix code $\mathcal{C}$ can be thought of as an $\mathbb{F}_q$-subspace of $\mathbb{F}_q^{mn}$, and thus we can represent $\mathcal{C}$ by a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times mn}$, similar to what is done for linear codes. Indeed, if $\mathcal{C}$ is an $[m \times n, k]$ matrix code over $\mathbb{F}_q$, we denote its vectorization by $\mathrm{Vec}(\mathcal{C})$:

$$\mathrm{Vec}(\mathcal{C}) := \{\mathrm{Vec}(\mathbf{C}) \colon \mathbf{C} \in \mathcal{C}\}.$$

In this case, $\mathrm{Vec}(\mathcal{C})$ is a k-dimensional $\mathbb{F}_q$-subspace of $\mathbb{F}_q^{mn}$.

§2 Isometries

Definition 3. Let V be a linear space equipped with a metric d and thus a weight w. An *isometry* is a linear map $\mu : V \to V$ that preserves the weight,

$$\mu(w(v)) = w(\mu(v)), \quad \text{for all } v \in V.$$

An isometry of codes $\mu : \mathcal{C} \to \mathcal{D}$ for codes $\mathcal{C}, \mathcal{D} \subseteq V$ preserves the weight of all codewords

$$\mu(w(c)) = w(\mu(c)), \quad \text{for all } c \in \mathcal{C}.$$

Isometries are important in code-based cryptography as they capture a notion of similarity between codes. Take for example any matrix code $\mathcal{C}$ and swap the first two columns of each code word. This results in a code $\mathcal{C}'$ that is not equal

to $\mathcal{C}$, but 'feels' like the same object. Isometries capture this feeling, by defining a notion of 'equivalence' between codes, beyond the general notion of equality.

Definition 4. Two codes $\mathcal{C}, \mathcal{D}$ are *equivalent* if there exists an isometry $\mu : \mathcal{C} \to \mathcal{D}$.

It is natural to ask if isometries of codes $\mu : \mathcal{C} \to \mathcal{C}$ extend to the space V they are contained in. For the Hamming metric, the result is satisfactory: the MacWilliams Extension Theorem [265] shows that any isometry μ of a linear code $\mathcal{C}$ can be extended to an isometry $\mu' : V \to V$ such that $\mu'_{|\mathcal{C}} = \mu$.

For the rank metric, this no longer holds. It is not difficult to construct counterexamples of isometries that do not extend, and recent research has explored the question to what extent such a theorem may hold for the rank metric [199]. In light of such obstructions, we have to admit defeat and only consider isometries of the full matrix space $\mathbb{F}_q^{m\times n}$ in this thesis. Luckily, such isometries are easy to categorize.

Lemma 5. Let $\mu : \mathbb{F}_q^{m\times n} \to \mathbb{F}_q^{m\times n}$ be an isometry with respect to the rank metric. If $m \neq n$, there exist matrices $\mathbf{A} \in \mathrm{GL}_m(q)$, $\mathbf{B} \in \mathrm{GL}_n(q)$ such that

$$\mu(\mathbf{M}) = \mathbf{AMB},$$

for all $\mathbf{M} \in \mathbb{F}_q^{m\times n}$. When $m = n$, we may additionally transpose $\mathbf{M}$,

$$\mu(\mathbf{M}) = \mathbf{AM}^\top\mathbf{B}.$$

Combining Definition 4 with Lemma 5, we will slightly abuse terminology and say that two rank-metric codes $\mathcal{C}, \mathcal{D} \subseteq \mathbb{F}_q^{m\times n}$ are equivalent, if there exist two matrices $\mathbf{A} \in \mathrm{GL}_m(q)$, $\mathbf{B} \in \mathrm{GL}_n(q)$ such that for $\mathbf{C} \in \mathcal{C}$,

$$\mathbf{C} \mapsto \mathbf{ACB}$$

is an isometry $\mathcal{C} \to \mathcal{D}$. This terminology disregards isometries of the codes that do not extend to the full space, isometries of the full space that transpose codewords, and automorphisms of the underlying finite field $\mathbb{F}_q$. However, this seems to have no impact on the cryptographic applications in this thesis. In short, when we write about isometries in the rank metric, we assume isometries in $\mathcal{I}_{m,n}(q) := \mathrm{GL}_m(q) \rtimes \mathrm{GL}_n(q)$.

For any scalars $\lambda, \nu \in \mathbb{F}_q^*$, the isometry $\mu = (\mathbf{A}, \mathbf{B})$ and $\mu' = (\lambda\mathbf{A}, \nu\mathbf{B})$ are essentially equal, for all purposes. We can thus think of isometries as elements $(\mathbf{A}, \mathbf{B}) \in \mathrm{GL}_m(q) \rtimes \mathrm{PGL}_n(q)$. This is conceptually the better interpretation, but

often cumbersome. Thus, we stay with the notation $\mu = (\mathbf{A}, \mathbf{B})$ with $\mathbf{A} \in \mathrm{GL}_m(q)$, $\mathbf{B} \in \mathrm{GL}_n(q)$ and refer to the projective interpretation where needed. The set of isometries between two codes $\mathcal{C}$ and $\mathcal{D}$ is denoted $\mathcal{I}(\mathcal{C}, \mathcal{D}) \subseteq \mathcal{I}_{m,n}(q)$.

ISOMETRIES OF VECTORIZED CODES. The equivalence between two matrix codes can similarly be expressed using the Kronecker product of $\mathbf{A}^\top$ and $\mathbf{B}$, as this behaves well with respect to vectorization. For $\mathbf{C} \in \mathcal{C}$,

$$\mathbf{ACB} = \mathbf{D} \quad \text{if and only if} \quad (\mathbf{B}^\top \otimes \mathbf{A}) \operatorname{Vec}(\mathbf{C}) = \operatorname{Vec}(\mathbf{D}).$$

Thus, between the vectorized codes $\operatorname{Vec}(\mathcal{C}), \operatorname{Vec}(\mathcal{D}) \subseteq \mathbb{F}_q^{mn}$, the isometry becomes $\mathbf{B}^\top \otimes \mathbf{A} \in \mathrm{GL}_{mn}(q)$. For their generator matrices, we get a similar result.

Lemma 6. Let $\mathcal{C}$ and $\mathcal{D}$ be two $[m \times n, k]$ matrix codes over $\mathbb{F}_q$. Suppose that $\mathcal{C}$ and $\mathcal{D}$ are equivalent with $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$, with $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$. If $\mathbf{G}$ and $\mathbf{G}'$ are generator matrices for $\operatorname{Vec}(\mathcal{C})$ and $\operatorname{Vec}(\mathcal{D})$ respectively, then there exists a $\mathbf{T} \in \mathrm{GL}_k(q)$ such that $\mathbf{G}' = \mathbf{TG}(\mathbf{A}^\top \otimes \mathbf{B})$.

It is common to write the generator matrices in *systematic form*, as a matrix of the shape $(\mathbf{I}_k \mid \mathbf{M})$ for some $\mathbf{M} \in \mathbb{F}_q^{k \times (mn-k)}$. We denote this operation $G \mapsto (\mathbf{I}_k \mid \mathbf{M})$ by SF. Following Lemma 6, this gives us that $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$ if and only if $\mathsf{SF}(\mathbf{G}') = \mathsf{SF}(\mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B}))$.

2.1 AUTOMORPHISMS OF MATRIX CODES

Definition 7. An *automorphism* of a code $\mathcal{C}$ is an isometry $\mu : \mathcal{C} \to \mathcal{C}$. The full group of automorphisms is denoted $\mathrm{Aut}^\bullet(\mathcal{C})$. We denote the subgroup of automorphisms for matrix codes derived from $\mu : \mathbb{F}_q^{m \times n} \to \mathbb{F}_q^{m \times n}$ by

$$\mathrm{Aut}(\mathcal{C}) = \mathcal{I}(\mathcal{C}, \mathcal{C}) = \{(\mathbf{A}, \mathbf{B}) \in \mathcal{I}_{m,n}(q) \mid \mathbf{A}\mathcal{C}\mathbf{B} = \mathcal{C}\}.$$

We write $\mathrm{Aut}_L(\mathcal{C})$, resp. $\mathrm{Aut}_R(\mathcal{C})$ to denote the subgroups of $\mathrm{Aut}(\mathcal{C})$ where $\mathbf{B} = \lambda \mathbf{I}_n$, resp. $\mathbf{A} = \lambda \mathbf{I}_m$.

Necessarily, $(\lambda \mathbf{I}_m, \nu \mathbf{I}_n) \in \mathrm{Aut}(\mathcal{C})$ for any code $\mathcal{C}$ and any scalars $\lambda, \nu \in \mathbb{F}_q^*$. If the automorphism groups contains no other elements than these, we say the automorphism group is *trivial*[1].

1 From the projective perspective, this is the trivial subgroup of $\mathrm{GL}_m(q) \rtimes \mathrm{PGL}_m(q)$.

Both $\mathrm{Aut}_L(\mathcal{C})$ and $\mathrm{Aut}_R(\mathcal{C})$ are normal subgroups of $\mathrm{Aut}(\mathcal{C})$. We can thus define the group $\mathrm{Aut}_{LR}(\mathcal{C}) := \mathrm{Aut}(\mathcal{C})/(\mathrm{Aut}_L(\mathcal{C}) \times \mathrm{Aut}_R(\mathcal{C}))$, which measures how many distinct non-trivial two-sided automorphisms a code has.

Lemma 8. Let $\mathcal{C}$ and $\mathcal{D}$ be equivalent codes. Then, for any $\mu \in \mathcal{I}(\mathcal{C}, \mathcal{D})$,

$$\mu \cdot \mathrm{Aut}(\mathcal{C}) = \mathcal{I}(\mathcal{C}, \mathcal{D}) = \mathrm{Aut}(\mathcal{D}) \cdot \mu.$$

Proof. As $\mathcal{C}$ and $\mathcal{D}$ are equivalent, there is some $\mu \in \mathcal{I}(\mathcal{C}, \mathcal{D})$. For any $\mu' \in \mathrm{Aut}(\mathcal{C})$, it is clear that $\mu \cdot \mu' \in \mathcal{I}(\mathcal{C}, \mathcal{D})$. For any other $\mu' \in \mathcal{I}(\mathcal{C}, \mathcal{D})$, we get $\mu' = \mu \cdot \mu^{-1} \cdot \mu'$ with $\mu^{-1} \cdot \mu' \in \mathrm{Aut}(\mathcal{C})$. □

As a consquence, for any two codes $\mathcal{C}$ and $\mathcal{D}$ with $\mathcal{I}(\mathcal{C}, \mathcal{D})$ non-empty, $\mathcal{I}(\mathcal{C}, \mathcal{D})$ is as large as $\mathrm{Aut}(\mathcal{C})$, which is therefore as large as $\mathrm{Aut}(\mathcal{D})$.

§3 Matrix Code Equivalence

Finally, we discuss the *Matrix Code Equivalence problem* (MCE). Part 3 focuses on cryptography based on isometries, whose security relies on the hardness of MCE.

Problem 1 (Matrix Code Equivalence)**.** Given two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subseteq \mathbb{F}_q^{m \times n}$, find, if it exists, a map $(\mathbf{A}, \mathbf{B}) \in \mathcal{I}_{m,n}(q)$ such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mathbf{ACB} \in \mathcal{D}$.

We will refer to Matrix Code Equivalence by MCE. Its hardness has been studied previously [38, 138], and we extend this research in Chapter XII.

Remark 1. An MCE instance of dimension k with $m \times n$ matrices over $\mathbb{F}_q$ can be viewed as a 3-tensor problem, which is symmetrical in its arguments k, m and n. This means that it is equivalent to an MCE instance of dimension m with $k \times n$ matrices and to an MCE instance of dimension n with $k \times m$ matrices. Switching to equivalent instances is a matter of changing perspective on the $k \times m \times n$ object over $\mathbb{F}_q$ defined by $A_{ijl} = A_{ij}^{(l)}$. In other words, each basis matrix $m \times n$ defines a slice of a cube, and one can take different slices for isomorphic instances.

3.1 Group Action from Isometries

The group of isometries $\mathcal{I}_{m,n}(q)$ acts on the set X of k-dimensional matrix codes by simply mapping any code $\mathcal{C} \in X$ to $\mathbf{A}\mathcal{C}\mathbf{B} \in X$ for an isometry $(\mathbf{A}, \mathbf{B})$. By

abuse of terminology, we also refer to this group action by MCE. This is justified because MCE is precisely the vectorisation problem from Definition I.8 for this group action.

With respect to the properties given in Section I.2.1, we find that this group action is neither commutative, nor transitive, nor free. However, MCE is *efficient*, and so it can be used to construct a Sigma protocol. The non-commutativity is both positive and negative: although it limits the cryptographical design possibilities, e.g. key exchange becomes hard, it prevents quantum attacks to which commutative cryptographic group actions are vulnerable, such as Kuperberg's algorithm for the dihedral hidden subgroup problem [248].

With regards to efficiency, it is immediate to notice that MCE is promising, given that all of the operations in the proposed protocols are simple linear algebra; this is in contrast with code-based literature (where complex decoding algorithms are usually required) and other group actions (e.g. isogeny-based) which are burdened by computationally heavy operations.

After applying the Fiat-Shamir transform, we obtain a signature scheme called *Matrix Equivalence Digital Signature* (MEDS). This signature scheme, and its efficiency, is studied in Chapter XIII.

More generally, we can apply a range of techniques to improve the performance of such Sigma protocols. We give a unified taxonomy of such techniques, both well-known and novel, in Chapter XIV and apply this framework to a number of schemes, including MEDS, to analyze optimal scheme parameters.

Part 1

ISOGENIES (CSIDH)

THE LANDSCAPE OF CSIDH

That is... somehow... quite cool...

Dr. F. Kamp-Horst

At the start of this thesis, CSIDH [96] was only recently introduced, but quickly became the main topic of research in isogeny-based cryptography, as it is the only post-quantum cryptographic group action. The quantum security was hotly debated [49, 60, 102, 307], constant-time techniques were constantly improving [98, 274, 276, 301], new mathematical tools were developed to increase performance [44, 88, 92, 107, 213], and the maturity of the scheme had already reached the stage of side-channel analysis [78, 255]. Beyond the NIKE, the CSIDH group action was used for signature schemes [54, 144, 155], threshold schemes [150], oblivious transfer [250], and more. This part of the thesis focuses on the security and efficiency of the CSIDH-NIKE, divided into two subparts.

In the first subpart, we focus on the *physical security* of CSIDH.

- In Chapter IV, we show a *passive attack*, using side-channel analysis, to exploit zero-value and correlation information, which allows us to detect if a secret isogeny walk passes over the zero curve E_0. This allows us to completely recover the secret key used in CSIDH, and its variants, in unprotected implementations. As a side-step, we show how the same technique can be applied to SIDH/SIKE [145, 221] to recover the secret key. Although SIDH/SIKE is famously broken [86, 266, 329], before this break this side-channel attack was a serious threat to the practical security of SIKE.
- In Chapter V, we show an *active attack*, using fault-injections, to flip a single bit and cause a *disorienation fault* in CSIDH, changing the directions of some of the steps of a secret isogeny walk. This introduces a new family of fault attacks on CSIDH, and we show that about 2^8 disorientation faults are enough to recover the secret key in unprotected CSIDH implementations. We provide lightweight contermeasures against this attack and discuss their security.

In the second subpart, we focus on the *efficiency and optimisations* of the class group action evaluation.

- In Chapter VI, we challenge the effectiveness of radical isogenies [92] to improve the class group action evaluation. The analysis of constant-time radical isogenies was left as an open problem by [92] and we show that neither a straightforward nor an optimized implementation of radical isogenies achieves significant improvements. To show this, we develop projective variants of radical isogenies and a hybrid strategy combining radical isogenies with traditional 'Vélu' isogenies and benchmark the results in a state-of-the-art implementation. We conclude that radical isogenies seem unfit for constant-time implementations of the CSIDH class group action.

- In Chapter VII, we take a closer look at the state of the art for CSIDH and ask ourselves if CSIDH is (close to) ready for real-world applications, five years after its initial publication. We take into account the developments of Chapters IV to VI and other recent developments [20] to develop a highly-optimized fully-deterministic dmmy-free CSIDH implementation with conservative estimates of quantum security and benchmark this implementation in TLS as a real-world use case. Although we achieve significant speed ups compared to previous CSIDH implementations and smaller communication requirements compared to other post-quantum KEMs and signature schemes, we nevertheless obtain latencies that are prohibitive for usage in the real world. This reduces the practicality of CSIDH as a NIKE to niche cases, when following the most conservative quantum security arguments.

- In Chapter VIII, we solve some of the issues raised in Chapter VII using pairing-based techniques. Although pairings have been used previously in isogeny-based cryptography, their applicability was limited to curves over fields with primes of a specific shape as pairings seemed too costly for the type of primes that are often used in isogeny-based cryptography. For example, their effectiveness for CSIDH had so far not been analysed. This chapter fixes this gap and optimises computations of the Tate pairing for CSIDH and other isogeny-based protocols. We furthermore present new algorithms using pairings, such as supersingularity verification and torsion basis generation, which can be used more generally in isogeny-based cryptography.

IV

PATIENT ZERO & PATIENT SIX

Recent works have started side-channel analysis on SIKE and show the vulnerability of isogeny-based systems to zero-value attacks. In this chapter, we expand on such attacks by analyzing the behavior of the zero curve E_0 and six curve E_6 in CSIDH and SIKE. We demonstrate an attack on static-key CSIDH and SIKE implementations that recovers bits of the secret key by observing via zero-value-based resp. exploiting correlation-collision-based side-channel analysis whether secret isogeny walks pass over the zero or six curve. We apply this attack to fully recover secret keys of SIKE and two state-of-the-art CSIDH-based implementations: CTIDH and SQALE. We show the feasibility of exploiting side-channel information for the proposed attacks based on simulations with various realistic noise levels. Additionally, we discuss countermeasures to prevent zero-value and correlation-collision attacks against CSIDH and SIKE in our attacker model.

This chapter is based on the paper

> Fabio Campos, Michael Meyer, Krijn Reijnders, and Marc Stöttinger. "Patient Zero & Patient Six: Zero-Value and Correlation Attacks on CSIDH and SIKE". in: *International Conference on Selected Areas in Cryptography*. Springer. 2022, pp. 234–262.

In terms of content, I have removed the appendices and incorporated these into the text where possible to provide a more streamlined reading experience. Beyond this, I have mostly made only minor editorial changes that ensure consistency with the rest of the chapters. At the time of writing this thesis, SIDH/SIKE is fully broken [86, 266, 329]. I have nevertheless left the content as is, to reflect the understanding at the time of publishing of the paper.

§1 Introduction

Isogeny-based cryptography is a promising candidate for replacing pre-quantum schemes with practical quantum-resistant alternatives. In general, isogeny-based schemes feature very small key sizes, while suffering from running times that are at least an order of magnitude slower than e.g. lattice- or code-based schemes. Therefore, they present a viable option for applications that prioritize bandwidth over performance. SIKE [220], a key encapsulation mechanism (KEM) based on the key exchange SIDH [223], is the lone isogeny-based participant of the NIST post-quantum cryptography standardization process, and proceeded to the fourth round. In 2018, only after the NIST standardization process started, the key exchange scheme CSIDH was published [96]. Due to its commutative structure, a unique feature among the known post-quantum schemes, CSIDH allows for a non-interactive key exchange, which gained much attention among the research community. Together with its efficient key validation, which enables a static-static key setting, this makes CSIDH a promising candidate for a drop-in replacement of classical Diffie–Hellman-style schemes.

In this work, we focus on a side-channel attack against CSIDH and SIKE. We follow the main idea of [151], which reconstructs SIKE private keys through *zero-value* attacks. This attack approach tries to force zero values for some intermediate values of computations related to secret key bits. By recognizing these zero values via side-channel analysis (SCA) an attacker can recover bits of the secret key. While *coordinate randomization* is an effective method to mitigate general *Differential Power Analysis* (DPA) and *Correlation Power Analysis* (CPA) and is incorporated in SIKE [220], it has no effect on zero values, so that forcing zeros bypasses this countermeasure, Similar to [151], the recent *Hertzbleed attack* exploits such zero values in SIKE [376].

While [151] focuses on forcing values connected to elliptic curve points to be zero, we discuss the occurrence of zero values as curve parameters. This was first proposed in [244], yet [151] concludes that this idea is unlikely to be applicable in a realistic scenario, since curve representations in SIKE cannot produce a zero. In spite of this fact, we show that some curves in SIKE and CSIDH, for example the zero curve and the six curve, have a special correlation in their representations, so that we can observe these via side-channel analysis.

The secret isogeny computation in SIKE essentially consists of two phases: scalar multiplication and isogeny computation. In general, the first phase is believed to be more vulnerable to physical attacks, since private-key bits are directly used there (see [127]). We propose the first passive implementation attack using side-channel analysis that exclusively targets the second phase of

the SIKE isogeny computation. Notably, countermeasures like coordinate or coefficient randomization [127] or the *CLN test* [132, 151] do not prevent this attack.

Contributions

In this chapter, we present zero-value and correlation attacks against state-of-the-art implementations of CSIDH and SIKE. For CSIDH, we use the fact that the zero curve E_0, i.e., the Montgomery curve with coefficient $A = 0$, represents a valid curve. Thus, whenever a secret isogeny walk passes over this curve, this can be detected via side-channel analysis. We present a passive adaptive attack that recovers one bit of the secret key per round by forcing the target to walk over the zero curve.

Some implementations, like SQALE and SIKE, represent the zero curve without using zero values. Nevertheless, in such a case there is often (with probability $1/2$ in SQALE and probability 1 in SIKE) a strong correlation between certain variables, which also occurs for the supersingular six curve E_6 with coefficient $A = 6$. Via CPA, we exploit this correlation to detect these curves, and mount a similar adaptive attack.

Using these two approaches, we present a generic attack framework, and apply this attack to the state-of-the-art CSIDH implementations SQALE [102] and CTIDH [20] in Section 3, and to SIKE in Section 4. We explore the practical feasibility of the proposed attacks in Section 5, and show the results of our simulations in Section 6. Finally, in Section 7, we discuss different types of countermeasures. Our code is available in the public domain:

https://github.com/PaZeZeVaAt/simulation

Related work.

The analysis of physical attacks on isogeny-based schemes has only recently gained more attention, including both side-channel [151, 194, 244, 376, 378] and fault attacks [2, 78, 79, 193, 255, 363, 366]. Introduced for classical elliptic curve cryptography (ECC) in [6, 200, 218], zero-value attacks were adapted to SIKE in [151], which applies t-tests to determine zero values within power traces [336].

An approach to identify certain structures within traces, similar to the ones occurring in non-zero representations of the zero curve and six curve in our case, are correlation-enhanced power-analysis collision attacks [283], such as [35] for

ECC. This attack combines the concept of horizontal side-channel analysis [290] with correlation-enhanced power-analysis collision attacks to extract leakage from a single trace.

From a constructive perspective, our attacks follows the idea of steering isogeny paths over special curves, as proposed for the zero curve in [244]. Furthermore, the attack on SIKE uses the framework of [2] to produce suitable public keys. However, our attack is a *passive* attack that is much easier to perform in practice compared to the elaborate fault injection required for [2].

§2 Preliminaries

Chapter II gives an introduction to curves and their isogenies, and a thorough introduction to CSIDH, and its variants SQALE and CTIDH. Here, we only introduce finer arithmetical details required for our attacks.

Recall that we only work with curves in Montgomery form (Definition II.2) and that the public key in CSIDH is the supersingular curve E_A which corresponds to the to class group action of the secret key $\mathfrak{a}$ applied to the publicly known starting curve $E_0 : y^2 = x^3 + x$:

$$E_A = \mathfrak{a} * E_0 = \mathfrak{l}_1^{e_1} * \cdots * \mathfrak{l}_n^{e_n} * E_0. \tag{1}$$

2.1 Representation of Montgomery coefficient

To decrease computational cost by avoiding costly inversions, the curve E_A is almost always represented using *projective* coordinates. The following two are used most in current CSIDH-based implementations:

- the Montgomery form $(A : C)$, such that the curve coefficient is A/C, with C non-zero,
- and the alternative Montgomery form $(A + 2C : 4C)$, such that the curve coefficient is A/C, with C non-zero.

The alternative Montgomery form is most common, as it is used in projective scalar point-multiplication formulas. Hence, in most state-of-the-art implementations of CSIDH-based systems, the Montgomery coefficient is mapped to alternative Montgomery form and remains in this form until the end, where it is mapped back to affine form for the public key resp. shared secret (e.g., in SQALE [102]). CTIDH [20] switches between both representations after each isogeny, and maps back to affine form at the end. For most values of $(A : C)$

and $(A + 2C : 4C)$, the associated curve is either ordinary or supersingular. The exceptions are $C = 0$, which represents no algebraic object, and $A = \pm 2C$, which represents the singular curves $E_{\pm 2}$. Specifically the supersingular zero curve E_0 is represented as $(0 : C)$ in Montgomery form and $(2C : 4C)$ in alternative Montgomery form, where $C \in \mathbb{F}_p$ can be any non-zero value.

2.2 ISOGENY COMPUTATION IN PROJECTIVE FORM

When using projective representations to compute isogenies for a curve E with Montgomery coefficient $(A : C)$, most implementations use projectivized versions of Vélu's formulas, described in [44, 282, 373]. To compute the action of $\mathfrak{l}_i^{\pm 1}$ on E, one finds a point P of order ℓ_i on E and computes the x-coordinates of the points $\{P, [2]P, \ldots, [\frac{\ell_i - 1}{2}]P\}$. Let $(X_k : Z_k)$ denote the x-coordinate of $[k]P$ in projective form. Then, the projective Montgomery coefficient $(A' : C')$ of $E' = \mathfrak{l}_i * E$ using Montgomery form $(A : C)$ is computed by

$$B_z = \prod_{k=1}^{\frac{\ell_i - 1}{2}} Z_k, \quad A' = (A + 2C)^{\ell_i} \cdot B_z^8, \tag{2}$$

$$B_x = \prod_{k=1}^{\frac{\ell_i - 1}{2}} X_k, \quad C' = (A - 2C)^{\ell_i} \cdot B_x^8. \tag{3}$$

When using alternative Montgomery form $(\alpha : \beta) = (A + 2C : 4C)$, we use similar formulas

$$B_z = \prod_{k=1}^{\frac{\ell_i - 1}{2}} Z_k, \quad \alpha' = \alpha^{\ell_i} \cdot B_z^8, \tag{4}$$

$$B_x = \prod_{k=1}^{\frac{\ell_i - 1}{2}} X_k, \quad \beta' = \alpha' - (\alpha - \beta)^{\ell_i} \cdot B_x^8, \tag{5}$$

where $(\alpha' : \beta')$ represents E' in alternative Montgomery form. Note that the values $(A + 2C)$ in (2), $(A - 2C)$ in (3), α in (4) and $(\alpha - \beta)$ in (5) are never zero: In all cases, this implies $A/C = \pm 2$, i.e., the singular curves $E_{\pm 2}$.

Remark 1. So far, we know of no deterministic implementations based on the class group action[1]. This is because in order to perform the isogenies, all current implementations sample a random point P on the curve and compute the scalar

1 A determinstcit implementation is presented in Chapter VII, inspired by this work.

multiple of P required to perform isogenies. The projective coordinates $(X_k : Z_k)$ are then non-deterministic, and hence the output of Equations (2) to (5) is non-deterministic. This implies that the representations $(A : C)$ or $(A + 2C : 4C)$ is non-deterministic after the first isogeny. A deterministic approach, e.g. as sketched in [49] using Elligator, ensures a deterministic representation of the curve coefficient but has so far not been put into practice.

2.3 SIKE

In SIKE, we pick a prime of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$ such that $2^{e_A} \approx 3^{e_B}$, and work with supersingular elliptic curves over $\mathbb{F}_{p^2}$ in Montgomery form. We choose to work with curves such that $E(\mathbb{F}_{p^2}) = (p+1)^2$, and we have $E(\mathbb{F}_{p^2}) \xrightarrow{\sim} \mathbb{Z}^2_{2^{e_A}} \times \mathbb{Z}^2_{3^{e_B}}$ for these curves. Thus, the full 2^{e_A}- and 3^{e_B}-torsion subgroups lie in $E(\mathbb{F}_{p^2})$. Any point R_A of order 2^{e_A} then uniquely (up to isomorphism) determines a 2^{e_A}-isogeny and codomain curve $E' = E/\langle R_A \rangle$ with kernel $\langle R_A \rangle$. For choosing an appropriate point, the SIKE setup defines basis points P_A and Q_A of the 2^{e_A}-torsion of the public starting curve. Picking an integer $\mathsf{sk_A} \in [0, 2^{e_A} - 1]$ and computing $R_A = P_A + [\mathsf{sk_A}]Q_A$ then results in choosing such a kernel generator R_A of order 2^{e_A}.

In practice, such a 2^{e_A}-isogeny is computed as a sequence of 2-isogenies of length e_A. This can be interpreted as a sequence of steps through a graph: For a prime ℓ with $\ell \neq p$, the ℓ-isogeny graph consists of vertices that represent (j-invariants of) elliptic curves, and edges representing ℓ-isogenies. Due to the existence of dual isogenies, edges are undirected. For supersingular curves, this graph is an $(\ell+1)$-regular expander graph and contains approximately $p/12$ vertices. Hence, a sequence of 2-isogenies of length e_A corresponds to a walk of length e_A through the 2-isogeny graph. An analogous discussion applies to the case of 3^{e_B}-isogenies. For reasons of efficiency, we often combine two 2-isogeny steps into one 4-isogeny.

The secret keys $\mathsf{sk_A}$, $\mathsf{sk_B}$ can be decomposed as

$$\mathsf{sk_A} = \sum_{i=0}^{e_2-1} \mathsf{sk}_i \cdot 2^i \quad \mathsf{sk}_i \in \{0,1\}, \qquad \mathsf{sk_B} = \sum_{i=0}^{e_3-1} \mathsf{sk}_i \cdot 3^i \quad \mathsf{sk}_i \in \{0,1,2\}.$$

We refer to these sk_i as the *bits* resp. the *trits* of the secret key $\mathsf{sk_A}$ resp. $\mathsf{sk_B}$. For a given sk, we use $\mathsf{sk}_{<k}$ to represent the key up to the k-th bit/trit.

The SIKE scheme.

The main idea behind SIDH and SIKE is to use secret isogenies to set up a key exchange scheme resp. key encapsulation mechanism. SIDH fixes E_6 as starting curve, and torsion basis points P_A, Q_A and P_B, Q_B. It uses the following subroutines:

- Keygen_A samples a secret key $\mathsf{sk_A} \in [0, 2^{e_A} - 1]$, computes $R_A = P_A + [\mathsf{sk_A}]Q_A$, and the secret isogeny $\varphi_A : E_6 \to E_6/\langle R_A \rangle$. It outputs the key pair $(\mathsf{sk_A}, \mathsf{pk_A})$, where $\mathsf{pk_A} = (\varphi_A(P_B), \varphi_A(Q_B), \varphi_A(Q_B - P_B))$. We write $\mathsf{Keygen}_A(\mathsf{sk})$ if Keygen_A does not sample a secret key, but gets sk as input.
- Keygen_B proceeds analogously with swapped indices A and B. The public key is $\mathsf{pk_B} = (\varphi_B(P_A), \varphi_B(Q_A), \varphi_B(Q_A - P_A))$.
- Derive_A takes as input $(\mathsf{sk_A}, \mathsf{pk_B}) = (S_A, T_A, T_A - S_A)$. It computes the starting curve E_B from the points in $\mathsf{pk_B}$, the secret point $R'_A = S_A + [\mathsf{sk_A}]T_A$, and the isogeny $\varphi'_A : E_B \to E_B/\langle R'_A \rangle$.
- Derive_B proceeds analogously with input $(\mathsf{sk_B}, \mathsf{pk_A})$, and computes the codomain curve $E_A/\langle R'_B \rangle$.

When running this key exchange, both parties arrive at a curve (isomorphic to) $E_6/\langle R_A, R_B \rangle$, and (a hash of) its j-variant can serve as a shared secret. SIKE uses the SIDH subroutines Keygen and Derive to construct three algorithms Keygen, Encaps, and Decaps. Furthermore, we define H and H' to be cryptographic hash functions.

- Keygen generates a (static) key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Keygen}_B$.
- Encaps encapsulates a random value m in the following way:
  - Get an ephemeral key pair $(\mathsf{ek}, c) \leftarrow \mathsf{Keygen}_A(\mathsf{ek})$ with $\mathsf{ek} = \mathsf{H}(\mathsf{pk}, m)$.
  - Compute the shared secret $s \leftarrow \mathsf{Derive}_A(\mathsf{ek}, \mathsf{pk})$.
  - Compute the ciphertext $\mathsf{ct} = (c, \mathsf{H}'(s) \oplus m)$.
- Decaps receives a ciphertext (c_0, c_1), and proceeds as follows:
  - Compute the shared secret $s' \leftarrow \mathsf{Derive}_B(\mathsf{sk}, c_0)$.
  - Recover $m' \leftarrow c_1 \oplus \mathsf{H}'(s')$.
  - Recompute $\mathsf{ek}' = \mathsf{H}(\mathsf{pk}, m')$.
  - Compute $(\mathsf{ek}', c') \leftarrow \mathsf{Keygen}_A(\mathsf{ek}')$ and check if $c' = c_0$.

  Passing this check guarantees that the ciphertext has been generated honestly, and $m' = m$ can be used to set up a session key.

Representation of Montgomery coefficients.

As in CSIDH, the curve E is almost always represented using projective coordinates, with the caveat that coefficents now live in $\mathbb{F}_{p^2}$. The following two representations are used throughout SIKE computations, although in different subroutines.

- The alternative Montgomery form $(A + 2C : 4C)$, such that the coefficient is equal to A/C with C non-zero. This representation is used for Alice's computations as it is the most efficient for computing 2- or 4-isogenies. It is often written as $(A_{24}^+ : C_{24})$ with $A_{24}^+ = A + 2C$ and $C_{24} = 4C$.
- The form $(A + 2C : A - 2C)$, such that the coefficient is equal to A/C, with C non-zero. This representation is used for Bob's computations as it is the most efficient for computing 3-isogenies. It is often written as $(A_{24}^+ : A_{24}^-)$ with $A_{24}^+ = A + 2C$ and $A_{24}^- = A - 2C$

Note that the values A, C, A_{24}^+, A_{24}^- and C_{24} are in $\mathbb{F}_{p^2}$. When necessary, we write them as $\alpha + \beta i$ with $\alpha, \beta \in \mathbb{F}_p$ and $i^2 = -1$. Equal to CSIDH, both forms represent either an ordinary or a supersingular curve, with the exceptions $C = 0$, which represents no algebraic object, and $A = \pm 2C$, which represents the singular curves $E_{\pm 2}$. For the rest of the paper, we are interested in representations of the supersingular six curve E_6. Fortunately, E_6 must have $A = 6C$ and so in *both* forms is represented as $(8C : 4C)$, with $C = \alpha + \beta i \in \mathbb{F}_{p^2}$ any non-zero element. For the goal of the paper, this means that the analysis is similar for both forms.

Isogeny computation in projective form.

SIKE uses the above projective representations to compute the codomain $E_{\tilde{a}}$ of a 3- or 4-isogeny $\varphi : E_a \to E_{\tilde{a}}$.

4-ISOGENY. Given a point P of order 4 on E with x-coordinate $x(P) = (X : Z)$, the codomain $E' = E/\langle P \rangle$, represented by $(\tilde{A}_{24}^+ : \tilde{C}_{24})$, is computed by

$$\tilde{A}_{24}^+ = 4 \cdot X^4, \qquad \tilde{C}_{24} = 4 \cdot Z^4. \tag{6}$$

3-ISOGENY. Given a point P of order 3 on E with x-coordinate $x(P) = (X : Z)$, the codomain $E' = E/\langle P \rangle$, represented by $(\tilde{A}_{24}^+ : \tilde{A}_{24}^-)$, is computed by

$$\tilde{A}_{24}^+ = (3X + Z)^3 \cdot (X - Z), \quad \tilde{A}_{24}^- = (3X - Z)^3 \cdot (X + Z). \tag{7}$$

§3 Recovering CSIDH Keys with Side-Channel Leakage

In this section, we explore how side-channel information can leak information on secret isogeny walks. As shown in [151], it is possible to detect zero values in isogeny computations using side-channel information. In Section 3.1, we explore how both representations of the zero curve E_0, i.e. $(0 : C)$ and $(2C : 4C)$, leak secret information, even though the value $C \in \mathbb{F}_p$ is assumed to be a uniformly random non-zero value. As E_0 is always a valid and isogenous supersingular $\mathbb{F}_p$-curve in CSIDH, we can always construct a walk that potentially passes over E_0 from a valid CSIDH public key. This allows us to describe a generic approach to leak a given bit of information of the secret isogeny walk, hence, a general attack on the class group action as introduced in CSIDH.

We apply this attack in more detail to the two current state-of-the-art cryptosystems based on this class group action: SQALE in Section 3.2 and CTIDH in Section 3.3. We discuss the practical feasibility of these attacks in Section 5 and simulate these attacks in Section 6. We note that the proposed general attack applies to all variants of CSIDH that we know of, e.g. also [96, 98].

Throughout this work, we assume a static-key setting, i.e., that a long-term secret key $\mathfrak{a}$ is used, and that the attacker can repeatedly trigger key-exchange executions on the target device using public-key curves of their choice. Formally, this means that we adaptively feed curves E_{PK} and get side-channel information on the computations $\mathfrak{a} * E_{\mathrm{PK}}$. We exploit this information to reveal $\mathfrak{a}$ bit by bit.

3.1 Discovering a Bit of the Secret Walk

Detecting E_0 in Montgomery form.

As described in Remark 1, the representation of the Montgomery coefficient as $(A : C)$ or $(A + 2C : 4C)$ is non-deterministic after the first isogeny, so we assume they contain random $\mathbb{F}_p$-values, representing an affine Montgomery coefficient of some curve E. This makes it hard to get any information on E using side channels. However, in Montgomery form the curve E_0 is special: It is simply represented by $(0 : C)$ for some $C \in \mathbb{F}_p$. We define such a representation containing a zero as a *zero-value representation*.

Definition 1. Let E be an elliptic curve over $\mathbb{F}_p$. A *zero-value representation* is a representation of the Montgomery coefficient in projective coordinates $(\alpha : \beta)$ such that either $\alpha = 0$ or $\beta = 0$.

A representation of E_0 in Montgomery form must be a zero-value representation. As is known for ECC and SIKE, an attacker can observe zero-value representations in several different ways using side-channel analysis [151]. We will expand on this in Section 5 to show that E_0 leaks secret information in implementations that use Montgomery form.

Detecting E_0 in alternative Montgomery form.

Using the alternative Montgomery form, no non-singular curve has a zero-value representation, as $(A + 2C : 4C)$ can only be zero for $A = -2C$ corresponding to the singular curve E_{-2}. Thus, the alternative Montgomery form avoids the side-channel attack described above. Nevertheless, the representation of E_0 is remarkable: Whenever $2C$ is smaller than $p/2$, doubling $2C$ does not require a modular reduction, and hence the bit representation of $4C$ is precisely a bit shift of $2C$ by one bit to the left. Such strongly correlated values can be observed using side-channel analysis, as we detail later in Section 5.

Definition 2. Let E be an elliptic curve over $\mathbb{F}_p$. A *strongly-correlated representation* is a representation of the Montgomery coefficient in projective coordinates $(\alpha : \beta)$ such that the bit representations of α and β are bit shifts.[2]

For E_0, for any non-zero value C with $2C \leq p/2$, the representation in alternative Montgomery form by $(2C : 4C)$ is a strongly-correlated representation. As C is effectively random during the computation of the class group action, we assume that the representation of E_0 is strongly correlated half the time. For random cures E, as representation $(A + 2C : 4C)$ seems indistinguishable from random $(\gamma : \delta)$, and so an attacker can differentiate E_0 from such curves. From this, an attacker only needs a few traces to determine accurately whether a walk passes over E_0 or not, as discussed in Section 5.

Remark 2. Other curves have strongly-correlated representations too, e.g., the curve E_6 requires $A = 6C$ which gives $(8C : 4C)$ with $C \in \mathbb{F}_p$ random and non-zero, and so E_6 can be detected in precisely the same way as E_0. For simplicity, we focus on the zero curve in the CSIDH attack. We note that analyzing this attack to any curve with strongly-correlated representations is of independent interest for CSIDH and other isogeny-based schemes, such as SIKE.

Remark 3. In the case where $2C$ is larger than $p/2$, the modular reduction by p decreases the correlation between $2C$ and $4C$ significantly, which is why

2 This definition may be expanded to cover other types of correlation, whenever such correlation can be distinguished from random values using side-channel information.

we disregard these cases. However, a modular reduction does not affect all bits, and so this correlation remains for unaffected bits. Especially for primes with large cofactor 2^k in $p+1$, or primes close to a power of 2, the correlation between unaffected bits should be exploitable. For the primes used in the CSIDH instances in this work, this effect is negligible. However, the primes used in SIDH and SIKE do have this form and we exploit this in Section 4.

Thus, the idea is to detect E_0 in a certain step k of the computation $\mathfrak{a} * E_{\mathtt{PK}}$. In order to ensure that this happens precisely in step k, the computation needs to be performed in a known order of isogeny steps $E \leftarrow \mathfrak{l}^{(k)} * E$. In general, due to how isogenies are computed, such a step can fail with a certain probability. The following definition takes this into account.

Definition 3. Let the ideal $\mathfrak{a}$ be interpreted as a secret isogeny walk. An *ordered evaluation* of $\mathfrak{a} * E$ is an evaluation in a fixed order

$$\mathfrak{l}^{(n)} * \ldots * \mathfrak{l}^{(1)} * E$$

of n steps, assuming that no step fails. We write $\mathfrak{a}_k * E$ for the first k steps of of such an evaluation,

$$\mathfrak{l}^{(k)} * \ldots * \mathfrak{l}^{(1)} * E.$$

We define $p_{\mathfrak{a}}$ resp. $p_{\mathfrak{a}_k}$ as the probability that $\mathfrak{a}$ resp. $\mathfrak{a}_k$ is evaluated without failed steps.

Generic approach to discover isogeny walks using E_0.

Given the ability to detect E_0 in a walk for both the Montgomery form and the alternative Montgomery form, we sketch the following approach to discover bits of a secret isogeny walk $\mathfrak{a}$ that has an ordered evaluation. Assuming we know the first $k-1$ steps $\mathfrak{l}^{(k-1)} * \ldots * \mathfrak{l}^{(1)}$ in the secret isogeny walk $\mathfrak{a}$, denoted by $\mathfrak{a}_{k-1}$, we want to see if the k-th step $\mathfrak{l}^{(k)}$ equals $\mathfrak{l}_i$ or $\mathfrak{l}_i^{-1}$ for some i. We compute $E = \mathfrak{l}_i^{-1} * E_0$ and $E' = \mathfrak{l}_i^{-1} * E$, and as a public key we use $E_{\mathtt{PK}} = \mathfrak{a}_{k-1}^{-1} * E$. Then, when applying the secret walk $\mathfrak{a}$ to $E_{\mathtt{PK}}$, the k-th step either goes over E_0 or over E'. From side-channel information, we thus observe if the k-th step applies $\mathfrak{l}_i^e = \mathfrak{l}_i^1$ or $\mathfrak{l}_i^e = \mathfrak{l}_i^{-1}$, and set $\mathfrak{l}^{(k)} = \mathfrak{l}_i^e$, as shown in Figure 8. We then repeat with $\mathfrak{a}_k = \mathfrak{l}_i^e \cdot \mathfrak{a}_{k-1}$.

If E_0 is not detected in the above setting, i.e. $e = -1$, we can confirm this by an additional measurement: We compute $\tilde{E} = \mathfrak{l} * E_0$ and $\tilde{E}' = \mathfrak{l} * \tilde{E}$, and use $\tilde{E}_{\mathtt{PK}} = \mathfrak{a}_{k-1}^{-1} * \tilde{E}$ as public key. If $e = -1$, the isogeny walk now passes over E_0, which can be recognized via side-channel analysis. More formally, we get:

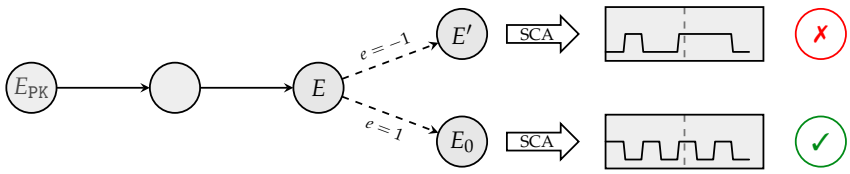


Figure 8: Generic approach to learn secret bits using side-channel information.

Lemma 4. Let $\mathfrak{a}$ be any isogeny walk of the form $\mathfrak{a} = \prod \mathfrak{l}_i^{e_i}$. Assume the evaluation of $\mathfrak{a}$ is an ordered evaluation. Then, there exists a supersingular curve E_{PK} over $\mathbb{F}_p$ such that $\mathfrak{a} * E_{\mathrm{PK}}$ passes over E_0 in the k-th step.

After successfully detecting all steps $\mathfrak{l}^{(k)}$, the private key elements e_i can simply be recovered by counting how often $\mathfrak{l}_i$ resp. $\mathfrak{l}_i^{-1}$ appeared in the evaluation.

This generic approach has a nice advantage: If one detects the k-th step to walk over E_0, this confirms all previous steps were guessed correctly. In other words, guessing wrongly in a certain step will be noticed in the next step: Denote a wrong guess by $\mathfrak{a}_k^{\text{wrong}} = \mathfrak{l}^{-e} \cdot \mathfrak{a}_{k-1}$. The attacker computes E from E_0 so that $\mathfrak{l}' * E = E_0$ and gives the target E_{PK} such that $\mathfrak{a}_k^{\text{wrong}} * E_{\mathrm{PK}} = E$. Due to the wrong guess, neither $e = 1$ nor $e = -1$ lead to E_0, as the *actual* secret walk $\mathfrak{a}$ leads to $E' = \mathfrak{a}_k * E_{\mathrm{PK}}$, and the case $e = 1$ leads to $E_{\neg 0} = \mathfrak{l}' * E' = \mathfrak{l}^{-2e} * E_0$, as shown in Figure 9.

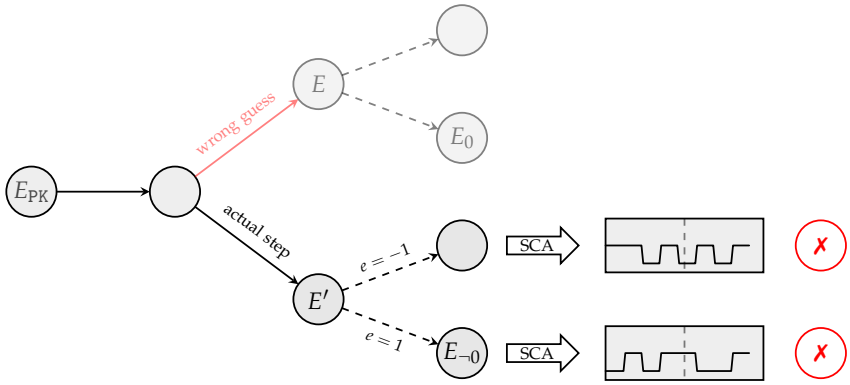


Figure 9: Due to a wrong guess of the isogeny path $\mathfrak{a}_k$, an attacker miscomputes E_{PK} and the actual walk does not pass over E_0.

Remark 4. The curve E_{PK} as given by Lemma 4 is a valid CSIDH public key, so public key validation [96] does not prevent this attack.

Probability of a walk passing over E_0.

Due to the probabilistic nature of the computation of the class group action, not every evaluation $\mathfrak{a} * E_{\mathrm{PK}}$ passes over E_0 in the k-th step: One of the steps $\mathfrak{l}^{(j)}$ for $1 \leq j \leq k-1$ can fail with probability $1/\ell^{(j)}$, and if so, the k-th step passes over a different curve. With E_{PK} as given by Lemma 4, the probability that an ordered evaluation $\mathfrak{a} * E_{\mathrm{PK}}$ passes over E_0 is then described by $p_{\mathfrak{a}_k}$, which we compute in Lemma 5.

Lemma 5. Let $\mathfrak{a}$ be an isogeny walk computed as an ordered evaluation $\mathfrak{l}^{(n)} * \ldots * \mathfrak{l}^{(1)} * E_{\mathrm{PK}}$. Then $p_{\mathfrak{a}_k}$, the probability that the first k isogenies succeed, is

$$p_{\mathfrak{a}_k} := \prod_{j=1}^{k} \frac{\ell^{(j)} - 1}{\ell^{(j)}}$$

where $\ell^{(j)}$ is the degree of the isogeny $\mathfrak{l}^{(j)}$ in the j-th step.

As $p_{\mathfrak{a}_k}$ describes the chance that we pass over E_0 in the k-th step, $1/p_{\mathfrak{a}_k}$ gives us the estimated number of measurements of $\mathfrak{a} * E_{\mathrm{PK}}$ we need in order to pass over E_0 in step k. We apply this more concretely in Sections 3.2 and 3.3.

Remark 5. Instead of learning bit by bit starting from the beginning of the secret isogeny walk, we can also start at the end of the walk. To do so, we use the twist E_{-t} of the target's public key E_t, for which $\mathfrak{a} * E_{-t} = E_0$. As for the generic attack, we feed $E_{\mathrm{PK}} = \mathfrak{l}^{-1} * E_{-t}$ and $\tilde{E}_{\mathrm{PK}} = \mathfrak{l} * E_{-t}$. The computation then passes over E_0 in the *last* step instead of the *first*. This approach requires the same probability $p_{\mathfrak{a}_k}$ to recover the k-th bit, but assumes knowledge of all bits after k instead of before. Hence, we can discover starting and ending bits of $\mathfrak{a}$ in parallel.

3.2 RECOVERING SECRET KEYS IN SQALE

SQALE [102] is the most recent and most efficient constant-time implementation of CSIDH for large parameters, featuring prime sizes between 1024 and 9216 bit. In this section, we explain how the attack from Section 3.1 can be applied to SQALE, leading to a full key recovery. For concreteness, we focus on SQALE-2048, which uses parameters $n = 231$ and secret exponents $e_i \in \{-1, 1\}$ for $1 \leq i \leq 221$. The ℓ_i with $i > 221$ are not used in the group action.

Algorithmic description of SQALE.

Given a starting curve E_A, the SQALE implementation computes the group action in the following way:

- Sample random points $P_+ \in E_A[\pi - 1]$ and $P_- \in E_A[\pi + 1]$, and set $E \leftarrow E_A$.
- Iterate through $i \in \{1, \ldots, n\}$ in ascending order, and attempt to compute $\varphi : E \to \mathfrak{l}_i^{e_i} * E$ using P_+ resp P_-. Push both points through each φ.
- In case of point rejections, sample fresh points and attempt to compute the corresponding isogenies, until all $\mathfrak{l}_i^{e_i}$ have been applied.

In order to speed up computations, SQALE additionally pushes intermediate points through isogenies, which saves computational effort in following steps [107]. However, the exact design of the computational strategy inside CSIDH is not relevant for the proposed attack. Using the above description, we sketch the adaptive attack on SQALE-2048 to recover the secret key bit by bit. If there are zero point rejections, the order of steps in which $\mathfrak{a} * E_{\mathrm{PK}}$ is computed in SQALE is deterministic, and thus we can immediately apply Lemmas 4 and 5:

Corollary 6. If no point rejections occur, the computation $\mathfrak{a} * E_{\mathrm{PK}}$ in SQALE is an ordered evaluation with

$$\mathfrak{l}^{(n)} * \ldots * \mathfrak{l}^{(1)} * E_{\mathrm{PK}} = \mathfrak{l}_{221}^{e_{221}} * \ldots * \mathfrak{l}_1^{e_1} * E_{\mathrm{PK}}.$$

Hence, $p_{\mathfrak{a}_k} = \prod_{i=1}^{k} \frac{\ell_i - 1}{\ell_i}$.

SQALE uses coefficients in alternative Montgomery form $(A + 2C : 4C)$, so that passing over the curve E_0 can be detected as described in Section 3.1.

Recovering the k-th bit.

Recovering the k-th bit of a SQALE secret key works exactly as described in Figure 8, as in a successful run SQALE performs each step $\mathfrak{l}_i^{\pm 1}$ in ascending order. Thus, the k-th step, in a run where the first k steps succeed, computes $E \to \mathfrak{l}_k^{\pm 1} * E$. For the attack, we assume knowledge of the first $k - 1$ bits of the secret to produce public keys E_{PK} resp. $\tilde{E}_{\mathrm{PK}}$ that lead the target through E_0 via an application of $\mathfrak{l}_k^{-1}$ resp. $\mathfrak{l}_k$, as given by Lemma 4. For one of these cases, with probability $p_{\mathfrak{a}_k}$ (Lemma 5), the target passes over E_0 on the k-th step, and we learn the k-th secret bit e_k from side-channel information.

As k increases, $p_{\mathfrak{a}_k}$ decreases: In order for the target to pass over E_0 in one of the two cases, *all* previous isogenies have to succeed, for which Corollary 6 gives the probability $p_{\mathfrak{a}_k}$. Thus, the fact that SQALE first computes small-degree isogenies is slightly inconvenient for the attack, due to their low success probabilities. Nevertheless, attacking the last round of SQALE-2048 has a success probability of roughly $p_{\mathfrak{a}_{221}} = \prod_{j=1}^{221}(\ell_j - 1)/\ell_j \approx 19.3\%$, so that in about 1 in 5 runs, every isogeny succeeds and we pass over E_0 for the 221-th bit, compared to 2 in 3 runs to pass over E_0 for the first bit ($p_{\mathfrak{a}_1} = \frac{2}{3}$). This means that we need about three times as many measurements to discover the last bit, than the first bit. Nonetheless the required total number of measurements for all bits is very managable; we get with Lemma 5:

Corollary 7. Assuming a pass over E_0 leaks the k-th bit when the representation is strongly correlated, the estimated number of measurements to recover a SQALE-2048 key is

$$4 \cdot \sum_{k=1}^{221} \frac{1}{p_{\mathfrak{a}_k}} = 4 \cdot \sum_{k=1}^{221} \prod_{i=1}^{k} \frac{\ell_i}{\ell_i - 1} \approx 4 \cdot 1020.$$

Here, the factor 4 accounts for the fact that we need to feed both E_{PK} and $\tilde{E}_{\mathrm{PK}}$, and that only half the time ($2C : 4C$) is strongly-correlated. In practice, for more certainty, we increase the number of attempts per bit by some constant α, giving a total of $\alpha \cdot 4 \cdot 1020$ expected attempts. We come back to this in the simulations in Section 6.

3.3 RECOVERING SECRET KEYS IN CTIDH

CTIDH [20] is the most efficient constant-time implementation of CSIDH to date, although the work restricts to the CSIDH-512 and CSIDH-1024 parameter sets. We note that techniques from CTIDH can be used to significantly speed up CSIDH for larger parameters too, yet this appears to require some modifications that have not been explored in the literature yet. In this section, we explain how zero-value curve attacks can be mounted on CTIDH, leading to a partial or full key recovery, depending on the number of measurements that is deemed possible. For concreteness, we focus on the CTIDH parameter set with a 220-bit key space, dubbed CTIDH-511 in [20], which uses 15 batches of up to 8 primes. The bounds satisfy $M_i \leq 12$.

Algorithmic description of CTIDH.

Given a starting curve E_A, CTIDH computes the group action by multiple rounds of the following approach:

- Set $E \leftarrow E_A$, sample random points $P_+ \in E[\pi - 1]$ and $P_- \in E[\pi + 1]$.
- Per batch B_i, (attempt to) compute $\varphi : E \to \mathfrak{l}_{i,j}^{\mathrm{sign}(e_{i,j})} * E$ using P_+ resp P_- (or dummy when all $\mathfrak{l}_{i,j}^{e_{i,j}}$ are performed). Push both points through each φ.
- Repeat this process until all $\mathfrak{l}_{i,j}^{e_{i,j}}$ and dummy isogenies have been applied.

Furthermore, the following design choices in CTIDH are especially relevant:

- Per batch B_i, CTIDH computes real isogenies first, and (potential) dummy isogenies after, to ensure M_i isogenies are computed, independent of $(e_{i,j})$.
- Per batch B_i, CTIDH computes the actual $\ell_{i,j}$-isogenies in ascending order.
- Per batch B_i, CTIDH scales the point rejection probability to the largest value, $1/\ell_{i,1}$. This slightly changes the computation of $p_{\mathfrak{a}_k}$.
- The order in which batches are processed is deterministic.

Example 1. Let $B_1 = \{3, 5\}$ with $M_1 = 6$, and let $e_{1,1} = 2$ and $e_{1,2} = -3$. For B_1, we first try to compute $E \to \mathfrak{l}_1 * E$, until this succeeds twice. Then, we try to compute $E \to \mathfrak{l}_2^{-1} * E$, until this succeeds three times. After the real isogenies, we try to compute the remaining B_1-dummy isogeny. All B_1-isogenies, including dummies, have success probability $2/3$. If all six of the B_1-isogenies are performed but other B_i are unfinished, we skip B_1 in later rounds.

As for SQALE, the above description gives us that the order in which each $\mathfrak{l}$ is applied in CTIDH is deterministic, assuming that none of the steps fail, and so we get with Lemmas 4 and 5 again:

Corollary 8. If no point rejections occur, the computation $\mathfrak{a} * E$ in CTIDH is an ordered evaluation $\mathfrak{l}^{(n)} * \ldots * \mathfrak{l}^{(1)} * E$, with $n = \sum M_i$, including dummy isogenies.

Hence we can perform the adaptive attack on CTIDH-511 to recover the secret key bit by bit. The CTIDH implementation [20] uses coefficients in alternative Montgomery form $(A + 2C : 4C)$, but passes over to Montgomery form $(A : C)$ after each isogeny. Hence, E_0 always has a zero-value representation and we detect E_0 as described in Section 3.1. We argue in Section 5 that zero-value representations are easier to detect than strongly-correlated representations.

Recovering the k-th bit.

CTIDH introduces several difficulties for the attack, compared to SQALE. In particular, let $B_i = \{\ell_{i,1}, \ldots, \ell_{i,n_i}\}$ be the batch to be processed at step k. Then, since usually $n_i > 1$, we do not get a binary decision at each step as depicted in Figure 8, but a choice between $2 \cdot n_i$ real isogeny steps $\mathfrak{l}_{i,j}^{\pm 1}$, or possibly a dummy isogeny. In practice, with high probability, we do not need to cover all $2 \cdot n_i + 1$ options, as the following example shows.

Example 2. As CTIDH progresses through the batch ascendingly from $\ell_{i,1}$ to ℓ_{i,n_i}, the first step of a batch can often be recovered as in Figure 8, using public keys that are one $\ell_{i,1}$-isogeny away from E_0 respectively. If both do not pass over E_0, we deduce that $e_{i,1} = 0$, and we repeat this approach using an $\ell_{i,2}$-isogeny. In case of a successful attempt for $\ell_{i,j}$, we learn that the respective key element satisfies $e_{i,j} \leq -1$ resp. $e_{i,j} \geq 1$, depending on which of the binary steps was successful.[3] If we do not succeed in detecting E_0 after trying all $\ell_{i,j}^{\pm 1}$ in B_i, we learn that the target computes a B_i-dummy isogeny, and so all $e_{i,j} = 0$ for $\ell_{i,j} \in B_i$. We can easily confirm dummy isogenies: If the k-th step is a dummy isogeny, then using E_{PK} such that $\mathfrak{a} * E_{\mathrm{PK}}$ passes over E_0 in step $k - 1$, we do not move to a different curve in step k and so we observe E_0 using side-channel information after steps $k - 1$ *and* k.

This approach to recover the k-th bit in CTIDH-511 only differs slightly from Section 3.2: Given the knowledge of the secret path up to step $k - 1$, we recover the k-th step by iterating through the target batch $B_i = \{\ell_{i,1}, \ldots, \ell_{i,n_i}\}$, until we detect E_0 for a given degree $\ell_{i,j}$, or otherwise assume a dummy isogeny. This iteration becomes easier in later rounds of each batch:

- If a previous round found that some $e_{i,j}$ is positive, we only have to check for positive $\ell_{i,j}$-isogeny steps later on (analogously for negative).
- If a previous round computed an $\ell_{i,j}$-isogeny, we immediately know that the current round cannot compute an $\ell_{i,h}$-isogeny with $h < j$.

3 Note that the $e_{i,j}$ are not limited to $\{-1, 1\}$ in CTIDH, in contrast to e_i in SQALE.

- If a previous round detected a dummy isogeny for batch B_i, we can skip isogenies for B_i in all later rounds, since only dummy isogenies follow.

Thus, knowledge of the previous isogeny path significantly shrinks the search space for later steps. As in SQALE, the probability $p_{\mathfrak{a}_k}$ decreases the further we get: Batches containing small degrees ℓ_i appear multiple times, and steps with small ℓ_i have the most impact on $p_{\mathfrak{a}_k}$. For the last step $\mathfrak{l}^{(n)}$, the probability that *all* steps $\mathfrak{l}^{(k)}$ in CTIDH-511 succeed without a single point rejection, is roughly 0.3%. This might seem low at first, but the number of measurements required to make up for this probability does not explode; we are able to recover the full key with a reasonable amount of measurements as shown in Section 6. Furthermore, this probability represents the absolute lower bound, which is essentially the worst-case scenario: It is the probability that for the worst possible key, with no dummy isogenies, all steps must succeed in one run. In reality, almost all keys contain dummy isogenies, and we can relax the requirement that none of the steps fail, as failing dummy isogenies do not impact the curves passed afterwards.

Example 3. Let $B_1 = \{3, 5\}$ with $M_1 = 6$ as in CTIDH-511. Say we want to detect some step in the eighth round of some B_i for $i > 1$; it is not relevant in which of the seven former rounds the six B_1-isogenies are computed, and thus we can effectively allow for one point rejection in these rounds. This effect becomes more beneficial when dummy isogenies are involved. For example, if three of these six B_1-isogenies are dummies, we only need the three actual B_1-isogenies to be computed within the first seven rounds. Furthermore, after detecting the first dummy B_1-isogeny, we do not need to attack further B_1-isogenies as explained above, and therefore save significant attack effort.

Remark 6. The generic attack requires that all first k steps succeed. This is not optimal: Assuming that some steps fail increases the probability of success of passing over E_0. For example, to attack isogenies in the sixth round and knowing that $e_{1,1} = 5$, it is better to assume that one or two out of these five fail and will be performed after the $\ell_{i,j}$-isogeny we want to detect, than it is to assume that all five of these succeed in the first five rounds. This improves the success probability of passing over E_0 per measurement, but makes the analysis of the required number of measurements harder to carry out. Furthermore, this optimal approach highly depends on the respective private key. We therefore do not pursue this approach in our simulations. A concrete practical attack against a single private key that uses this improved strategy should require a smaller number of measurements.

Remark 7. For CTIDH with large parameters, one would expect more large ℓ_i and fewer isogenies of low degrees, relative to CTIDH-511. This improves the performance of the attack, as the probability of a full-torsion path increases, and so we expect more measurements to pass over E_0. However, the details of such an attack are highly dependent on the implementation of a large-parameter CTIDH scheme. As we know of no such implementation, we do not analyze such a hypothetical implementation in detail.

Remark 8. At a certain point, it might be useful to stop the attack, and compute the remaining key elements via a simple meet-in-the-middle search. Especially for later bits, if some dummy isogenies have been detected and most of the key elements $e_{i,j}$ are already known, performing a brute-force attack will be much faster than this side-channel attack.

§4 Recovering SIKE Keys with Side-Channel Leakage

We now apply the same strategy from Section 3 to SIKE. In this whole section, we focus on recovering Bob's static key $\mathsf{sk_B}$ by showing side-channel leakage in Derive_B, used in Decaps. In general, the idea would apply as well to recover Alice's key $\mathsf{sk_A}$ in static SIDH or SIKE with swapped roles, as we do not use any specific structure of 3-isogenies. One can easily verify that the attack generalizes to SIDH based on ℓ_A or ℓ_B-isogenies for *any* ℓ_A, ℓ_B. We repeat many of the general ideas from Section 3, with some small differences as SIKE operates in isogeny graphs over $\mathbb{F}_{p^2}$ instead of $\mathbb{F}_p$. Fortunately, these differences make the attack *easier*.

Detecting E_6.

As remarked in Section 2, the curve E_6 is represented as $(8C : 4C)$ for both representations used in SIDH and SIKE, with C some random non-zero $\alpha + \beta i \in \mathbb{F}_{p^2}$. Similar to the CSIDH situation, whenever 4α or 4β is smaller than $p/2$, doubling $4C$ does not require a modular reduction for these values, and hence the bit representation of 8α resp. 8β of $8C$ is precisely a bit shift of 4α resp. 4β of $4C$ by one bit to the left. Such strongly-correlated values can be observed in several ways using side-channel analysis, as we show in Section 5. Different from the CSIDH situation are the following key observations:

- The prime used in SIKE is of the form $p = 2^{e_A} \cdot 3^{e_B} - 1$. As observed in Remark 3, this large cofactor 2^{e_A} in $p + 1$ implies that a modular reduction does *not* affect the lowest $e_A - 1$ bits, except for the shift. Hence, even

when 4α or 4β is larger than $p/2$, we see strong correlation between their lowest bits.

- C is now an $\mathbb{F}_{p^2}$ value, so we get strong correlation between 8α and 4α *and* between 8β and 4β. This implies at least $2 \cdot (e_A - 1)$ strongly-correlated bits in the worst case (25 %), up to $2 \cdot (\log_2(p) - 1)$ strongly-correlated bits in the best case (25%).

Again, for a random curve E, the representations of the Montgomery coefficient are indistinguishable from random $(\alpha + \beta i : \gamma + \delta i)$, and so an attacker can differentiate E_6 from such curves. From this, an attacker only needs a few traces to determine accurately whether a walk passes over E_6 or not, as discussed in Section 5.

General approach to recover the k-th trit.

Assuming we know the first $k-1$ trits sk_i of a secret key sk, i.e. $\mathsf{sk}_{<k-1} = \sum_{i=0}^{k-2} \mathsf{sk}_i \cdot 3^i$, we want to find $\mathsf{sk}_{k-1} \in \{0,1,2\}$. We construct three candidate secret keys, $\mathsf{sk}^{(0)}, \mathsf{sk}^{(1)}, \mathsf{sk}^{(2)}$ as

$$\mathsf{sk}^{(i)} = \mathsf{sk}_{<k-1} + i \cdot 3^{k-1} \qquad \text{for } i \in \{0,1,2\}.$$

We must have $\mathsf{sk}_{<k} = \mathsf{sk}^{(i)}$ for some $i \in \{0,1,2\}$. Thus, we use these three keys to construct (see Lemma 9) three public keys $\mathsf{pk}^{(0)}, \mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}$ such that $\mathsf{Derive}_B(\mathsf{sk}^{(i)}, \mathsf{pk}^{(i)})$ computes E_6. When we feed these three keys to Bob, the computation $\mathsf{Derive}_B(\mathsf{sk}, \mathsf{pk}^{(i)})$ will then pass over E_6 in the k-th step *if and only if* $\mathsf{sk}_{k-1} = i$. By observing E_6 from side-channel information, we find sk_{k-1}.

In this attack scenario, another key observation makes the attack on SIDH and SIKE easier than the attack on CSIDH: The computation $\mathsf{Derive}_B(\mathsf{sk}, \mathsf{pk}^{(i)})$ *always* passes over the same curves, as there are no "steps that can fail" as in CSIDH. We know with certainty that Bob will pass over E_6 in step k in precisely one of these three computations. Hence, the number of traces required reduces drastically, as we do not need to worry about probabilities, such as $p_{\mathfrak{a}_k}$, that we have for CSIDH.

Constructing $\mathsf{pk}^{(i)}$ *from* $\mathsf{sk}^{(i)}$ *using backtracking.*

Whereas in CSIDH it is trivial to compute a curve E_{PK} such that $\mathfrak{a} * E_{\mathsf{PK}}$ passes over E_0 in the k-th step (see Lemma 4), in SIDH and SIKE it is not immediatly clear how to construct $\mathsf{pk}^{(i)}$ for $\mathsf{sk}^{(i)}$. We follow [2, § 3.3], using *backtracking* to

construct such a pk.[4] The main idea is that any $\mathsf{sk}_{<k}$ corresponds to some *kernel point* R_B of order 3^k for some k, so to an *isogeny* $\varphi^{(k)} : E_6 \to E^{(k)}$. Here, the trits sk_i determine the steps

$$E_6 = E^{(0)} \xrightarrow{\mathsf{sk}_0} E^{(1)} \xrightarrow{\mathsf{sk}_1} \dots \xrightarrow{\mathsf{sk}_{k-1}} E^{(k)}.$$

The kernel of the dual isogeny $\hat{\varphi}^{(k)} : E^{(k)} \to E_6$ is then generated by the point $\varphi^{(k)}([3^{e_B-k}]Q_B)$, by [286]. This leads to [2, Lemma 2], which we repeat here for our context.

Lemma 9 (Lemma 2, [2])**.** Let sk be a secret key, and let $\varphi : E_6 \to E^{(k)}$ be the corresponding isogeny for the first k steps, with kernel generator $R_k = [3^{e_B-k}](P_B + [\mathsf{sk}_{<k}]Q_B)$. Let $T \in E^{(k)}[3^{e_B}]$ such that $[3^{e_B-k}]T \neq \pm[3^{e_B-k}]\varphi(Q_B)$. Then

$$\mathsf{pk}' = (\varphi(Q_B) + [\mathsf{sk}_{<k}]T,\ -T,\ \varphi(Q_B) + [\mathsf{sk}_{<k} - 1]T)$$

is such that $\mathsf{Derive}_B(\mathsf{sk}, \mathsf{pk}')$ passes over E_6 in the k-th step.

It is necessary that such a pk' is not rejected by a SIKE implementation.

Corollary 10 (p. 15, [2])**.** The points P' and Q' for a $\mathsf{pk}' = (P', Q', Q' - P')$ as constructed in Lemma 9 form a basis for the 3^{e_B}-torsion of $E^{(k)}$ and therefore pass the CLN test [132].

Given Lemma 9 and $\mathsf{sk}_{<k-1}$, we can therefore easily compute the $\mathsf{pk}^{(i)}$ corresponding to $\mathsf{sk}^{(i)}$ for $i \in \{0, 1, 2\}$. One of the three attempts $\mathsf{Derive}_B(\mathsf{sk}, \mathsf{pk}^{(i)})$ will then pass over E_6 in the k-th step, while the other two will not. Only the representation of E_6 by $(8C : 4C)$ is then strongly-correlated, and by detecting this representation using side-channel information, we recover sk_{k-1}.

Remark 9. A straightforward attack computes $\mathsf{pk}^{(0)}$, $\mathsf{pk}^{(1)}$ and $\mathsf{pk}^{(2)}$, and feeds all three to Bob, and so requires 3 traces to recover a single trit sk_{k-1}. Clearly, when we already detect E_6 in the trace of $\mathsf{Derive}_B(\mathsf{sk}, \mathsf{pk}^{(0)})$, we do not need the traces of $\mathsf{pk}^{(1)}$ and $\mathsf{pk}^{(2)}$, similarly for $\mathsf{Derive}_B(\mathsf{sk}, \mathsf{pk}^{(1)})$. This approach would require on average $\frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 2 + \frac{1}{3} \cdot 3 = 2$ traces per trit. We can do even better: If we do not detect E_6 in both $\mathsf{Derive}_B(\mathsf{sk}, \mathsf{pk}^{(0)})$ and $\mathsf{Derive}_B(\mathsf{sk}, \mathsf{pk}^{(1)})$, we do not need a sample for $\mathsf{Derive}_B(\mathsf{sk}, \mathsf{pk}^{(2)})$, as sk_{k-1} *must* equal 2. This gives $\frac{5}{3}$ samples per trit, for a total of $\frac{5}{3} \cdot e_B$ traces.

4 It is important that such a $\mathsf{pk} = (P, Q, Q - P)$ passes the *CLN test* [132]: P and Q are both of order 3^{e_B} and $[3^{e_B-1}]P \neq [\pm 3^{e_B-1}]Q$, so that they generate $E[3^{e_B}]$.

§5 Feasibility of Obtaining Side-Channel Information

In this section, we discuss the practical feasibility of obtaining the required side-channel information.

Zero-value representations.

For zero-value representations as in CTIDH, where E_0 is represented by $(0 : C)$ in Montgomery form, we exploit side-channel analysis methods to distinguish between the zero curve and others. In particular, as shown in [151], one can apply Welch's t-test [336] to extract the required information from the power consumption of the attacked device. Further, as mentioned in [151], one can use correlation-collision SCA methods to identify zero values using multiple measurements. Thus, the feasibility, as demonstrated in [151] for SIKE, carries over to detect zero-value representations.

Strongly-correlated representations.

The attacks presented in Sections 3.2 and 4 for implementations using strongly-correlated representations, such as SQALE and SIKE, are more challenging in practice, since no zero values occur. A naïve approach to mount the proposed attack for such instances would be to apply side-channel attacks like CPA or DPA, and estimate or guess the values of intermediate codomain curves. Revealing those intermediate values would require a fitting power model and a sufficiently high signal-to-noise ratio[5] (SNR). By exploiting the pattern similarity in the strongly-correlated representation $(2C : 4C)$ for SQALE or $(8C : 4C)$ for SIKE, we reduce the SNR required to successfully perform the attack. To achieve this, we apply the concept of correlation-collision attacks, so that there is no need to reveal the actual value of C via a sophisticated power model.

We exploit side-channel correlation-collision attacks [283] to find similar values by searching for strongly-correlated patterns versus non-correlated patterns. Instead of measuring *multiple* computations to identify similar or identical patterns, as in [283], we apply the concept of a horizontal side-channel attack as in [290]. That is, we extract the required side-channel information from a *single* segmented power trace. Such a segmented power trace contains the power values of the processed limbs (each limb is 64 bits), required to represent $\mathbb{F}_p$-values, which form a fingerprint characteristic of such a value.

5 SNR is the ratio between the variance of the signal and the variance of noise. Too small SNR values make information and noise indistinguishable.

These fingerprints then serve as input to calculate the correlation between $2C$ and $4C$ for SQALE, or $4\alpha, 4\beta, 8\alpha$ and 8β for SIKE, from which we judge their similarity. For strongly-correlated representations of E_0 and E_6, this gives a higher correlation between the fingerprints than for representations of random curves E_a as either $(A + 2C : 4C)$ or $(A + 2C : A - 2C)$, with $A, C \neq 0$.

For both CSIDH attacks, we assume no point rejections prior to the respective isogeny computation, so that the specific isogeny steps are known in advance. For SIKE, there are no such probabilities involved in the isogeny computation, and so here too the specific isogeny steps are known in advance. This implies that an attacker will know where the values of interest are computed and used within the power trace, and can distinguish the relevant information from the rest of the trace. Thus, in all cases, the points of interest (position of the limbs) within the power trace are known in advance, and segmenting each power trace into vectors of the corresponding processed limbs for mounting the correlation-collision attack is easy.

§6 Simulation of the Attacks

To demonstrate the proposed attacks, we implemented Python (version 3.8.10) simulations for our CTIDH-511[6] and SQALE-2048[7] attacks, and a C simulation of the attack on SIKE.[8] The C code for key generation and collecting the simulated power consumption were compiled with gcc (version 9.4.0). Security-critical spots of the attacked C code remained unchanged in both cases.

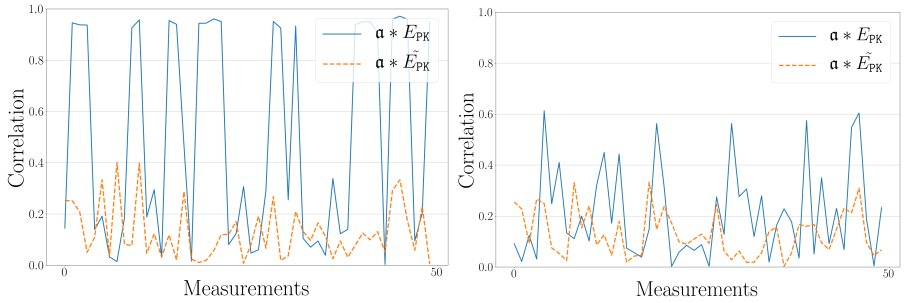
For the SQALE and CTIDH attacks, the implemented simulation works as follows: First, we generate the corresponding public keys E_{PK} and $\tilde{E}_{\mathrm{PK}}$ for the current k-th step, as described in Section 3.1. Then we collect the bit values of the resulting codomain curve after the computation of the k-th step $E \leftarrow \mathfrak{l}^{(k)} * E$ in the group action $\mathfrak{a} * E_{\mathrm{PK}}$ resp. $\mathfrak{a} * \tilde{E}_{\mathrm{PK}}$ to simulate the power consumption.

We calculate the Hamming weight of these values and add a zero-mean Gaussian standard distribution to simulate noise in the measurement. We picked different values of the standard deviation to mimic realistic power measurements with different SNR values. By varying the SNR in such a way, we can determine up to which SNR the attacks are successful, and compare this to known SNR values achieved in physical attacks. For SQALE and CTIDH, we are only interested in power traces passing over E_0, and so we need the first k

6 http://ctidh.isogeny.org/high-ctidh-20210523.tar.gz
7 https://github.com/JJChiDguez/sqale-csidh-velusqrt, commit a95812f
8 https://github.com/Microsoft/PQCrypto-SIDH, commit ecf93e9

steps to succeed. We therefore take enough samples to ensure high probability that passing over E_0 happens multiple times for either E_{PK} or $\tilde{E}_{\mathsf{PK}}$. Finally, based on the set of collected bit vectors for all these samples, we decide on which of the two cases contains paths over E_0, and therefore reveal the k-th bit of the secret key.

For the SIKE attack, we generate $\mathsf{pk}^{(0)}$, $\mathsf{pk}^{(1)}$ and $\mathsf{pk}^{(2)}$ for the current k-th step, as described in Section 4, and collect the bit values of the resulting codomain curve in the computation of the k-th step of Derive_B in Decaps. For simplicity, there is no noise in the simulation, as the results are exactly the same as for the SQALE situation after extracting the bit values. Deciding which sample has strongly-correlated values is easy, as is clear from Figure 10.



(a) Correlation results without noise. **(b)** Correlation results with SNR of 1.40.

Figure 10: Experimental results to discover bit $k = 1$: the correct hypothesis ($\mathfrak{a} * E_{\mathsf{PK}}$) in blue and the wrong hypothesis ($\mathfrak{a} * \tilde{E}_{\mathsf{PK}}$) in orange, for SQALE-2048.

As described in previous sections, due to the different representations, the decision step differs between CTIDH and SQALE. For SIKE, the probability to pass over E_6 is 100%, and so a single sample per $\mathsf{pk}^{(i)}$ is enough to decide what the k-th trit sk_{k-1} is.

In order to reduce the running time of our simulations for SQALE and CTIDH, we terminate each group action run after returning the required bit values of the k-th step. Furthermore, we implemented a threaded version so that we collect several runs in parallel, which speeds up the simulation. All experiments were measured on AMD EPYC 7643 CPU cores.

Attacking CTIDH-511.

As shown in [151, § 4] a practical differentiation between zero and non-zero values, even with low SNR, is feasible with a single trace containing the zero

value. Hence, in CTIDH, where E_0 is represented by $(0 : C)$, a single occurrence of E_0 leaks enough information for the decision in each step. Thus, the number of required attempts can be calculated as follows: Given $p_{\mathfrak{a}_k}$ from Lemma 5, the success probability of having *at least one* sequence that passes over E_0 in the k-th step in t_k attempts is $P_{\text{exp}}(X \geq 1) = 1 - (1 - p_{\mathfrak{a}_k})^{t_k}$. We can calculate t_k to achieve an expected success rate P_{exp} by $t_k = \log_{(1-p_{\mathfrak{a}_k})}(1 - P_{exp})$. For CTIDH-511, to achieve $P_{\text{exp}} \geq 99\%$ for all k, we get an estimate of $\sum t_k \approx 130,000$ attempts for full key recovery. In simulations, the required number of attempts for full key recovery was $\approx 85,000$ on average over 100 experiments, due to effects mentioned in Section 3.3. The average execution time was about 35 minutes (single core) or 5 minutes (120 threads). As described in Remark 8, finding the last few key bits by brute force drastically reduces the required measurements, as $p_{\mathfrak{a}_k}$ is low.

Attacking SQALE-2048.

In this case, we simulate a correlation-collision attack as described in Section 5: We calculate the correlation between the 64-bit limbs that represent the $\mathbb{F}_p$-values, and apply the standard Hamming-weight model with noise drawn from a normal distribution. Even with an SNR as low as 1.40, strongly-correlated representations leak enough information to guess the k-th bit, as can be seen in Figure 10 for $k = 1$. Both without noise and with SNR 1.40, we are able to determine the right bit in 74% of measurements (where 75% is the theoretical optimum, as $2C \leq \frac{p}{2}$ only half the time). An SNR value of 1.40 is considered *low*: The SNR value of a common embedded device, using a measurement script[9] provided by the ChipWhisperer framework for a ChipWhisperer-Lite board with an ARM Cortex-M4 target, obtains an SNR of 8.90. Figure 11 shows the success rate for different values.

We evaluated the following methods for decision-making:

- Decide based on the number of cases with a higher resulting correlation, as exemplified in Figure 10.
- Decide based on the sum of the resulting correlations for each case, to reduce the number of attempts required for a given success rate.

Empirical results show that the sum-based approach reduces the required number of attempts for key recovery by a factor 3 on average (from $\approx 24,819$ to $\approx 8,273$), which leads to an average execution time of 35 minutes (120 threads).

9 https://github.com/newaetech/chipwhisperer-jupyter/blob/master/archive/PA_Intro_3-Measuring_SNR_of_Target.ipynb, commit 44112f6

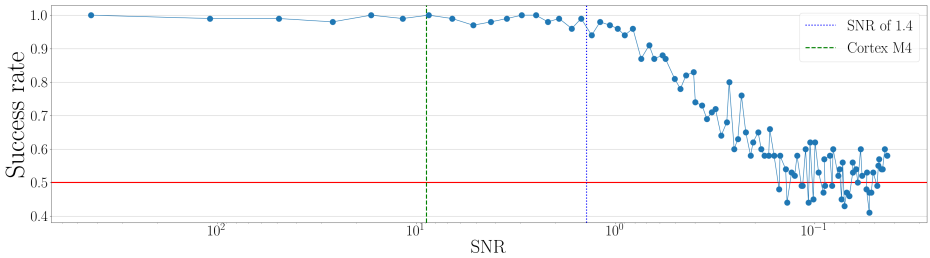


Figure 11: Relation between SNR and success rate. Rate of 0.5 equals random guess.

Attacking SIKE.

For SIKE, the analysis after collecting the bit values is similar to that of the SQALE case, and hence the results from Figure 10 apply to these simulated samples too. Furthermore, for SIKE we have the advantage that **i)** we know that one of the three samples per trit must be an E_6-sample, **ii)** we know that even with modular reduction, there is strong correlation between the lowest limbs and **iii)** we can use both $\mathbb{F}_p$-values α and β for $C = \alpha + \beta i \in \mathbb{F}_{p^2}$.

As explained in Section 4, we need on average 5/3 samples per trit to find sk_i, for all e_B trits. For SIKEp434, this gives an average of 228 samples to recover sk_B. The average running time over 100 evaluations in each case was about 4 seconds for SIKEp434, about 8 seconds for SIKEp503, about 17 seconds for SIKEp610, and about 42 seconds for SIKEp751 respectively.

Scheme	SQALE-2048	CTIDH-511	SIKEp434	SIKEp503	SIKEp610	SIKEp751
Samples	8,273	85,000	228	265	320	398

Table 1: Average number of samples to reconstruct secret key in simulations.

§7 Countermeasures and Conclusion

We have shown that both CSIDH and SIKE are vulnerable to leakage of specific curves. For CSIDH, we have shown that both Montgomery form and alternative Montgomery form leak secret information when passing over E_0, and for SIKE we have shown that in both forms, the representation of E_6 by $(8C : 4C)$ leaks secret information. As described in Section 5, zero-value representations are easiest to detect, and accordingly one should prefer the alternative Montgomery form over the Montgomery form throughout the whole computation for CSIDH

variants. However, more effective countermeasures are required to avoid strongly-correlated representations in CSIDH and SIKE.

7.1 PUBLIC KEY VALIDATION

As mentioned in Section 3, public keys in the proposed attacks on CSIDH variants consist of valid supersingular elliptic curves. Hence, the attack cannot be prevented by public key validation.

For the SIKE attack, the situation is different: Instead of containing valid points $(\varphi_A(P_B), \varphi_A(Q_B), \varphi_A(Q_B - P_B))$ (see Section 2), we construct public keys differently, as described in Lemma 9. However, such public key points are not detected by partial validation methods contained in the current SIKE software, such as the CLN test (see Corollary 10). In general, the full validation of SIDH or SIKE public keys is believed to be as hard as breaking the schemes themselves [190]. It remains an open question if there is an efficient partial validation method to detect the specific public-key points generated by our attack.

7.2 AVOIDING E_0 OR E_6

A straightforward way of mitigating the attacks is to avoid paths that lead over E_0 or E_6, or any other vulnerable curve. As argued in [2, 151], avoiding these altogether seems difficult. We discuss techniques to achieve this.

Avoid the danger zone.

Similar to the rejection proposal from [2], it may appear intuitive to define a certain *danger zone* around vulnerable curves, e.g. for CSIDH, containing all curves $\mathfrak{l}_i^{\pm 1} * E_0$ for $1 \leq i \leq n$, and abort the execution of the protocol whenever an isogeny path enters this zone. In the SIKE attack, this zone could include the four curves that are 3-isogenous to E_6. However, the attacker can simply construct public keys that would or would not pass through this zone, and observe that the protocol aborts or proceeds. This leaks the same information as the original attack.[10]

10 Pun aficionados may wish to dub this scenario the *highway to the danger zone*.

Masking on isogeny level.

One can fully bypass this danger zone by masking with a (small) isogeny before applying secret isogeny walks [2, 244]. For CSIDH, for a masking ideal $\mathfrak{z}$ and a secret $\mathfrak{a}$ we have by commutativity $\mathfrak{a} * E = \mathfrak{z}^{-1} * (\mathfrak{a} * (\mathfrak{z} * E))$, so this route avoids the danger zone when $\mathfrak{z}$ is sufficiently large. Drawing $\mathfrak{z}$ randomly from a masking-key space of k bits would require the attacker to guess the random ephemeral mask correctly in order to get a successful walk over E_0, which happens with probability 2^{-k}. Thus, a k-bit mask increases the number of samples needed by 2^k. Similarly, as detailed in [2], the secret isogeny in the SIKE attack can be masked by a 2^k-isogeny, where keeping track of the dual requires some extra cost. Although masking comes at a significant cost if the masking isogeny needs to be large, this appears to be the only known effective countermeasure that fully avoids the proposed attacks.

Randomization of order (CSIDH).

For CSIDH variants, intuitively, randomizing the order of isogenies, and as proposed in [255] the order of real and dummy isogenies, might seem beneficial to prevent our attacks. However, we can then simply *always* attack the first step of the isogeny path, with a success probability of $1/n$. With enough repetitions, we can therefore statistically guess the secret key, where the exact success probabilities highly depend on the respective CSIDH variant. This countermeasure also significantly impacts performance, making it undesirable.

Working on the surface[11] *(CSIDH).*

An interesting approach to avoid vulnerable curves, specific to CSIDH, is to move to the *surface* of the isogeny graph. That is, we use curves E_A with $\mathbb{F}_p$-rational endomorphism ring $\mathbb{Z}[\frac{1+\pi}{2}]$ instead of $\mathbb{Z}[\pi]$, and use a prime $p = 7 \mod 8$. This idea was proposed in [88] and dubbed CSURF. We are not aware of any vulnerable curves on the surface, but it seems difficult to prove that vulnerable curves do not exist there. More analysis is necessary to rule out such curves. Nevertheless, working on the surface offers other benefits, and we see no reason to work on the floor with known vulnerabilities, instead of on the surface.

11 We thank the anonymous reviewers of SAC 2022 for this suggestion.

Using radical isogenies.

In [92] an alternative method to compute the action of $\mathfrak{a}$ is proposed, using *radical isogenies*. Radical isogenies are compute on a different curve model, the Tate normal form. Hence, it could potentially be possible to avoid zero-value or strongly-correlated representations on these curve models. Nevertheless, even in such a case, to use radical isogenies efficiently current implementations move between different Tate normal forms using Montgomery curves in between. This intermediate Montgomery curve can then be targeted.

Precomposition in SIKE.

A potential countermeasure specific to SIKE is precomposing with a random isomorphism, as proposed in [244]. In our attack scenario, the isogeny walk then passes a curve isomorphic to E_6 instead of E_6, which may eliminate the leakage. However, as discussed in [133], each isomorphism class contains exactly six Montgomery curves, and the isomorphism class of E_6 also contains E_{-6}, which shares the same vulnerability as E_6. Thus, in $1/3$ of cases, leakage still occurs, only moderately increasing the number of required measurements. On the other hand, finding an isomorphism that guarantees the isogeny walk not to pass E_6 or E_{-6} only from public-key information seems infeasible. Furthermore, the computation of isomorphisms usually contains expensive square root computations.

7.3 AVOIDING CORRELATIONS

Another approach to mitigate the attacks is to ensure that vulnerable curves such as E_0 and E_6 do not leak information when passing over them. This requires adapting the representations of such curves.

Avoiding correlations in alternative Montgomery form.

As noted for CSIDH variants, the representation $(2C : 4C)$ leaks secret information whenever $2C < \frac{p}{2}$. In order to avoid this, we can try to represent the alternative Montgomery form $(A + 2C : 4C)$ differently and use a *flipped* alternative Montgomery form $(A + 2C : -4C)$ instead, which we write as ɐlʇɐɹuǝɹlʇıʌǝ Montgomery form for brevity. In the case of E_0, this means that the coefficients $2C$ and $-4C$ are *not* simple shifts of each other for $2C < \frac{p}{2}$, which prevents the correlation attack. In order to still achieve constant-time behavior, we should flip $4C$ for all curves, since otherwise E_0 would possibly be detectable via side

channels. The correctness of computations can be guaranteed by corresponding sign flips in computations that would normally include $4C$. Analogously, we can define a flipped representation of curves in SIKE.

Although the ɘʌiʇɐuɹɘʇlɐ Montgomery form is effective in preventing leakage of E_0, it creates other vulnerable curves. For example, the curve E_{-6} is *not* strongly correlated using the alternative Montgomery form, but by switching to ɘʌiʇɐuɹɘʇlɐ Montgomery form, we get the strongly-correlated representation $(-4C : -4C)$. The attack is then applicable by replacing E_0 by E_{-6}. It remains an open question to find a representation without zero-value representations or strongly-correlated representations.

Masking a single value.

Assuming we are working with the representations $(A_{24}^+ : A_{24}^-)$ or $(A_{24}^+ : C_{24})$ for either CSIDH or SIKE, masking is non-trivial, as it needs to respect the ratio A/C during the computation. However, it is possible to multiply by some random α during the computation of A_{24}^+, and to multiply by $1/\alpha$ in the next computations that use A_{24}^+. This requires a careful analysis and implementation, in order to guarantee that no leak of A_{24}^+ or some different correlation occurs at a given point in the computation.

V

DISORIENTATION FAULTS ON CSIDH

We investigate a new class of fault-injection attacks against the CSIDH family of cryptographic group actions. Our *disorientation attacks* effectively flip the direction of some isogeny steps. We achieve this by faulting a specific subroutine, connected to the Legendre symbol or Elligator computations performed during the evaluation of the group action. These subroutines are present in almost all known CSIDH implementations.

Post-processing a set of faulty samples allows us to infer constraints on the secret key. The details are implementation specific, but we show that in many cases, it is possible to recover the full secret key with only a modest number of successful fault injections and modest computational resources. We provide full details for attacking the original CSIDH proof-of-concept software as well as the CTIDH constant-time implementation. Finally, we present a set of lightweight countermeasures against the attack and discuss their security.

§1 Introduction

Isogeny-based cryptography is a contender in the ongoing quest for post-quantum cryptography. Perhaps the most attractive feature is small key size, but there are other reasons in favor of isogenies: Some functionalities appear dif-

This chapter is based on the paper

> Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska. "Disorientation faults in CSIDH". in: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 310–342.

In terms of content, I have made small editorial changes to keep notation consistent and have removed a table related to the performance of `pubcrawl` to ensure a smoother reading experience.

ficult to construct from other paradigms. For instance, the CSIDH [96] scheme gives rise to non-interactive key exchange. CSIDH uses the action of an ideal-class group on a set of elliptic curves to mimic (some) classical constructions based on discrete logarithms, most notably the Diffie–Hellman key exchange. Recently, more advanced cryptographic protocols have been proposed based on the CSIDH group action: the signature schemes SeaSign [144] and CSI-FiSh [54], threshold schemes [150], oblivious transfer [250], and more. The main drawback of isogeny-based cryptography is speed: CSIDH takes hundreds of times longer to complete a key exchange than pre-quantum elliptic-curve cryptography (ECC).

The group action in CSIDH and related schemes is evaluated by computing a sequence of small-degree *isogeny steps*; the choice of degrees and "directions" is the private key. Thus, the control flow of a straightforward implementation is directly related to the secret key, which complicates side-channel resistant implementations [20, 49, 213, 274, 276], as we have also seen in Chapter IV.

In a side-channel attack, passive observations of physical leakage (such as timing differences, electromagnetic emissions, or power consumption) during the execution of sensitive computations help an attacker infer secret information. A more intrusive class of physical attacks are *fault-injection attacks* or *fault attacks*: By actively manipulating the execution environment of a device, for instance, by altering the characteristics of the power supply, or by exposing the device to electromagnetic radiation, the attacker aims to trigger an error during the execution of sensitive computations and later infer secret information from the outputs, which are now potentially incorrect, i.e., *faulty*.

Two major classes of faults are *instruction skips* and *variable modifications*. Well-timed skips of processor instructions can have far-reaching consequences, e.g., omitting a security check entirely, or failing to erase secrets which subsequently leak into the output. Variable modifications may reach from simply randomized CPU registers to precisely targeted single-bit flips. They cause the software to operate on unexpected values, which may lead to exploitable behavior, especially in a cryptographic context. In practice, the difficulty of injecting a particular kind of fault or a combination of multiple faults depends on various parameters; generally speaking, less targeted faults are easier.

Contributions

We analyze the behavior of existing CSIDH implementations under a new class of attacks that we call *disorientation faults*. These faults occur when the attacker confuses the algorithm about the *orientation* of a point used during the

computation: The effect of such an error is that a subset of the secret-dependent isogeny steps will be performed in the opposite direction, resulting in an incorrect output curve.

The placement of the disorientation fault during the algorithm influences the distribution of the output curve in a key-dependent manner. We explain how an attacker can post-process a set of faulty outputs to fully recover the private key. This attack works against almost all existing CSIDH implementations.

To simplify exposition we first assume access to a device that applies a secret key to a given public key (i.e., computing the shared key in CSIDH) and returns the result (e.g., a hardware security module providing a CSIDH accelerator). We also discuss variants of the attack with weaker access such as a *hashed* version where faulty outputs are not revealed as-is, but passed through a key-derivation function first, as is commonly done for a Diffie–Hellman-style key exchange, and made available to the attacker only indirectly, e.g., as a MAC under the derived key.

Part of the tooling for the post-processing stage of our attack is a somewhat optimized meet-in-the-middle *path-finding* program for the CSIDH isogeny graph, dubbed `pubcrawl`. This software is intentionally kept fully generic with no restrictions specific to the fault-attack scenario we are considering, so that it may hopefully be usable out of the box for other applications requiring "small" neighborhood searches in CSIDH in the future. Applying expensive but feasible precomputation can speed up post-processing for all attack variants and is particularly beneficial to the hashed version of the attack.

To defend against disorientation faults, we provide a set of *countermeasures*. We show different forms of protecting an implementation and discuss the pros and cons of each of the methods. In the end, we detail two of the protections that we believe give the best security. Both of them are lightweight and do not significantly add to the complexity of the implementation.

Note on security.

We emphasize that CSIDH and the protocols based on the CSIDH group action are not affected by the recent attacks that break the isogeny-based scheme SIDH [86, 266, 329]. These attacks exploit specific auxiliary information which is revealed in SIDH but does not exist in CSIDH.

CSIDH is a relatively young cryptosystem, being introduced only in 2018, but it is based on older systems due to Couveignes [137] and Rostovtsev and Stolbunov [335] which have received attention since 2006. The best non-quantum attack is a meet-in-the-middle attack running in $\mathcal{O}(\sqrt[4]{p})$; a low-memory version

was developed in [156]. On a large quantum computer, Kuperberg's attack can be mounted as shown in [108]. This attack runs in $L_{\sqrt{p}}(1/2)$ calls to a quantum oracle. The number of oracle calls was further analyzed in [60] and [307] for concrete parameters, while [49] analyzes the costs per oracle call in number of quantum operations. Combining these results shows that breaking CSIDH-512 requires around 2^{60} qubit operations on logical qubits, i.e., not taking into account the overhead for quantum error correction. Implementation papers such as CTIDH [20] use the CSIDH-512 prime for comparison purposes and also offer larger parameters. Likewise, we use the CSIDH-512 and CTIDH-512 parameters for concrete examples.

Related work.

Prior works investigating fault attacks on isogeny-based cryptography mostly target specific variants or implementations of schemes and are different from our approach. *Loop-abort* faults on the SIDH cryptosystem [193], discussed for CSIDH in [78], lead to leakage of an intermediate value of the computation rather than the final result. Replacing torsion points with other points in SIDH [363, 366] can be used to recover the secret keys, but this attack cannot be mounted against CSIDH due to the structural and mathematical differences between SIDH and CSIDH.

Recently, several CSIDH-specific fault attacks were published. One can modify memory locations and observe if this changes the resulting shared secret [79]. A different attack avenue is to target fault injections against dummy computations in CSIDH [78, 255]. We emphasize that these are attacks against specific implementations and variants of CSIDH. Our work, in contrast, features a generic approach to fault attacks, exploiting an operation and data flow present in almost all current implementations of CSIDH.

§2 Preliminaries

Chapter II gives an introduction to curves and their isogenies, and a thorough introduction to CSIDH, and its variants SQALE and CTIDH. Here, we only introduce definitions specific to this work and some arithmetic details required for our attacks.

Recall that CSIDH is based on the class group action on the set of supersingular elliptic curves over $\mathbb{F}_p$ in Montgomery form $E_A : y^2 = x^3 + Ax^2 + x$ with a unique $A \in \mathbb{F}_p$, and with quadratic twist E_{-A}. We use the notation $\mathcal{E}$ in this chapter to denote this set of curves E_A with $A \in \mathbb{F}_p$.

For any ℓ_i and any $E_A \in \mathcal{E}$ there are two ℓ_i-isogenies, each leading to another curve in $\mathcal{E}$. One has kernel generated by any point P_+ of order ℓ_i with both coordinates in $\mathbb{F}_p$. We say this ℓ_i-isogeny is in the *positive direction* and the point P_+ has *positive orientation*. The other ℓ_i-isogeny has kernel generated by any point P_- of order ℓ_i with x-coordinate in $\mathbb{F}_p$ but y-coordinate in $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$. We say this isogeny is in the *negative direction* and the point P_- has *negative orientation*. Replacing E_A by the codomain of a positive and negative ℓ_i-isogeny from E_A is a *positive and negative ℓ_i-isogeny step*, respectively. As the name suggests, a positive and a negative ℓ_i-isogeny step cancel.

2.1 ALGORITHMIC ASPECTS

Every step is an oriented isogeny, so applying a single $\mathfrak{l}_i^{\pm 1}$ step requires a point P with two properties: P has order ℓ_i and the right orientation. The codomain of $E \to E/\langle P \rangle$ is computed using either the Vélu [373] or $\sqrt{\text{élu}}$ [44] formulas.

DETERMINING ORIENTATIONS. All state-of-the-art implementations of CSIDH use x-only arithmetic and completely disregard y-coordinates. So, we sample a point P by sampling an x-coordinate in $\mathbb{F}_p$. To determine the orientation of P, we then find the field of definition of the y-coordinate through a Legendre symbol computation. An alternative method is the "Elligator 2" map [45] which generates a point of the desired orientation.

SAMPLING ORDER-ℓ POINTS. There are several methods to compute points of given order ℓ. The following Las Vegas algorithm is popular for its simplicity and efficiency: As above, sample a uniformly random point P of either positive or negative orientation, and compute $Q := [(p+1)/\ell]P$. Since P is uniformly random in a cyclic group of order $p+1$, the point Q has order ℓ with probability $1 - 1/\ell$. With probability $1/\ell$, we get $Q = \infty$. Retry until $Q \neq \infty$. Filtering for points of a given orientation is straightforward.

MULTIPLE ISOGENIES FROM A SINGLE POINT. To amortize the cost of sampling points and determining orientations, implementations usually pick some set S of indices of exponents of the same sign, and attempt to compute one isogeny per degree ℓ_i with $i \in S$ from one point. If $d = \prod_{i \in S} \ell_i$ and P a random point, then the point $Q = [\frac{p+1}{d}]P$ has order dividing d. If $[d/\ell_i]Q \neq \infty$ we can use it to construct an isogeny step for $\ell_i \in S$. The image of Q under that

isogeny has the same orientation as P and Q and order dividing d/ℓ_i, so we continue with the next ℓ_j.

In CSIDH and its variants, the set S of isogeny degrees depends on the secret key and the orientation s of P. For example in Algorithm 1 [96], for the first point that is sampled with positive orientation, the set S is $\{i \mid e_i > 0\}$.

Algorithm 1 Evaluation of CSIDH group action

Input: $A \in \mathbb{F}_p$ and a list of integers $(e_1, \ldots, e_n)$.
Output: $B \in \mathbb{F}_p$ such that $\prod [\mathfrak{l}_i]^{e_i} * E_A = E_B$

while some $e_i \neq 0$ **do**
  Sample a random $x \in \mathbb{F}_p$, defining a point P.
  Set $s \leftarrow \texttt{IsSquare}\ (x^3 + Ax^2 + x)$.
  Let $S = \{i \mid e_i \neq 0,\ \text{sign}(e_i) = s\}$. If S is empty, **skip** this x.
  Let $k \leftarrow \prod_{i \in S} \ell_i$ and compute $Q \leftarrow [\frac{p+1}{k}]P$.
  for each $i \in S$ **do**
    Set $k \leftarrow k/\ell_i$ and compute $R \leftarrow [k]Q$. If $R = \infty$, **skip** this i.
    Compute $\varphi : E_A \to E_B$ with kernel $\langle R \rangle$.
    Set $A \leftarrow B$, $Q \leftarrow \varphi(Q)$, and $e_i \leftarrow e_i - s$.
return A.

The order of a random point P is not divisible by ℓ_i with probability $1/\ell_i$. This means that in many cases, we will not be able to perform an isogeny for *every* $i \in S$, but only for some (large) subset $S' \subset S$ due to P lacking factors ℓ_i in its order for those remaining $i \in S \setminus S'$. In short, a point P performs the action $\prod_{i \in S'} \mathfrak{l}_i^s$ for some $S' \subset S$, with s the orientation of P (interpreted as ± 1). Sampling a point and computing the action $\prod_{i \in S'} \mathfrak{l}_i^s$ is called a *round*; we perform rounds for different sets S until we compute the full action $\mathfrak{a} = \prod \mathfrak{l}_i^{e_i}$.

STRATEGIES. There are several ways of computing the group action as efficiently as possible, usually referred to as *strategies*. The strategy in Algorithm 1 is called *multiplicative strategy* [49, 96, 276]. Other notable strategies from the literature are the *SIMBA strategy* [274], *point-pushing strategies* [107], and *atomic blocks* [20].

1-POINT AND 2-POINT APPROACHES. The approach in Algorithm 1 samples a single point, computes some isogenies with the same orientation, and repeats this until all steps $\mathfrak{l}_i^{\pm 1}$ are processed. Such an approach is called a *1-point approach*. In contrast, one can sample two points per round, one with positive and one with negative orientation, and attempt to compute isogenies for each

degree ℓ_i per round, independent of the sign of the e_i [301]. Constant-time algorithms require choosing S independent of the secret key, and all state-of-the-art constant-time implementations use the *2-point approach* [20, 102].

KEYSPACE. In both CSIDH and CTIDH, each party's private key is an integer vector $(e_1, \ldots, e_n)$ sampled from a bounded subset $\mathcal{K} \subset \mathbb{Z}^n$, the *key space* . Different choices of $\mathcal{K}$ have different performance and security properties. The original scheme [96] uses $\mathcal{K}_m = \{-m, \ldots, m\}^n \subset \mathbb{Z}^n$, e.g. $m = 5$ for CSIDH-512. As suggested in [96, Rmk. 14] and shown in [274], using different bounds m_i for each i can improve speed. The shifted key space $\mathcal{K}_m^+ = \{0, \ldots, 2m_i\}^n$ was used in [274]. Other choices of $\mathcal{K}$ were made in [98, 102, 301], and CTIDH [20].

§3 Attack Scenario and Fault Model

Throughout this work, we assume physical access to some hardware device containing an unknown CSIDH private key $\mathfrak{a}$. In the basic version of the attack, we suppose that the device provides an interface to pass in a CSIDH public-key curve E and receive back the result $\mathfrak{a} * E$ of applying $\mathfrak{a}$ to the public key E, simulating the second step of a key exchange.

Remark 1. Diffie–Hellman-style key agreements typically *hash* the shared secret to derive symmetric key material, instead of directly outputting curves as in our scenario. Our attacks are still applicable in this *hashed version* of the attack, although the complexity for post-processing steps from Section 4 will increase significantly. To simplify exposition, we postpone this discussion to Section 7.

We assume that the attacker is able to trigger an error during the computation of the orientation of a point in a specific round of the CSIDH algorithm: whenever a point P with orientation $s \in \{-1, 1\}$ is sampled during the algorithm, we can flip the orientation $s \mapsto -s$ as shown below. This leads to some isogenies being computed in the opposite direction throughout the round. The effect of this flip will be explored in Section 4.

SQUARE CHECK. In CSIDH, cf. Algorithm 1, the point P is generated in Step 2 and its orientation s is determined in Step 3. The function `IsSquare` determines s by taking as input the non-zero value $z = x^3 + Ax^2 + x$, and computing the Legendre symbol of z. Hence, $s = 1$ when z is a square and $s = -1$ when z is not a square. Many implementations simply compute $s \leftarrow z^{\frac{p-1}{2}}$.

A successful fault injection in the computation $z \leftarrow x^3 + Ax^2 + x$, by skipping an instruction or changing the value randomly, ensures random input to `IsSquare` and so in about half of the cases the output will be flipped by $s \mapsto -s$. In the other half of the cases, the output of `IsSquare` remains s. The attacker knows the outcome of the non-faulty computation and can thus discard those outputs and continue with those where the orientation has been flipped.

Remark 2. There are other ways to flip the orientation s. For example, one can also inject a random fault into x after s has been computed, which has a similar effect. The analysis and attack of Sections 4 and 5 apply to all possible ways to flip s, independent of the actual fault injection. The countermeasures introduced in Section 9 prevent all possible ways to flip s that we know of.

Faulting the Legendre symbol computation in `IsSquare`, in general, leads to a random $\mathbb{F}_p$-value as output instead of ± 1. The interpretation of this result is heavily dependent on the respective implementation. For instance, the CSIDH implementation from [96] interprets the output as boolean value by setting $s = 1$ if the result is $+1$, and -1 otherwise. In this case, faults mostly flip in one direction: from positive to negative orientation. Thus, faulting the computation of z is superior for our attack.

ELLIGATOR. Implementations using a 2-point strategy often use Elligator 2 [45]. On input of a random value, Elligator computes two points P and P' of opposite orientations. An `IsSquare` check is used to determine the orientation of P. If P has positive orientation, we set $P_+ \leftarrow P$ and $P_- \leftarrow P'$. Otherwise, set $P_+ \leftarrow P'$ and $P_- \leftarrow P$. Again, we can fault the input to this `IsSquare` check, which flips the assignments to P_+ and P_-; hence, the orientation of *both* points is flipped.

As before, this means that all isogenies computed using either of these points are pointing in the wrong direction. A notable exception is CTIDH, where two independent calls to Elligator are used to produce points for the 2-point strategy. This is due to security considerations, and the algorithmic and attack implications are detailed in Section 5.2.

§4 Exploiting Orientation Flips

In Section 3, we defined an attack scenario that allows us to flip the orientation s in Line 3. If this happens, the net effect is that we will select an incorrect set S' with opposite orientation, and hence perform a part of the full isogeny walk in the *opposite* direction for all the indices in S'. Equivalently, the set S selected

in Line 3 has opposite orientation to the point P. For simplicity, we will always fix the set S first and talk about the point P being flipped. We assume that we can successfully flip the orientation in any round r, and that we get the result of the faulty evaluation, which is some *faulty curve* $E_t \neq \mathfrak{a} * E$.

We first study the effect of orientation flips for full-order points in Section 4.2, and then discuss effects of torsion in Section 4.3 and Section 4.4. We organize the faulty curves into components according to their orientation and round in Section 4.5 and study the distance of components from different rounds in Section 4.6. In Section 4.7, we use faulty curves to recover the secret key $\mathfrak{a}$.

4.1 IMPLICATIONS OF FLIPPING THE ORIENTATION OF A POINT

In this section, all points are assumed to have full order, so Line 7 never skips an i.

Suppose we want to evaluate the group action $\prod_{i \in S} \mathfrak{l}_i * E_A$ for some set of steps S. Suppose we generate a negatively oriented point P and flip its orientation. This does not change the point, it is still negatively oriented, but if we use P to evaluate the steps in what we believe is the *positive* direction, we will in fact compute the steps in the negative direction: $E_f = \prod_{i \in S} \mathfrak{l}_i^{-1} * E_A$. More generally, if we want to take steps in direction s and use a point of opposite orientation, we actually compute the curve $E_f = \prod_{i \in S} \mathfrak{l}_i^{-s} * E_A$.

Suppose we flip the orientation of a point in one round of the isogeny computation $E_B = \mathfrak{a} * E_A$ and the rest of the computation is performed correctly. The resulting curve E_t is called a *faulty curve*. If the faulted round computes steps in S with direction s, the resulting curve satisfies $E_t = \prod_{i \in S} \mathfrak{l}_i^{-2s} * E_B$, that is, the faulty curve differs from the correct curve by an isogeny whose degree is given by the (squares of) primes ℓ_i for $i \in S$, the set S in the round we faulted. We call S the *missing set* of E_t.

DISTANCE BETWEEN CURVES. We define the *distance* d between two curves E and E' as the lowest number of different degrees for isogenies $\varphi : E \to E'$. Note that the distance only tells us *how many* primes we need to connect two curves, without keeping track of the individual primes ℓ_i or their multiplicity. Specifically for a faulty curve with $E_B = \prod_{i \in S} \mathfrak{l}_i^{2s} * E_t$, we define the distance to E_B as the number of flipped steps $|S|$. Although each $\mathfrak{l}_i$ appears as a square, this gets counted *once* in the distance.

POSITIVE AND NEGATIVE PRIMES. Suppose the secret key $\mathfrak{a}$ is given by the exponent vector (e_i). Then every ℓ_i is used to take e_i steps in direction

$\mathrm{sign}(e_i)$. Define the set of *positive* primes $L_+ := \{i \mid e_i > 0\}$, *negative* primes $L_- := \{i \mid e_i < 0\}$, and neutral primes $L_0 := \{i \mid e_i = 0\}$. For 1-point strategies and any faulty curve E_t with missing set S, we always have $S \subset L_+$ or $S \subset L_-$. However, using 2-point strategies, the sets S may contain positive and negative primes. We use the terminology 'flipping a batch' when we refer to the effect of an orientation flip to the primes being performed: when we flip the orientation s of a negative point from negative to positive, the final result has performed a batch of positive primes in the negative direction.

Example 1. Take CSIDH-512. Assume we flip the orientation $s \mapsto -s$ of the first point P. From Algorithm 1, we see the elements of S are exactly those i such that $|e_i| \geq 1$ and $\mathrm{sign}(e_i) = -s$. Therefore, we have $S = L_{-s}$.

4.2 FAULTY CURVES AND FULL-ORDER POINTS

We continue to assume that all points have full order, so Line 7 never skips an i, and analyze which faulty curves we obtain by flipping the orientation in round r. We treat the general case in Section 4.3 and Section 4.4.

EFFECTIVE CURVES. For any strategy (cf. Section 2.1), the computation in round r depends on what happened in previous rounds.

In a 1-point strategy, we sample 1 point per round, and only perform isogenies in the direction of that point. So the set S in round r depends additionally on what was computed in previous rounds. However, the computation in round r only depends on previous rounds with *the same orientation*. The orientation of a round refers to which primes were used. Hence, a *positive* round means that the steps were performed for the positive primes, in the positive or negative direction.

In a 2-point strategy, we sample both a negative and a positive point and use these to perform the isogenies in both directions. So assuming points of full order, the round-r computation and the set S do not depend on the previous round but only the secret key.

NOTATION. Let $+$ and $-$ denote the positive and negative orientation, respectively. For a 1-point strategy, we encode the choices of orientations by a sequence of $\pm$. We denote the round r in which we flip the orientation of a point by parentheses $(\cdot)$. We truncate the sequence at the moment of the fault because the rest of the computation is computed correctly. Hence, $++(-)$ means a computation starting with the following three rounds: the first two rounds

were positive, the third one was a negative round with a flipped orientation, so the steps were computed for the negative primes, but in the positive direction.

Consider a flip of orientation in the second round. There are four possible scenarios:

$+(+)$. Two positive rounds, but the second positive batch of primes was flipped and we took the steps in negative direction instead.

$+(-)$. One positive round, one negative batch flipped to positive.

$-(+)$. One negative round, one positive batch flipped to negative.

$-(-)$. Two negative rounds, the second batch flipped to positive.

All four cases are equally likely to appear for 1-point strategies, but result in different faulty curves. Since the computation only depends on previous rounds with the same orientation, the case $+(-)$ is easily seen to be the same as $++(-)$ or $(-)$: all three are examples of cases where the orientation of the point was flipped the first time a negative round occurred. However, the cases $+(+)$ and $-(+)$ are different: the latter is equivalent to $(+)$. For example, in CSIDH, the set S for $(+)$ is $\{i \mid e_i \geq 1\}$, and the set S' for $+(+)$ is $\{i \mid e_i \geq 2\}$, differing exactly at the primes for which $e_i = 1$.

Example 2 (CSIDH)**.** For a secret key $(1, -2, -1, 3)$ in CSIDH with primes $L = \{3, 5, 7, 11\}$, the case $+(-)$ takes us to a faulty curve that is two $\{5, 7\}$-isogenies away from the desired curve, whereas the case $-(-)$ results in a curve two 5-isogenies away.

EFFECTIVE ROUND. Let $E^{r,+}$ be the faulty curve produced by the sequence $+\cdots+(+)$ of length r, and $E^{r,-}$ the curve produced by sequence $-\cdots-(-)$. We call the curves $E^{r,\pm}$ *effective round-r curves*. For a 2-point strategy, all faulty curves from round r are effective round-r curves. For 1-point strategies, effective round-r curves can be produced from other sequences as well, e.g. $+(-)$ produces the effective round-1 curve $E^{1,-}$ and $++--+(-)$ produces an effective round-3 curve $E^{3,-}$. To get an effective round-r sample $E^{r,+}$ from a round-n orientation flip, the last sign in the sequence must be $(+)$, and the sequence must contain a total of r pluses.

Lemma 1. Assume we use a 1-point strategy. The probability to get any effective round-r sample if we successfully flip in round n is equal to $\binom{n-1}{r-1} \cdot \frac{1}{2^{n-1}}$.

Remark 3. For a 2-point strategy, all curves resulting from a fault in round r are effective round-r curves.

TORSION SETS. Define the set $S^{r,s}$ as the missing set of the effective round-r curve with orientation s, i.e., $E_B = \prod_{i \in S^{r,s}} \mathfrak{l}_i^{2s} * E^{r,s}$. We refer to such sets as (missing) *torsion sets*.

Example 3 (CSIDH)**.** The torsion sets $S^{1,\pm}$ were already discussed in Example 1. In general, $S^{r,+} = \{i \mid e_i \geq r\}$ and $S^{r,-} = \{i \mid e_i \leq -r\}$.

4.3 MISSING TORSION FROM POINTS OF NON-FULL ORDER

In Section 4.2, we worked under the unrealistic assumption that all points we encounter have full order. In this section, we relax this condition somewhat: we assume that every point up until round r has full order, and so all isogenies were computed, but the point P generated in round r potentially has smaller order. In this case, we say that P is *missing torsion*. The remaining relaxation of non-full order points in earlier rounds is postponed to Section 4.4.

If the point P used to compute isogenies in round r does not have full order, the faulty curve E_t will differ from the effective round-r curve $E^{r,s}$ by the primes ℓ_i with $i \in S^{r,s}$ which are missing in the order of P.

ROUND-R FAULTY CURVES. For simplicity, assume that we are in round r, in the case $+\cdots+(+)$, and that none of the isogenies in the previous rounds failed. In round r, a negative point P is sampled, but we flip its orientation, so the batch of positive primes will be computed in the negative direction.

If the point P has full order, we obtain the curve $E^{r,+}$ at the end of the computation, which differs from E_B exactly at primes contained in $S^{r,+}$. If, however, the point P does not have full order, a subset $S \subset S^{r,+}$ of steps will be computed, leading to a different faulty curve E_t. By construction, the curve E_t is related to E_B via $E_B = \prod_{i \in S} \mathfrak{l}_i^2 * E_t$.

If we repeat this fault in T runs, we should find a set of different faulty curves E_t. Let $n(E_t)$ be the number of times the curve E_t occurs among the T samples. For each faulty curve E_t, we know $E_B = \prod_{i \in S_t} \mathfrak{l}_i^{2s} * E_t$, where $S_t \subset S^{r,+}$ is determined by the order of P_t. As P_t is a randomly sampled point, it has probability $\frac{1}{\ell_i}$ that its order is not divisible by ℓ_i. This gives us the probability to end up at E_t: the order of the point P_t should be divisible by all ℓ_i for $i \in S_t$, but not by those ℓ_i for $i \in S^{r,+} \setminus S_t$. This is captured in the following result.

Proposition 2. Let P_t be a random negative point, where we flip the orientation s to positive. The probability that we compute the faulty curve $E_t = \prod_{i \in S_t} \mathfrak{l}_i^{-2} * E_B$ is exactly $p_t = \prod_{i \in S_t} \frac{\ell_i - 1}{\ell_i} \cdot \prod_{i \in S^{r,+} \setminus S_t} \frac{1}{\ell_i}$.

Remark 4. In CTIDH, the success probability of each point to match that of the smallest prime in the batch to hide which prime is handled. But for fixed batches, an analogous results to Proposition 2 can be given.

The expected number of appearances $n(E_t)$ of a curve E_t is $n(E_t) \approx p_t \cdot T$ for T runs. As $\frac{\ell_i - 1}{\ell_i} \geq \frac{1}{\ell_i}$ for all ℓ_i, the probability p_t is maximal when $S_t = S^{r,+}$. We denote this probability by $p^{r,+}$. Hence, the curve that is likely to appear the most in this scenario over enough samples, is the curve $E^{r,+}$ which we defined as precisely that curve with missing set $S^{r,+}$. For now, we focused solely on the positive curves. Taking into account the negative curves too, we get:

Corollary 3. Let $E^{r,+} = \prod_{i \in S^{r,+}} \mathfrak{l}_i^{-2} * E_B$ and let $E^{r,-} = \prod_{i \in S^{r,-}} \mathfrak{l}_i^{2} * E_B$. Then $E^{r,+}$ and $E^{r,-}$ have the highest probability to appear among the effective round-r faulty curves. As a consequence, the largest two values $n(E)$ of all effective round-r curves are most likely $n(E^{r,+})$ and $n(E^{r,-})$

Example 4 (CSIDH)**.** Take the set $S^{1,+} = \{i \mid e_i \geq 1\}$ and let $p^{1,+}$ denote the probability that a random point P has order divisible by all primes in $S^{1,+}$. This probability depends on the secret key (e_i), but can be estimated if we collect enough faulty curves. Moreover, if $e_1 \neq 0$, then $\ell_1 = 3$ dominates either $p^{1,+}$ or $p^{1,-}$ through the relatively small probability of $2/3$ that P has order divisible by 3. Thus, if the largest pile of faulty curves is $E^{1,\pm}$, we expect $S^{1,\pm}$ not to contain $i = 1$. Thus, if e_1 is positive, $p^{1,-}$ is larger than $p^{1,+}$ and so we expect $n(E^{1,-})$ to be larger than $n(E^{1,+})$. In this case, we would expect to see another faulty curve E_t with $n(E_t)$ half the size of $n(E^{1,+})$; this curve E_t has *almost* full missing set $S^{1,+}$, but does not miss the 3-isogeny. That is, $S_t = S^{1,+} \setminus \{1\}$, with probability $p_t := \frac{1}{\ell_1} \cdot \frac{\ell_1}{\ell_1 - 1} \cdot p^{1,+} = \frac{1}{2} \cdot p^{1,+}$. This curve E_t is "close" to $E^{1,+}$; they are distance 1 apart, precisely by $\mathfrak{l}_1^2$.

The precise probabilities $p^{r,+}$ and $p^{r,-}$ depend highly on the specific implementation we target. Given an implementation, the values of $p^{r,+}$ and $p^{r,-}$ allow for a concrete estimate on the size of $n(E)$ for a specific curve E. Because the ℓ_i that are missing in the order of P_t skip the misoriented steps, the curves in the neighborhood of $E^{r,+}$ differ by two ℓ_i-isogenies for $i \in S^{r,+} \setminus S_t$ in positive direction while those around $E^{r,-}$ differ by two ℓ_i-isogenies for $i \in S^{r,-} \setminus S_t$ in negative direction.

DISTANCE BETWEEN SAMPLES. We can generalize Example 4 for any two faulty curves E_t and $E_{t'}$ that are effective round-r samples of the same orientation, using Proposition 2.

Corollary 4. Let E_t and $E_{t'}$ both be effective round-r samples with the same orientation s and missing torsion sets S_t and $S_{t'}$. Let S_Δ denote the difference in sets S_t and $S_{t'}$, i.e., $S_\Delta = (S_t \setminus S_{t'}) \cup (S_{t'} \setminus S_t)$. Then E_t and $E_{t'}$ are distance $|S_\Delta|$ apart, by $E_t = \left(\prod_{i \in S_{t'} \setminus S_t} \mathfrak{l}_i^{2s} \cdot \prod_{i \in S_t \setminus S_{t'}} \mathfrak{l}_i^{-2s}\right) * E_{t'}$. In particular, any effective round-r curve E_t with orientation s is close to $E^{r,s}$ and so in particular S_Δ is small.

Example 5 (CSIDH)**.** For a secret key $(2,3,1,2)$ in CSIDH with primes $L = \{3,5,7,11\}$, the first positive point with full torsion P will perform a 3, 5, 7 and 11-isogeny, so $S^{1,+} = \{1,2,3,4\}$, with $S^{2,+} = \{1,2,4\}$ and $S^{3+} = \{2\}$. In one faulted run, the first point P_{t_1} might only have $\{5,7,11\}$-torsion, while in another faulted run the first point P_{t_2} might only have $\{3,7,11\}$-torsion. The faulty curves E_{t_1} and E_{t_2} differ from E^+ by two 3-isogenies and two 5-isogenies, respectively, and have a distance 2 towards each other: their S_Δ is $\{1,2\}$, so they are two $\{3,5\}$-isogenies apart. The two samples E_{t_1} and E_{t_2} therefore show that both 1 and 2 are in S^+ and show that $e_1 \geq 1$ and $e_2 \geq 1$.

Corollary 4 will be essential to recover information on $S^{r,+}$ out of the samples E_t: Recovering small isogenies between samples allows us to deduce which i are in $S^{r,+}$ or $S^{r,-}$, and so leaks information about e_i.

4.4 TORSION NOISE

Orthogonally to Section 4.3, we now examine the case that missing torsion occurred in an earlier round than the round we are faulting.

Example 6 (CSIDH)**.** Suppose that $e_1 = 1$ and that in the first positive round, the point generated in Line 2 of Algorithm 1 was (only) missing ℓ_1-torsion and so the ℓ_1-isogeny attempt fails in the first positive step. Consider now the second positive round. From Section 4.2, we would expect to be computing steps in $S^{2,+} = \{i \mid e_i \geq 2\}$. But no ℓ_1-isogeny has been computed in the first round, so it will be attempted in this second positive round. If we now fault the second positive point, we obtain a faulty curve that is *also missing* ℓ_1, that is, $E_t = \mathfrak{l}_1^{-2} * E^{2,+}$. Unlike the faulty curves from Section 4.3, the positively oriented isogeny goes from E_t *towards* $E^{2,+}$. Also, note that in this scenario if $e_1 = 2$, a fault in round 2 would still result in the curve $E^{2,+}$, because the set $S^{2,+}$ contains ℓ_1 already, and so the missed ℓ_1-isogeny from round 1 will be computed in later rounds.

We refer to the phenomenon observed in Example 6 as *torsion noise*. More concretely, torsion noise happens when we fault the computation in round r for

a run which is computing an ℓ_i-isogeny in round r for $|e_i| < r$ because it was skipped in a previous round.

Torsion noise is more rare than missing torsion but still happens: the isogeny computation needs to fail and the fault must come when we are "catching up" with the computation. For CSIDH, torsion noise can only happen if $r > |e_i|$ and the computation of the ℓ_i-isogeny failed in at least $r - |e_i|$ rounds. Torsion noise is less likely for large ℓ_i because the probability that an isogeny fails is about $1/\ell_i$. For small primes, such as $\ell_i \in \{3, 5, 7\}$, we observe a lot of torsion noise. This can slightly affect the results as described in Section 4.3, but has no major impact on the results in general. Concretely, torsion noise may make it impossible to determine the correct e_i for the small primes given only a few faulted curves. Nevertheless, their exact values can be brute-forced at the end of the attack.

Faulty curves affected by torsion noise require contrarily oriented isogenies to the curves $E^{r,s}$ than the remaining faulty curves. Therefore, if torsion noise happens and we find a path from such a curve $E_t \to E^{r,s}$, then we can infer not just the orientation of the primes in this path, but often also bound the corresponding exponents e_i.

4.5 CONNECTING CURVES FROM THE SAME ROUND

Suppose we have a set of (effective) round-r faulty curves with the same orientation s, and suppose r and s are fixed. In Corollary 4, we show that such curves are close to each other. As the path from E_t to $E^{r,s}$ uses only degrees contained in $S^{r,s}$, finding these short paths among faulty curves gives us information about $S^{r,s}$, and hence about the secret key.

COMPONENT GRAPHS. Give a set $\{E_t\}$ of round-r faulty curves with orientation s, we can define the graph $G^{r,s}$ as follows: The vertices of $G^{r,s}$ are given by $\{E_t\}$, and the edges are steps between the curves, labeled by i if the curves are connected by two ℓ_i-isogenies. For convenience, we sparsify the graph $G^{r,s}$ and regard it as a tree with $E^{r,s}$ as the root.

Starting from a set of faulty curves, it is easy to build the graphs $G^{r,s}$. We can identify the *roots* of these graphs $E^{r,s}$ using Corollary 3. Then the distance from the root to any round-r faulty curve with the same orientation is small (cf. Corollary 4). Therefore, we can find the edges by applying short walks in this graph. Edges of $G^{r,s}$ then give information on $S^{r,s}$.

If we do not have enough faulty curves $\{E_t\}$, it may not be possible to connect all faulty curves with single steps (understood as isogenies of square degree,

see Corollary 4). For convenience, we assume that we have enough curves. In practice, we include in the graph $G^{r,s}$ any curve on the path between E_t to $E^{r,s}$, taking steps with square prime degree.

We imagine the graphs $G^{r,s}$ as subgraphs of the *isogeny graph* of supersingular elliptic curves with edges given by isogenies. Computing short paths from $E^{r,s}$ will give us enough edges so that we can consider the graphs $G^{r,s}$ to be connected. Hence we call them *components*.

SECRET INFORMATION. An effective round-r faulty curve E_t with torsion set $S_t \subset S^{r,+}$ is connected to $E^{r,+}$ by a path with edges $S^{r,+} \setminus S_t$. Moreover, the orientation $E^{r,+} \to E_t$ is positive. Therefore, we can identify which components are positive, and all the labels of the edges are necessarily in $S^{r,+}$, that is, the prime ℓ_i is positive. This allows us to determie L_+, and similarly L_-. We recognize torsion noise instead by its opposite direction of edges. In either case, the components $G^{r,s}$ give us the orientation of all the primes occurring as labels of the edges.

SORTING ROUND-R SAMPLES. Suppose we are given a set of round-r faulty curves $\{E_t\}$ and we want to construct $G^{r,s}$, but we do not have information about the orientation yet. We use Corollary 3 to find the roots $E^{r,s}$ of the graph and take small isogeny steps to connect samples E_t to these roots. until we no unconnected samples are left, and we have two connected components G_1, G_2. It is easy to determine the direction of the edges given enough samples; ignoring torsion noise, the positively oriented root $E^{r,+}$ will have outgoing edges.

In summary, we try to move curves E_t from a pile of unconnected samples to one of the two graphs by finding collisions with one of the nodes in $G^{r,+}$ resp. $G^{r,-}$. The degrees of such edges reveal information on $S^{r,+}$ and $S^{r,-}$: An edge with label i in $G^{r,+}$ implies $i \in S^{r,+}$, and analogously for $G^{r,-}$ and $S^{r,-}$. We furthermore infer information on L_+ and L_- in this process. Figure 12 summarizes this process, where, e.g., $E^{r,+} \to E_7$ shows missing torsion and $E_8 \to E^{r,+}$ is an example of torsion noise.

4.6 CONNECTING THE COMPONENT GRAPHS

We now explain how to connect the two components $G^{r,s}$ for different rounds r. The distance of these components is related to the sets $S^{r,+}$ and $S^{r,-}$. We then show that it is computationally feasible to connect the components via a meet-in-the-middle attack, with help of the unfaulted curve E_B. Connecting

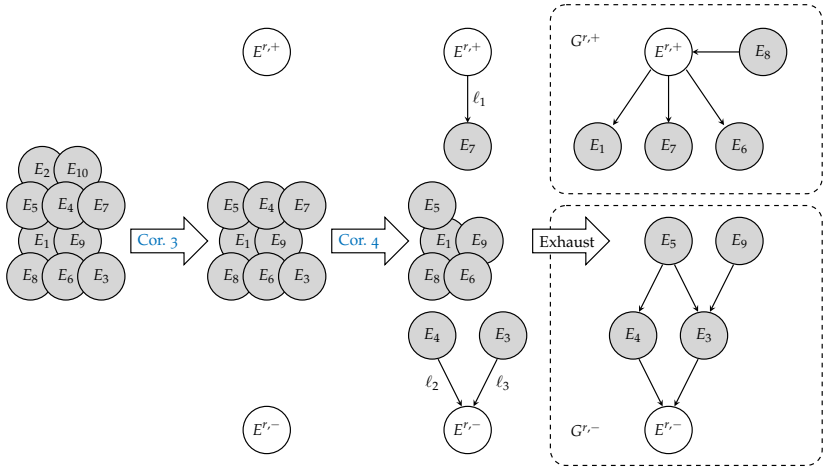


Figure 12: Building up the component graphs of faulty curves.

two components gives us almost complete knowledge on the sets $S^{r,+}$ and $S^{r,-}$, as the path $E^{r,s} \to E_B$ is made of precisely the steps in $S^{r,s}$. Thus, connecting all components $G^{r,s}$ to E_B is enough to reveal the secret $\mathfrak{a}$.

Example 7 (CSIDH)**.** Recall that we have $S^{r,+} = \{i \mid e_i \geq r\}$, and so $E^{r,+} = \prod_{i \in S^{r,+}} \mathfrak{l}_i^{-2} * E_B$. This means that, e.g., we have $S^{3,+} \subset S^{2,+}$, and $E^{2,+}$ has a larger distance from E_B than $E^{3,+}$. The path between $E^{3,+}$ and $E^{2,+}$ then only contains steps of degrees ℓ_i such that $i \in S^{2,+} \setminus S^{3,+}$, so $e_i = 2$. In general, it is easy to see that finding a single isogeny that connects a node E_{t_3} from $G^{3,+}$ and a node E_{t_2} in $G^{2,+}$ immediately gives the connection from $E^{3,+}$ to $E^{2,+}$. Hence, we learn all ℓ_i with $e_i = 2$ from the components $G^{3,+}$ and $G^{2,+}$. See also Figure 13.

In the general case, if we find an isogeny between two such graphs, say $G^{r,+}$ and $G^{r',+}$, we can compute the isogeny between the two roots $E^{r,+}$ and $E^{r',+}$ of these graphs. The degree of this isogeny $E^{r,+} \to E^{r',+}$ describes precisely the *difference* between the sets $S^{r,+}$ and $S^{r',+}$. The example above is the special case $r' = r + 1$, and in CSIDH we always have $S^{(r+1),+} \subset S^{r,+}$, so that the difference between $S^{r,+}$ and $S^{(r+1),+}$ is the set of ℓ_i such that $e_i = r$. In other CSIDH-variants, such sets are not necessarily nested, but connecting all components

still reveals e_i as Section 4.7 shows. In general, we connect two subgraphs by a distributed meet-in-the-middle search which finds the shortest connection first.

DISTANCE BETWEEN CONNECTED COMPONENTS. As we have shown, connecting two components $G^{r,+}$ and $G^{r',+}$ is equivalent to finding the difference in sets $S^{r,+}$ and $S^{r',+}$. The distance between these sets heavily depends on the implementation, as these sets are determined by the key $\mathfrak{a}$ and the evaluation of this key. For example, in CSIDH-512, the difference between $S^{r,+}$ and $S^{(r+1),+}$ are precisely those $e_i = r$, which on average is of size $\frac{74}{11} \approx 6.7$. In practice, this distance varies between 0 and 15. For an implementation such as CTIDH-512, the sets $S^{r,+}$ are smaller in general, on average of size 7, and the difference between such sets is small enough to admit a feasible meet-in-the-middle connection. See Section 6 for more details on how we connect these components in practice.

4.7 REVEALING THE PRIVATE KEY

So far, we have shown how connecting different components $G^{r,+}$ and $G^{r',+}$ reveals information on the difference between the sets $S^{r,+}$ and $S^{r',+}$. In this section, we show that when all components are connected, we can derive the secret $\mathfrak{a}$. This wraps up Section 4: Starting with disorientations in certain rounds r, we derive the secret $\mathfrak{a}$ from the resulting graph structure, assuming enough samples.

By connecting the graphs of all rounds, including the one-node-graph containing only the unfaulted curve E_B, we learn the difference between the sets $S^{r,s}$ and $S^{(r+1),s}$ for all rounds r. A single isogeny from any $G^{r,s}$ to $E_B = \mathfrak{a} * E_A$ then recovers $S^{r,s}$ for this round r: Such an isogeny gives us an isogeny from $E^{r,s} = \prod_{i \in S^{r,s}} \mathfrak{l}_i^{-s2} * E_B$ to E_B, whose degree shows us exactly those $\ell_i \in S^{r,s}$. From a connection between the components $G^{r,s}$ and $G^{r',s}$, we learn the difference in sets $S^{r,s}$ and $S^{r',s}$. Given $S^{r,s}$, we can then deduce $S^{r',s}$.

Therefore, if all graphs $G^{r,+}$ for all rounds r are connected, and we have at least one isogeny from a node to E_B, we learn the sets $S^{r,s}$ for all rounds r. From the knowledge of all sets $S^{r,s}$, we then learn $\mathfrak{a} = (e_i)$: the sign of e_i follows from observing in which of the sets $S^{r,+}$ or $S^{r,-}$ the respective ℓ_i appears, and $|e_i|$ equals the number of times of these appearances.

In practice however, due to missing torsion and torsion noise, connecting all components may not give us the *precise* sets $S^{r,+}$ resp. $S^{r,-}$. In such a case, one can either gather more samples to gain more information, or try to brute-force

the difference. In practice, we find that the actual set $S^{r,+}$ as derived from $\mathfrak{a}$ and the fuzzy set $\tilde{S}^{r,+}$ derived from our attack (leading to some $\mathfrak{a}'$) almost always have only a small difference. A simple meet-in-the-middle search between $\mathfrak{a}' * E_A$ and $\mathfrak{a} * E_A$ then quickly reveals the errors caused by missing torsion and torsion noise.

4.8 COMPLEXITY OF THE ATTACK

The full approach of this section can be summarized as follows:

1. Gather effective round-r samples E_t per round r, using Lemma 1.
2. Build the components $G^{r,s}$ using Corollaries 3 and 4.
3. Connect $G^{r,s}$ and $G^{r',s}$ to learn the *difference* in sets $S^{r,s}$ and $S^{r',s}$.
4. Finally, compute the precise sets $S^{r,s}$ for every round r to recover $\mathfrak{a}$.

The overall complexity depends on the number of samples E_t available, but is in general dominated by Item 3. For Item 2, nodes are in most cases relatively close to the root $E^{r,s}$ or to an already connected node E_t, as shown in Corollary 4.

For Item 3, the distance between components $G^{r,s}$ and $G^{r',s}$ depends heavily on the specific design choices of an implementation. In a usual meet-in-the-middle approach, where n is the number of ℓ_i over which we need to search and d is the distance between $G^{r,s}$ and $G^{r',s}$, the complexity of finding a connection is $\mathcal{O}(\binom{n}{d/2})$. Note that we can use previous knowledge from building components or finding small-distance connections between other components to reduce the search space and thus minimize n for subsequent connections. We analyze this in detail for specific implementations in Section 5.

§5 CASE STUDIES: CSIDH AND CTIDH

Section 4.8 defines a general strategy in four steps to recover the secret key $\mathfrak{a}$ in class group action computations using disorientation faults. The practical details of these steps depend highly on what specific implementation is used. To showcase the approach from Section 4 concretely, we attack two main implementations: CSIDH-512 [96] in Section 5.1, and CTIDH-512 [20] in Section 5.2. We analyze other implementations briefly in Section 5.3. For the remainder of the section, we always input E_0 into the target, which then computes a faulty version of $E_B = \mathfrak{a} * E_0$, its own public key. This makes the analysis a bit easier.

5.1 BREAKING CSIDH-512

The primes used in CSIDH-512 [96] are $L = \{3, 5, \ldots, 377, 587\}$ of size $|L| = 74$, and secret exponents are sampled as $e_i \in \{-5, \ldots, 5\}$ uniformly at random for all $i \in \{1, \ldots, 74\}$. For any $k \in \{-5, \ldots, 5\}$ we expect about $\frac{1}{11} \cdot 74$ primes ℓ_i with $e_i = k$. We expect to see about $\frac{5}{11} \cdot 74 \approx 33.6$ positive and negative primes each, and roughly $\frac{1}{11} \cdot 74 \approx 6.7$ neutral primes. Algorithm 1 shows the group action evaluation, using a 1-point strategy. In particular, after generating a point with orientation s, we get $S = \{i \mid e_i \neq 0,\ \mathrm{sign}(e_i) = s\}$. If the value of s is flipped, we set $S = \{i \mid e_i \neq 0,\ \mathrm{sign}(e_i) = -s\}$, but we perform the steps in direction s. We go over the four steps to secret-key recovery defined in Section 4.8.

BUILDING THE COMPONENT GRAPHS. Item 2 of the attack on CSIDH-512 works exactly as described in Section 4.5. If E_t and $E_{t'}$ are effective samples from the same round r with the same orientation t, their distance is small (Corollary 4). We can thus perform a neighborhood search on all of the curve samples until we have 10 connected components $G^{r,s}$ for $r \in \{1, \ldots, 5\}$, as in Figure 12. This step is almost effortless: most curves will be distance 1 or 2 away from the root $E^{r,s}$. In practice, using round information and number of occurrences, we identify the root nodes $E^{r,s}$ easily We then explore all paths of small length from those 10 roots, or connect them via a meet-in-the-middle approach (e.g., using `pubcrawl`, see Section 6). The degrees of the isogenies corresponding to the new edges in $G^{r,s}$ then reveal information on the sets $S^{r,s}$. This allows us to reduce the search space when connecting remaining components.

FILTER-AND-BREAK IT, UNTIL YOU MAKE IT. Item 3 is the most computationally intensive step, as it connects 11 components ($G^{r,s}$ and E_B) into a single large connected component. We argue that this step is practical for CSIDH-512. Specifically, we want to find the connecting isogenies between $G^{r,s}$ and $G^{(r+1),s}$, as well as between $G^{5,s}$ to E_B. This gives us 10 connections, corresponding to the gaps $\{i \mid e_i = k\}$ for $k \in [-5, 5] \setminus \{0\}$. Figure 13 shows an abstraction of this large connected component.

Since there are 74 primes in total, and only 10 gaps, at least one of these gaps is at most 7 primes. If we assume that at least 5 of the exponents are 0 (we expect ≈ 7 to be 0), then the smallest distance is at most 6 steps. Such gaps are easily found using a meet-in-the-middle search.

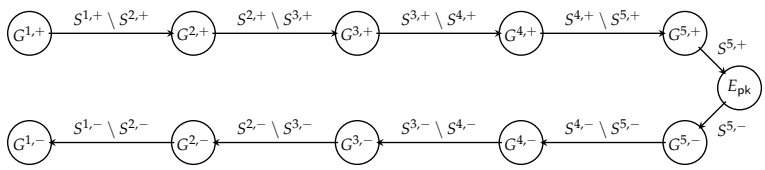


Figure 13: Large connected component associated to an attack on CSIDH-512.

We define the *support* of a search as the set of isogeny degrees used in a meet-in-the-middle neighborhood search. We can connect all components by a meet-in-the-middle search with support $\{\ell_1, \ldots, \ell_{74}\}$, but this becomes infeasible for large distances, so instead, we adaptively change the support. We start by finding short connections, and use the labels we find to pick a smaller support for searching between certain components, i.e., filter some of the ℓ_i out of the support.

First, we learn the orientation of the components by identifying $G^{1,s}$ and considering the direction of the edges. Effective round-1 samples do not have torsion noise, so the root $E^{1,+}$ has only outgoing edges, whereas the root $E^{1,-}$ has only incoming edges. The labels of the edges of $G^{1,+}$ must be positive primes $\ell_i \in L_+$, and all subsequent components $G^{r,s}$ in which these ℓ_i are must therefore also be positive. Next, all the labels that appear as degrees of edges in $G^{r,+}$ for any r are necessarily positive primes $\ell_i \in L_+$. Finally, positively oriented components $G^{r,+}$ can only be connected by positive primes $\ell_i \in L_+$, so we can remove from the support all the primes that we know are in L_-, and vice versa for the other search.

After finding the first connection between two components, we restrict the support even more: we know that any label i appears in at most *one* component connection $G^{r,s} \to G^{r's}$. Hence, we can reduce the support for further searches by those labels that have already appeared in a connection. The smaller support then allows us to find larger connections. We repeat this procedure with more and more restrictions on the support until we have connecteded all 11 components. We then recover the sets $S^{r,s}$ and compute the secret key $\mathfrak{a}$ as described in Section 4.7.

Example 8 (Toy CSIDH-103)**.** Figure 14 shows the resulting connected graph for a toy version of CSIDH using Algorithm 1 with the first $n = 21$ odd primes and private keys in $\{-3, \ldots, +3\}^n$. Each round was faulted 10 times.

The distances between the components are very small and hence connecting paths are readily found. We sparsify the graph to plot it as a spanning tree; the edges correspond to positive steps of the degree indicated by the label. This graph comes from the secret key $\mathfrak{a}$ with corresponding indices

$$(-1,+1,+2,+3,-2,+3,+2,+3,+1,+2,-3,-3,+2,+3,-2,-3,-2,+2,+1,-3,\ 0).$$

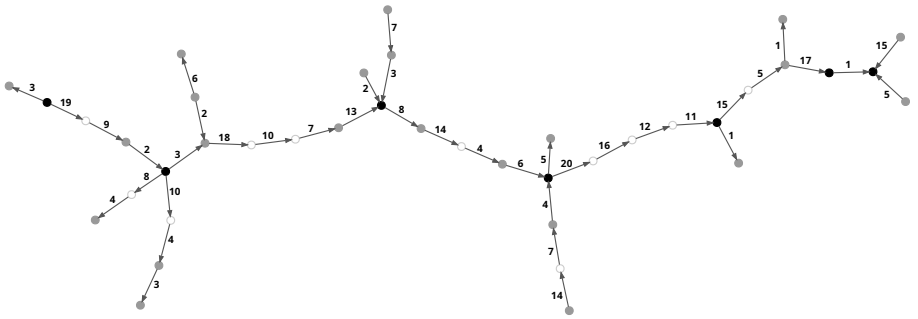


Figure 14: Example isogeny graph of faulty curves obtained from attacking toy CSIDH-103 from Example 8. An edge labeled i denotes the isogeny step $\mathfrak{l}_i$. The E_B curve and the root nodes $E^{r,s}$ are rendered in black starting with $E^{1,+}$ on the left, and ending with $E^{1,-}$) on the right. Other faulty curves appearing in the dataset are gray, and white circles are "intermediate" curves discovered while connecting the components. The indices j appearing on the connecting path between $E^{i,s}$ and $E^{i+1,s}$ are exactly those ℓ_j with $e_j = s \cdot i$. For example, only the indices appearing between $E^{1,+}$ and $E^{2,+}$, i.e. $j = 2, 9, 19$, have exponent $e_j = +1$ in the secret key.

Required number of samples.

Recovering the full secret exponent vector in CSIDH-512 requires us to compute all sets $S^{r,s}$. To do so, we build the components $G^{r,s}$ by acquiring enough effective round-r samples. More effective round-r samples may give more vertices in $G^{r,s}$, and more information about $S^{r,s}$.

Let T_r be the number of effective round-r samples and let $T = \sum T_r$. A first approach is to inject in round r until the probability is high enough that we have enough effective round-r samples. For CSIDH-512, we take $T_1 = 16$, $T_2 = 16$, $T_3 = 32$, $T_4 = 64$ and $T_5 = 128$, for a total of $T = 256$. From Lemma 1, we then

expect 8 round-5 samples (4 per orientation) and the probability that we do not get any of the elements of $G^{5,+}$ or $G^{5,-}$ is about 1.7%.

This strategy can be improved. We usually need round-5 samples, and so in any case we need T_5 rather large, in comparison to T_i with $i < 5$, to ensure we get effective round-5 samples. But gathering effective round-5 samples should already give us many samples from rounds before. Using Lemma 1 with $T_5 = 128$, we get on average 8 effective round-1 samples, 32 effective round-2 samples, 48 effective round-3 samples, 32 effective round-4 samples and 8 effective round-5 samples. In general, attacking different rounds offers different tradeoffs: attacking round 9 maximizes getting effective round-5 samples, but getting a round-1 sample in round 9 is unlikely. Faulting round 1 has the benefit that all faulty curves are effective round-1 curves, making them easy to detect in later rounds: no torsion noise appears, and missing torsion quickly allows us to determine the orientation of small primes, which reduces the search space for connecting the components. Finally, note that gathering T faulty samples requires approximately $2T$ fault injections, since, on average, we expect to flip the orientations in only 50% of faults.

5.2 BREAKING CTIDH-512

CTIDH [20] partitions the set of primes ℓ_j into b batches, and bounds the number of isogenies N_i per batch $\mathcal{B}_i$. For a list $N \in \mathbb{Z}^b_{>0}$ with $\sum N_i = n$ and a list of non-negative bounds $m \in \mathbb{Z}^b_{\geq 0}$, we define the key space as

$$\mathcal{K}_{N,m} := \left\{(e_1, \ldots, e_n) \in \mathbb{Z}^n \;\middle|\; \textstyle\sum_{j=1}^{N_i} |e_{i,j}| \leq m_i \text{ for } 1 \leq i \leq b\right\},$$

where $(e_{i,j})$ is a reindexed view of (e_i) given by the partition into batches.

CTIDH-512 uses 14 batches $\mathcal{B}_i$ with bounds $m_i \leq 18$, which requires at least 18 rounds using a 2-point strategy. In every round, we compute one isogeny per batch in either the positive or negative direction. Hence, all round-r samples are effective round-r samples.

INJECTING FAULTS. To sample oriented points, CTIDH uses the Elligator-2 map twice. First, Elligator is used to sample two points P_+ and P_- on the starting curve E_A. A direction s is picked to compute an isogeny, the point P_s is used to take a step in that direction to a curve $E_{A'}$, and the point P_s is mapped through the isogeny. Then another point P'_{-s} is sampled on $E_{A'}$ using Elligator. We will always assume that we inject a fault into only one of these two Elligator

calls (as in Section 3) and so, as for CSIDH and 1-point strategies, we again always obtain either positively or negatively oriented samples.

DIFFERENT ROUNDS FOR CTIDH-512. Per round, CTIDH performs one $\ell_{i,j}$ per batch $\mathcal{B}_i$. Within a batch, the primes $\ell_{i,j}$ are ordered in ascending order: if the first batch is $\mathcal{B}_1 = \{3, 5\}$ and the exponents are $(2, -4)$, then we first compute 2 rounds of 3-isogenies in the positive direction, followed by 4 rounds of 5-isogenies in the negative direction. We visualize this as a queue $[3+, 3+, 5-, 5-, 5-, 5-]$, which gets padded on the right with dummy isogenies for any remaining rounds up to m_1. CTIDH inflates the failure rate of each isogeny to that of the smallest prime in the batch to hide how often each prime is used; in our example, the failure rate is $1/3$, regardless of whether the degree is 3 or 5.

We thus find that the sets $S^{r,s}$ contain precisely the r-th prime in the queue per batch. With 14 batches and an equal chance for either orientation, we expect that each $S^{r,s}$ will contain about 7 primes. Furthermore, each set $S^{r,s}$ contains only one prime $\ell_{i,j}$ from the batch $\mathcal{B}_i$. This small number of batches and the precise ordering of primes within the batches make CTIDH especially easy to break using our disorientation attack.

COMPONENTS FOR CTIDH-512. Given enough samples, we construct the graphs $G^{r,s}$. The slightly higher failure probability of each isogeny somewhat increases the chances of missing torsion and torsion noise, but this has no practical effect. The distance of the root curves $E^{r,s}$ to the non-faulted curve E_B is bounded by the number of batches. Per round r, the sum of the distances of $E^{r,+}$ to E_B and of $E^{r,-}$ to E_B is at most 14, so we expect the average distance to be about 7. In stark contrast to the CSIDH-512 case, this makes it almost immediately feasible to connect $E^{r,s}$ to E_B directly.

Furthermore, the distance between two graphs $G^{r,s}$ and $G^{(r+1),s}$ is often small. We focus on positive orientation (the negative case is analogous). The distance between $G^{r,+}$ to $G^{(r+1),+}$ is given by the set difference of $S^{r,+}$ and $S^{(r+1),+}$. If these sets are disjoint and all primes in round r and $r+1$ are positive, the distance is 28, but this is unlikely. We expect significant overlap: The set difference of $S^{r,+}$ and $S^{(r+1),+}$ contains precisely those indices i such that either the last ℓ_i-isogeny is computed in round r or the first ℓ_i-isogeny is computed in round $r+1$. Note that these replacements need not come in pairs. In the first case, the prime ℓ_i is replaced by the next isogeny ℓ_j from the same batch only if ℓ_j is also positive. In the second case, the prime ℓ_i might have followed a negative prime that preceded it in the batch.

Therefore, given $S^{r,+}$, one can very quickly determine $S^{(r+1),+}$ by leaving out some ℓ_i's or including subsequent primes from the same batch. In practice, this step is very easy. Finding one connection $E_B \to E^{r,+}$ determines some set $S^{r,+}$, which we use to quickly find $S^{r+1,+}$. This approach naturally also works going backwards, to the set $S^{(r-1),+}$.

DIRECTED MEET-IN-THE-MIDDLE. Using a meet-in-the-middle approach, we compute the neighborhood of E_B and all the roots $E^{r,s}$ (or components $G^{r,s}$) up to distance 4, which connects E_B to all roots $E^{r,s}$ at distance at most 8. Disregarding orientation and information on batches, if we have N curves that we want to connect, the naive search will require about $2 \cdot \binom{74}{4} \cdot N \approx 2^{21} \cdot N$ isogenies. The actual search space is much smaller, as we can exclude searches using degrees $\ell_{i,j}$ and $\ell_{i,j'}$ from the same batch $\mathcal{B}_i$.

Moreover, isogenies in batches are in ascending order: if in round r we find that $\ell_{i,3}$ from batch $\mathcal{B}_i$ is used, none of the rounds $r' > r$ will involve the first two primes $\ell_{i,1}$ and $\ell_{i,2}$, and none of the rounds $r' < r$ will involve $\ell_{i,j} \in \mathcal{B}_i$ with $j > 3$ for that orientation. Late rounds typically contain many dummy isogenies and the corresponding faulty curves E_t are especially close to E_B, in particularly also $E^{r,s}$. Therefore, we usually rapidly recover $S^{r,s}$ for late rounds, and work backwards to recover earlier rounds.

Required number of samples.

In CTIDH, we can choose to inject a fault into the first call of Elligator or the second one. We do not see a clear benefit of prioritizing either call. Unlike for CSIDH and 1-point strategies, there is no clear benefit from targeting a specific round. If we perform c successful faults per round per Elligator call, we expect to get samples for both orientations per round. As CTIDH-512 performs 18 rounds, we require $T = 18 \cdot 2 \cdot c$ successful flips. Although in practice, due to isogeny steps failing, CTIDH typically requires 22 rounds, this has no significant implications for our attacks. Experimentally, we can recover secret keys using $c = 1$, hence $T = 36$ (or up to $T = 44$) samples, using moderate computer power.

With just one sample per round r (and per orientation s), the torsion effects will be significant and we will often not be able to recover $S^{r,s}$ *precisely*. Nevertheless, we recover some set $\tilde{S}^{r,s}$, and we can correct some of these torsion errors by looking at $\tilde{S}^{r',s}$ for rounds r' close to r. This torsion correction works as follows, consider for the moment only primes from the same batch $\mathcal{B}$:

- If no prime from $\mathcal{B}$ is contained in either $\tilde{S}^{r,+}$ or $\tilde{S}^{r,-}$ then either all primes from $\mathcal{B}$ have been performed or *missing torsion* must have happened. We can examine the primes in $\mathcal{B}$ that occur in neighboring rounds $\tilde{S}^{(rs1),s}$ and use the ordering in the batch to obtain guesses on which steps should have been computed, if any.
- If one prime from $\mathcal{B}$ is contained in $\tilde{S}^{r,+} \cup \tilde{S}^{r,-}$, we have no errors.
- If two primes from $\mathcal{B}$ are contained in $\tilde{S}^{r,+} \cup \tilde{S}^{r,-}$, then the smaller one must have come from missing torsion in a previous round and can be removed.

Remark 5. It is possible to skip certain rounds to reduce the number of samples, and recover the missing sets $S^{r,s}$ using information from the neighboring rounds. We did not perform the analysis as to which rounds can be skipped, as we feel that already two successful faults per round, and thus a total of only $T = 44$ flips at most, are low enough.

Even such a partial attack (obtaining information only from a few rounds) reveals a lot about the secret key thanks to the batches being ordered, and can significantly reduce the search space for the secret key. One may also select the rounds to attack adaptively, based on the information recovered from $S^{r,s}$.

Once we recover all the sets $S^{r,s}$, we quickly find the secret key $\mathfrak{a}$. If we misidentify $S^{r,s}$ due to torsion effects, we may have to perform a small search to correct for such mistakes.

5.3 OTHER VARIANTS OF CSIDH

In this section, we discuss some of the other implementations of CSIDH: all of these use `IsSquare` checks in the process of point sampling and are vulnerable to our attack. We analyze SIMBA [274], dummy-free implementations [3, 98, 107], and SQALE [102].

SIMBA. Implementations using SIMBA [274] can be attacked similarly to CSIDH (cf. Section 5.1). SIMBA divides the n primes ℓ_i into m *prides* (batches), and each round only computes ℓ_i-isogenies from the same pride. That is, each round only involves up to $\lceil n/m \rceil$ isogenies, and the setup of the prides is publicly known. In each round, fewer isogenies are computed, the sets $S^{r,s}$ are smaller and the distances between the components $G^{r,s}$ are shorter. It is therefore easier to find isogenies connecting the components, and recover the secret key.

DUMMY-FREE CSIDH. Dummy-free implementations [3, 98, 107] replace pairs of dummy ℓ_i-isogenies by pairs of isogenies that effectively cancel each other [98]. This is due to the fact that $\mathfrak{l}_i * (\mathfrak{l}_i^{-1} * E) = \mathfrak{l}_i^{-1} * (\mathfrak{l}_i * E) = E$. Thus, computing one ℓ_i-isogeny in positive direction and one ℓ_i-isogeny in negative direction has the same effect as computing two dummy ℓ_i-isogenies. However, this approach requires fixing the parity of each entry of the private key e_i, e.g., by sampling only even numbers from $[-10, 10]$ to reach the same key space size as before. Such dummy-free implementations mitigate certain fault attacks, such as skipping isogenies, which in a dummy-based implementation would directly reveal if the skipped isogeny was a dummy computation and give respective information on the private key.

In these implementations all isogenies start in the correct direction, we learn $|e_i|$ from disorientation faults if we know in which round the first $\mathfrak{l}_i$ is applied in the opposite direction. Therefore, if we apply the attack of Section 4 and learn all sets $S^{r,s}$, we can determine e_i precisely. Even better, it suffices to only attack every second round: It is clear that each prime will have the same orientation in the third round as in the second round, in the fifth and fourth, et cetera.

SQALE. SQALE [102] only uses exponent bounds $e_i \in \{-1, 1\}$. To get a large enough key space, more primes ℓ_i are needed; the smallest instance uses 221 ℓ_i. SQALE uses a 2-point strategy and only requires a single round, although the isogeny computation may fail and require further rounds.

Set $S^+ = S^{1,+} = \{i \mid e_i = 1\}$ and $S^- = S^{1,-} = \{i \mid e_i = -1\}$. If the sampled points in round 1 have full order, the round 1 faulty curves are either:

- the 'twist' of E_B: all the directions will be flipped (if both points are flipped),
- the curve $E^+ = (\prod_{S^+} \mathfrak{l}_i^{-2}) * E_B$, if the positive point was flipped,
- or the curve $E^- = (\prod_{S^-} \mathfrak{l}_i^{2}) * E_B$, if the negative point was flipped.

As $|S^+| \approx |S^-| \approx n/2 > 110$, we will not be able to find an isogeny to either of these curves using a brute-force or a meet-in-the-middle approach. However, SQALE samples random points to perform this single round, and so some of the isogeny computation will fail with non-negligible probability, producing faulty curves close to $E^{\pm}$. Getting enough faulty curves then allows the attacker to get the orientation of all the primes ℓ_i, and the orientation of the primes is exactly the secret key in SQALE. We note that [107] in a different context proposes to include points of full order into the system parameters and public keys such that missing torsion and torsion noise do not occur. If this is used for

SQALE, our attack would not apply, however, this incurs a costly verification that such points are of full order.

§6 The Pubcrawl Tool

The post-processing stage of our attack relies on the ability to reconstruct the graph of connecting isogenies between the faulty CSIDH outputs. We solve this problem by a meet-in-the-middle neighborhood search in the isogeny graph, which is sufficiently practical for the cases we considered. In this section, we report on implementation details and performance results for our pubcrawl software.[1]

We emphasize that the software is *not* overly specialized to the fault-attack setting and may therefore prove useful for other "small" CSIDH isogeny searches appearing in unrelated contexts.

ALGORITHM. The pubcrawl software implements a straightforward meet-in-the-middle graph search: Grow isogeny trees from each input node simultaneously and check for collisions; repeat until there is only one connected component left. The set of admissible isogeny degrees ("support") is configurable, as are the directions of the isogeny steps ("sign", cf. CSIDH exponent vectors), the maximum number of isogeny steps to take from each target curve before giving up ("distance"), and the number of prime-degree isogenies done per graph-search step ("multiplicity", to allow for restricting the search to square-degree isogenies).

SIZE OF SEARCH SPACE. The number of vectors in $\mathbb{Z}^n$ of 1-norm less than m is given by [116, § 3]

$$G_n(m) = \sum_{k=0}^{m} \binom{n}{k} \binom{m-k+n}{n}.$$

Similarly, the number of vectors in $\mathbb{Z}^n_{\geq 0}$ of 1-norm $\leq m$ equals

$$H_n(m) = \sum_{k=0}^{m} \binom{k+n-1}{n-1}.$$

1 The name refers only to *crawl*ing the graph of *pub*lic keys, and definitely *not* to the act of visiting a sequence of pubs, consuming one or more drinks at each.

IMPLEMENTATION. The tool is written in C++ using modern standard library features, most importantly hashmaps and threading. It incorporates the latest version of the original CSIDH software as a library to provide the low-level isogeny computations. Public-key validation is skipped to save time. The shared data structures (work queue and lookup table) are protected by a simple mutex; more advanced techniques were not necessary in our experiments.

We refrain from providing detailed benchmark results for the simple reason that the overwhelming majority of the cost comes from computing isogeny steps in a breadth-first manner, which parallelizes perfectly. Hence, both time and memory consumption scale almost exactly linearly with the number of nodes visited by the algorithm.

Concretely, on a server with two Intel Xeon Gold 6136 processors (offering a total of 24 hyperthreaded Skylake cores) using GCC 11.2.0, we found that each isogeny step took between 0.6 and 0.8 core milliseconds, depending on the degree. Memory consumption grew at a rate of roughly 250 bytes per node visited, although this quantity depends on data structure internals and can vary significantly.

There is no doubt that pubcrawl could be sped up if desired, for instance by computing various outgoing isogeny steps at once instead of calling the CSIDH library as a black box for each individually.

The pubcrawl software is available at

https://yx7.cc/code/pubcrawl/pubcrawl-latest.tar.xz.

§7 Hashed Version

As briefly mentioned in Remark 1, the attacker-observable output in Diffie–Hellman-style key agreements is usually not the shared elliptic curve, but a certain derived value. Typically, the shared elliptic curve is used to compute a key k using a key derivation function, which is further used for symmetric key cryptography. In such cases, we cannot expect to obtain (the Montgomery coefficient of) a faulty curve E_t but only a derived value such as $k = \text{SHA-256}(E_t)$ or even $\text{MAC}_k(\text{str})$ for some known fixed string str.

The attack strategies from Section 4 and Section 5 exploit the connections between the various faulty curves, but when we are only given a derived value, we are unable to apply isogenies. We argue that our attack, however, still extends to this more realistic setting as long as the observable value is computed deterministically from E_t and collisions do not occur. For simplicity,

we will refer to the observable values as *hashes* of the faulty curves and denote these $\mathsf{H}(E_t)$. Starting from a faulty curve E_t, we assume we can easily compute the hashed value $\mathsf{H}(E_t)$, but we cannot recover E_t from the hash $h = \mathsf{H}(E_t)$.

As we lack the possibility to apply isogenies to the hashes, we must adapt the strategy from Section 4. Given a set of faulty curves, we can no longer generate the neighborhood graphs, nor find connecting paths between these graphs, and it is harder to learn the orientation of primes, which helped to reduce the possible degrees of the isogenies when applying pubcrawl. If we only see hashes of the faulty curves, we cannot immediately form the neighborhood graphs and determine orientations. But from the frequency analysis (Corollary 3), we can still identify the two most frequent new hashes h_1, h_2 per round as the probable hashes of $\mathsf{H}(E^{r,s})$.

To recover E given a hash $\mathsf{H}(E)$, we run a one-sided pubcrawl search starting from E_B, where we hash all the curves we reach along the way, until we find a curve that hashes to $\mathsf{H}(E)$. In practice, we run pubcrawl with one orientation (or both, in parallel) until we find $\mathsf{H}(E^{r,s})$ for some r. Having identified $E^{r,s}$, we can then run a small neighborhood search around $E^{r,s}$ to identify the hashes of the faulty curves E_t close to $E^{r,s}$ to recover $G^{r,s}$. In contrast to the unhashed version, in the hashed version we can only recover the faulty curves E_t by a one-sided search from a known curve E, instead of a meet-in-the-middle attack. In particular, the only known curve at the beginning of the attack is the public key E_B.

Example 9 (CSIDH-512)**.** The distance of the curves $E^{r,s}$ to E_B is given by $|\{i \mid s \cdot e_i \geq r\}|$. Therefore, the curves $E^{5,s}$ have the smallest distance to E_B. Starting from the public key E_B, we thus first search the paths to the curves $E^{5,s}$. We do this by growing two neighborhoods (with positive and negative orientation) from E_B. Recall from Section 5.1 that the *average* distance $E_B \to E^{5,s}$ is about $74/11 \approx 7$, but can be larger in practice. Very large distances are rare: for example, the probability of both $E^{5,+}$ and $E^{5,-}$ having distance larger than 10 from E_B is $\sum_{n=11}^{74-11} \sum_{m=11}^{74-n} \left(\binom{74}{n}\binom{74-n}{m} 9^{74-n-m}\right) / 11^{74} \approx 0.3\%$. Hence, we expect to find a connection to at least one of the curves $E^{5,s}$ within distance 10, meaning that we expect the first connection to cost at most $2\sum_{i=0}^{10} \binom{74}{i} \approx 2^{40.6}$ isogeny step evaluations and likely less for at least some $\mathsf{H}(E_t)$ in the neighborhood. From there, we can reduce the support for remaining searches, making this first step the dominating factor in the complexity of this attack.

Example 10 (CTIDH-512)**.** The faulty curves for *any* round in CTIDH are closer to the public key E_B than in the CSIDH case: it is 14 in the worst case (one prime per batch all having the same orientation) and the distance is 7 on average

(Section 5.2). So the directed pubcrawl searches up to distance ≈ 7 (one with positive and one with negative orientation) are very likely to identify many of the hashed curves. Once we identify some faulty curves, we can identify other faulty curves quickly by small neighborhood searches thanks to the extra ordered structure of the CTIDH key space . We also benefit from the slightly increased probability of failure leading to more curves in the neighborhood of $E^{r,s}$.

SUMMARY. In the hashed version, the main difference compared to the approach in Section 5 is that we can no longer mount a meet-in-the-middle attack starting from E_B and from all faulty curves but can only search starting from E_B. Hence, we do not get the square-root speedup from meeting in the middle. Despite this increase in cost, it is still realistically possible to attack the hashed version. Other sizes and variants work the same way with the concrete numbers adjusted. The brute-force searches to connect the effective round-r curves in large CSIDH versions do get expensive remain far cheaper than the security level in bits.

§8 Exploiting the Twist

In this section, we explain how to use the twist E_B^T and precomputation to significantly speed up obtaining the private key $\mathfrak{a}$, given enough samples E_t, especially for the "hashed" version described in Section 7.

As before, the attack target is a public key $E_B = \mathfrak{a} * E_0$. Previously (Section 3), we attacked the computation of $\mathfrak{a} * E_0$ with disorientation faults. In this section, we will use E_{-B} as the input curve instead: Negating B is related to inverting $\mathfrak{a}$ because $E_{-B} = \mathfrak{a}^{-1} * E_0$. Moreover, applying $\mathfrak{a}$ to E_{-B} gives us back the curve E_0 and faulting this computation then produces faulty curves close to the fixed curve E_0. As E_{-B} is the *quadratic twist* of E_B, we will refer to this attack variant as *using the twist*.

The main trick that allows precomputation is that twisting induces a symmetry around the curve E_0. This can be used to speed up pubcrawl: the opposite orientation of E_t (starting from E_0) reaches E_{-t}, so we can check two curves at once. By precomputing a set $\mathcal{C}$ of curves of distance at most d to E_0, a faulty curve E_t at distance $d' \leq d$ is in $\mathcal{C}$ and can immediately be identified via a table lookup. Note that $\mathcal{C}$ can be precomputed once and for all, independent of the target instance, as for any secret key $\mathfrak{a}'$ the faulty curves end up close to E_0. The symmetry of E_{-t} and E_t also reduces storage by half.

Finally, this twisting attack cannot be prevented by simply recognizing that E_{-B} is the twist of E_B and refusing to apply the secret $\mathfrak{a}$ to such a curve: An attacker can just as easily pick a random masking value $\mathfrak{z}$ and feed $\mathfrak{z} * E_{-B}$ to the target device. The faulty curves E_t can then be moved to the neighborhood of E_0 by computing $\mathfrak{z}^{-1} * E_t$ at some cost per E_t, or the attacker can precompute curves around $\mathfrak{z} * E_0$. The latter breaks the symmetry of E_t and E_{-t} and does not achieve the full speedup or storage reduction, but retains the main benefits.

TWISTING CTIDH. The twisting attack is at its most powerful for CTIDH. As noted before, the sets $S^{r,s}$ are small in every round for CTIDH. The crucial observation is that in each round and for each orientation, we use at most one prime per batch (ignoring torsion noise, see Section 4.4). For a faulty curve E_t, the path $E_t \to E_0$ includes only steps with the same orientation and uses at most one prime per batch. With batches of size N_i, the total number of possible paths per orientation is $\prod_i (N_i + 1)$, which is about $2^{35.5}$ for CTIDH-512. Hence, it is possible to precompute *all* possible faulty curves that can appear from orientation flips from *any* possible secret key $\mathfrak{a}$.

Extrapolating the performance of pubcrawl (Section 6), this precomputation should take no more than a few core years. The resulting lookup table occupies $\approx 3.4\,\text{TB}$ when encoded naively, but can be compressed to less than 250 GB using techniques similar to [368, § 4.3].

TWISTING CSIDH. For this speed-up to be effective, the distance d we use to compute $\mathcal{C}$ must be at least as large as the smallest $|S^{r,s}|$. Otherwise, no faulty curves end up within $\mathcal{C}$. For CSIDH, the smallest such sets are $S^{r_{\max},s}$, where $r_{\max}$ is the maximal exponent permitted by the parameter; e.g., for CSIDH-512 $r_{\max} = 5$ and $S^{5,s}$ have an expected size ≈ 7. Precomputing $\mathcal{C}$ for $d \leq 7$ creates a set containing $\sum_{i=0}^{7} \binom{74}{i} \approx 2^{31}$ curves. Such a precomputation will either identify $S^{5,s}$ immediately, or allow us to find these sets quickly by considering a small neighborhood of the curves $E^{5,s}$.

Note that for all the earlier rounds $r < r_{\max}$, the sets $S^{r,s}$ include $S^{r_{\max},s}$. Therefore, if we have the orientation s and the set $S^{r_{\max},s}$, we can shift all the faulty curves by two steps for every degree in $S^{r_{\max},s}$. If we have misidentified the orientation, this shift moves the faulty curves in the wrong direction, away from E_0. This trick is particularly useful for larger r as eventually many isogenies need to be applied in the shifts and we will have identified the orientation of enough primes so that the search space for pubcrawl becomes small enough to be faster.

TWISTING IN THE HASHED VERSION. Precomputation extends to the hashed version from Section 7: we simply include $\mathsf{H}(E_t)$ in $\mathcal{C}$ instead of E_t for all E_t in some neighborhood of E_0. Again, this works best when attacking a hashed version of CTIDH and the effective round-$r_{\max}$ curves in CSIDH. To use precomputation for different rounds, one can replace the starting curve E_{-B} that is fed to the target device by the shift given exactly by the primes in $S^{r_{\max},S}$ (or, adaptively, by the part of the secret key that is known). This has the same effect as above: shifting all the curves E_t with the same orientation *closer* towards E_0, hopefully so that the $\mathsf{H}(E_t)$ are already in our database. If they are not then likely the opposite orientation appeared when we faulted the computation.

SUMMARY. The benefit of using the twist with precomputation is largest for the hashed versions: we need a brute-force search from E_0 in any case, and so we would use on average as many steps per round as the precomputation takes. For the non-hashed versions, the expensive precomputation competes with meet-in-the-middle attacks running in square-root time. This means that in the hashed version we do not need to amortize the precomputation cost over many targets and have a clear tradeoff between memory and having to recompute the same neighborhood searches all over again and again.

§9 Countermeasures

In this section, we present countermeasures against disorientation faults. We first review previous fault attacks on CSIDH and their countermeasures, as well as their influence on our attack and then discuss new countermeasures for one-point sampling from CSIDH and Elligator. The estimated costs of these countermeasures are given in Section 9.3.

9.1 PREVIOUS FAULT ATTACKS AND COUNTERMEASURES

One way to recover secret keys is to target dummy isogenies with faults [78, 255]. Although these attacks are implementation-specific, the proposed countermeasures impact our attack too. Typically, real isogenies are computed prior to dummy isogenies, but the order of real and dummy isogenies can be randomized [78, 255]. When applied to dummy-based implementations [274, 301], this randomization means dummy isogenies can appear in different rounds for each run, which makes the definitions of the curves $E^{r,\pm}$ almost obsolete. However, we can instead simply collect many faulted round-1 samples. Each

faulty curve E_t reveals a different set S_t due to the randomization, and with enough samples, a statistical analysis will quickly reveal all the $e_{i,j}$ just from the number of appearances among the sets S_t, again recovering the secret key.

Adapted to CTIDH, there are two possible variants of this randomization countermeasure: One could either keep the queue of real isogenies per batch as described above, but insert dummy isogenies randomly instead of at the end of the queue, or fully randomize the order of isogeny computations per batch including the dummy operations. In the first case, faulting round r if a dummy isogeny is computed in batch $\mathcal{B}_i$ means that no prime from this batch appears in the missing set. This effect is the same as missing torsion and thus our attack remains feasible. The net effect matches increased failure probabilities p_i and the larger neighborhoods simplify finding orientations. Note also that p_i is inflated more for batches with more dummy isogenies. In the second case, when the entire queue is randomized, the same arguments as for CSIDH apply, and we can recover the secret key from statistical analysis of round-1 samples.

Many fault attacks produce invalid intermediate values. Campos, Kannwischer, Meyer, Onuki, and Stöttinger [78] propose some low-level protections for dummy isogenies to detect fault injections, but such an approach does not prevent our disorientation attack, and is orthogonal to our proposed countermeasures. Its performance overhead for the CSIDH-512 implementation from [301] is reported to be 7%.

Faulting memory locations can identify dummy isogenies [79]. In addition to the countermeasures above, Campos, Krämer, and Müller [79] recommend using dummy-free implementations when concerned about fault attacks, with a roughly twofold slowdown [98]. However, as described in Section 5.3, dummy-free implementations are vulnerable to disorientation faults too.

Lastly, [78] reports that its fault attack could theoretically lead to a disorientation fault. Although the probability for this to happen is shown to be negligible, they nonetheless propose to counter this attack vector by checking the field of definition of each kernel generator (point R in Step 7 of Algorithm 1). This is rather expensive, with an overhead of roughly 30% for the implementation from [301], but also complicates the disorientation faults proposed in this work. We note that our countermeasures are significantly cheaper, but do not prevent the theoretical fault effect from [78].

9.2 PROTECTING SQUARE CHECKS AGAINST FAULT ATTACKS

The attack described in Section 3 can be applied to all implementations of CSIDH that use a call to `IsSquare` to determine the orientations of the involved

point(s). The main weakness is that the output of `IsSquare` is always interpreted as $s = 1$ or $s = -1$, and there is no obvious way of reusing parts of the computation to verify that the output is indeed related to the x-coordinate of the respective point.

REPEATING SQUARE CHECKS. One way to reduce the attacker's chances for a successful fault is to add redundant computations and repeat the execution of `IsSquare` k times. In principle, this means that the attacker has to fault all k executions successfully, hence reducing the overall fault success probability to $1/2^k$. However, if an attacker manages to reliably fault the computation of z or the Legendre symbol computation or to skip instructions related to the redundant computations, they might be able to circumvent this countermeasure.

Repeated square checks have been proposed for a different fault attack scenario [78]. There, `IsSquare` is used to verify the correct orientation for each point that generates an isogeny kernel. However, this countermeasure significantly impacts the performance of CSIDH, and could be bypassed as above.

USING Y-COORDINATES. In CSIDH, the field of definition of the y-coordinate determines the orientation of a point. So, another simple countermeasure relying on redundant computation is to work with both x- and y-coordinates, instead of x-only arithmetic. We can then easily recognize the orientation of each point. But this leads again to a significant performance loss due to having to keep y-coordinates during all point multiplications and isogeny evaluations. We expect that this countermeasure is significantly more expensive than repeating `IsSquare` k times for reasonable choices of k.

USING PSEUDO Y-COORDINATES. We propose a more efficient countermeasure: compute *pseudo* y-coordinates after sampling points. We sample a random x-coordinate and set $z = x^3 + Ax^2 + x$. If z is a square in $\mathbb{F}_p$, we can compute the corresponding y-coordinate $\tilde{y} \in \mathbb{F}_p$ through the exponentiation $\tilde{y} = \sqrt{z} = z^{(p+1)/4}$, and hence $\tilde{y}^2 = z$. Conversely, if z is a non-square in $\mathbb{F}_p$, the same exponentiation outputs $\tilde{y} \in \mathbb{F}_p$ such that $\tilde{y}^2 = -z$. Thus, as an alternative to `IsSquare`, we can determine the orientation of the sampled point by computing $z = x^3 + Ax^2 + x$, and the pseudo y-coordinate $\tilde{y} = z^{(p+1)/4}$. If $\tilde{y}^2 = z$, the point has positive orientation, if $\tilde{y}^2 = -z$, negative orientation. If neither of these cases applies, a fault must have occurred during the exponentiation, and we reject the point.

This method may seem equivalent to computing the sign s using `IsSquare`, as it does not verify that z has been computed correctly from x. But having an output value $\tilde{y} \in \mathbb{F}_p$ instead of the `IsSquare` output -1 or 1 allows for a stronger verification step in order to mitigate fault attacks on the point orientation. We present the details of the original CSIDH algorithm including this countermeasure in Algorithm 2.

Algorithm 2 Evaluation of CSIDH group action with countermeasure

Input: $A \in \mathbb{F}_p$ and a list of integers $(e_1, \ldots, e_n)$.
Output: $B \in \mathbb{F}_p$ such that $\prod [\mathfrak{l}_i]^{e_i} * E_A = E_B$

1: **while** some $e_i \neq 0$ **do**
2: Sample a random $x \in \mathbb{F}_p$, defining a point P.
3: Set $z \leftarrow x^3 + Ax^2 + x$, $\tilde{y} \leftarrow z^{(p+1)/4}$.
4: Set $s \leftarrow 1$ if $\tilde{y}^2 = z$, $s \leftarrow -1$ if $\tilde{y}^2 = -z$, $s \leftarrow 0$ otherwise.
5: Let $S = \{i \mid e_i \neq 0,\ \mathrm{sign}(e_i) = s\}$. **Restart** with new x if S is empty.
6: Let $k \leftarrow \prod_{i \in S} \ell_i$ and compute $Q' = (X_{Q'} : Z_{Q'}) \leftarrow [\frac{p+1}{k}]P$.
7: Compute $z' \leftarrow x^3 + Ax^2 + x$.
8: Set $X_Q \leftarrow s \cdot z' \cdot X_{Q'}$, $Z_Q \leftarrow \tilde{y}^2 \cdot Z_{Q'}$ and $Q \leftarrow (X_Q : Z_Q)$.
9: **for each** $i \in S$ **do**
10: Set $k \leftarrow k/\ell_i$ and compute $R \leftarrow [k]Q$. If $R = \infty$, **skip** this i.
11: Compute $\varphi : E_A \to E_B$ with kernel $\langle R \rangle$.
12: Set $A \leftarrow B$, $Q \leftarrow \varphi(Q)$, and $e_i \leftarrow e_i - s$.
13: **return** A.

Steps 3 and 4 of Algorithm 2 contain our proposed method to determine the orientation s without using `IsSquare`. In order to verify the correctness of these computations, we add a verification step. First, we recompute z via $z' = x^3 + Ax^2 + x$, and in case of a correct execution, we have $z = z'$. Thus, we have $s \cdot z' = \tilde{y}^2$, which we can use as verification of the correctness of the computations of s, z, z', and $\tilde{y}$. If this were implemented through a simple check, an attacker might be able to skip this check through fault injection. Hence, we perform the equality check through the multiplications $X_Q = s \cdot z' \cdot X_{Q'}$ and $Z_Q = \tilde{y}^2 \cdot Z_{Q'}$, and initialize $Q = (X_Q : Z_Q)$ only afterwards, in order to prevent an attacker from skipping Step 8. If $s \cdot z' = \tilde{y}^2$ holds as expected, this is merely a change of the projective representation of Q', and thus leaves the point and its order unchanged. However, if $s \cdot z' \neq \tilde{y}^2$, this changes the x-coordinate X_Q/Z_Q of Q to a random value corresponding to a point of different order. If Q does not have the required order before entering the isogeny loop, the isogeny computation will produce random outputs in $\mathbb{F}_p$ that do not represent

supersingular elliptic curves with overwhelming probability. We can either output this random $\mathbb{F}_p$-value, or detect it through a supersingularity check [25, 96] at the end of the algorithm and abort. The attacker gains no information in both cases. The supersingularity check can be replaced by a cheaper procedure: Sampling a random point P and checking if $[p+1]P = \infty$. This is much cheaper and has a very low probability of false positives.

There are several ways in which an attacker may try to circumvent this countermeasure. A simple way to outmaneuver the verification is to perform the *same* fault in the computation of z and z', such that $z = z'$, but $z \neq x^3 + Ax^2 + x$. To mitigate this, we recommend computing z' using a different algorithm and a different sequence of operations, so that there are no simple faults that can be repeated in both computations of z and z' that result in $z = z'$. Faults in the computation of both z and z' then lead to random $\mathbb{F}_p$-values, where the probability of $z = z'$ is $1/p$.

The attacker may still fault the computation of s in Step 4 of Algorithm 2. However, this will now also flip the x-coordinate of Q to $-x$, which in general results in a point of random order, leading to invalid outputs. The only known exception is the curve $E_0 \colon y^2 = x^3 + x$: In this case, flipping the x-coordinate corresponds to a distortion map taking Q to a point of the same order on the quadratic twist. Thus, for E_0, flipping the sign s additionally results in *actually* changing the orientation of Q, so these two errors effectively cancel each other.

PROTECTING ELLIGATOR. Recall from Section 3 that two-point variants of CSIDH, including CTIDH, use the Elligator map for two points simultaneously, which requires an execution of `IsSquare` in order to correctly allocate the sampled points to P_+ and P_-.

We can adapt the pseudo y-coordinate technique from Section 9.2: we determine orientations and verify their correctness by applying this countermeasure for both P_+ and P_- separately. We dub this protected version of the Elligator sampling Elligreator. An additional benefit is that faulting the computations of the x-coordinates of the two points within Elligator [98, Alg. 3] is prevented by Elligreator.

In CTIDH, each round performs two Elligator samplings, and throws away one point respectively. Nevertheless, it is not known a priori which of the two points has the required orientation, so Elligreator needs to check *both* points anyway in order to find the point of correct orientation.

On the one hand, adding dummy computations, in this case sampling points but directly discarding some of them, might lead to different vulnerabilities such as safe-error attacks. On the other hand, sampling both points directly

with Elligreator at the beginning of each round (at the cost of one additional isogeny evaluation) may lead to correlations between the sampled points [20]. It is unclear which approach should be favored.

9.3 IMPLEMENTATION COSTS

Implementing this countermeasure is straightforward. While `IsSquare` requires an exponentiation by $(p-1)/2$, our pseudo y-coordinate approach replaces this exponent by $(p+1)/4$, which leads to roughly the same cost. Furthermore, we require a handful of extra operations for computing z', X_Q, and Z_Q in Steps 7 and 8 of Algorithm 2. For the computation of z' we used a different algorithm than is used for the computation of z, incurring a small additional cost, for the reason discussed above. Therefore, using this countermeasure in a 1-point variant should not be noticeable in terms of performance, since the extra operations are negligible in comparison to the overall cost.

In 2-point variants, we use Elligreator, which requires two exponentiations instead of one as Elligator does. Thus, the countermeasure is expected to add a more significant, yet relatively small overhead in 2-point variants. CTIDH uses two calls to Elligreator per round, and both executions contain two pseudo-y checks respectively. We sketch the cost of our countermeasure in CTIDH-512: [20] reports an exponentiation by $(p-1)/2$ to cost 602 multiplications (including squarings). Since CTIDH-512 requires roughly 20 rounds per run and we add two additional exponentiations by $(p+1)/4$ per round, the overhead is approximately $2 \cdot 20 \cdot 602 = 24080$ multiplications. Ignoring the negligible amount of further multiplications we introduce, relative to the total cost of 438006 multiplications on average for a CTIDH-512 group action, the total overhead of our countermeasure is roughly 5.5%.

VI

PROJECTIVE RADICAL ISOGENIES

At Asiacrypt 2020, Castryck, Decru, and Vercauteren introduced radical isogenies, which require only one torsion point to initiate a chain of isogenies, in comparison to Vélu isogenies which require a torsion point per isogeny. These were implemented in affine coordinates, using non-constant-time techniques. However, in a realistic scenario, a constant-time implementation is necessary to mitigate risks of timing attacks. This question was left as an open problem.

In this chapter, we analyze this problem. A straightforward constant-time implementation of radical isogenies encounters too many issues to be cost effective, but we resolve some of these issues with two new optimization techniques: We introduce projective radical isogenies to save costly inversions, and we present a hybrid strategy for integration of radical isogenies in CSIDH implementations.

These improvements make radical isogenies almost twice as efficient in constant-time, however, our benchmarks show that the speed-up for constant-time CSIDH using radical isogenies is only about 3% for small parameter sets (CSIDH-512) and, as they scale worse than Vélu iogenies, disappears for larger parameter sets (beyond CSIDH-1792).

This chapter is an abridged version of the paper

> Jesús-Javier Chi-Domínguez and Krijn Reijnders. "Fully projective radical isogenies in constant-time". In: *Cryptographers Track at the RSA Conference*. Springer. 2022, pp. 73–95.

Instead of the full article, this chapter only summarizes the main results. This allows for a smooth reading experience of the thesis, while keeping the full length of the thesis no longer than necessary.

§1 Introduction

The results presented by Castryck, Decru, and Vercauteren [92] suggest a speed-up of about 19% when using radical isogenies instead of Vélu's formulas (for CSIDH-512). However, these experiments focused on a non-constant-time Magma implementation for both the group-action evaluation and the chain of radical isogenies. More specifically, the Magma code [92] performs field inversions in variable time depending on the input. Furthermore, the implementation computes exactly $|e_i|$ radical isogenies of degree ℓ_i, where $e_i \in [\![-m_i, \ldots, m_i]\!]$ is a secret exponent of the private key, which is immediately vulnerable to timing attacks. Thus, random non-constant time instances of radical isogenies perform an average $\frac{m_i}{2}$ isogenies per degree, whereas in constant-time implementations we must perform the fixed bound m_i per degree.

A straightforward constant-time implementation of radical isogenies would replace all non-constant-time techniques with constant-time techniques. This would, however, drastically reduce the performance of radical isogenies, as inversions become costly and we need to perform more (dummy) isogenies per degree. Such an implementation would be outperformed by any state-of-the-art CSIDH implementation in constant-time.

Contributions

In this chapter, we are interested in constant-time implementations of CSIDH, using radical isogenies. We summarize

1. *projective radical isogenies*, a non-trivial reformulation of radical isogenies in projective coordinates, and of the required isomorphisms between curve models. This allows us to perform radical isogenies without leaving projective coordinates. This saves an inversion per isogeny and additional inversions in the isomorphisms between curve models, which in total reduces the cost of radical isogenies in constant-time by almost 50%.

2. a *hybrid strategy* to integrate radical isogenies in CSIDH, which allows us to 're-use' torsion points that are used in the CSIDH group action evaluation to initiate a 'chain' of radical isogenies, and to keep cheap low degree Vélu isogenies. This generalizes the 'traditional' CSIDH evaluation, optimizes the evaluation of radical isogenies, and does not require sampling an extra torsion point to initiate a 'chain' of radical isogenies.

3. the *first constant-time implementation and performance benchmarks* using radical isogenies. Our implementation incorporates all state-of-the-art improvements, and thus allows for precise comparison in performance between CSIDH *with* radical isogenies in comparison to 'traditional' CSIDH, in total finite-field operations.[1] This improves the original Magma benchmarks [88, 92] and shows that the 19% speed-up diminishes to roughly 3% in a precise constant-time comparison.

We conclude that in small parameter sets, an implementation using radical isogenies performs roughly 3% better than CSIDH-512. However, for larger parameter sets with prime sizes of 1792 bits or larger, we find that radical isogenies only slow down a CSIDH implementation. The (Python) implementation used in this paper is freely available at

https://github.com/Krijn-math/Constant-time-CSURF-CRADS.

Remark 1. In this chapter, we consider CSURF [88] as a CSIDH implementation using radical isogenies of degree 2. Any mention of "CSIDH with radical isogenies" therefore also incorporates CSURF. Our results thus also show that CSURF is *not* faster for CSIDH-1792 and beyond.

§2 Preliminaries

Chapter II gives an introduction to curves and their isogenies, and an introduction to CSIDH. Here, we only introduce radical isogenies.

THE TATE NORMAL FORM. Any elliptic curve E with a point P of order N is isomorphic to a unique curve $E(b,c)$ over $\mathbb{F}_p$ given by

$$E(b,c): \; y^2 + (1-c)x - by = x^3 - bx^2, \quad b,c \in \mathbb{F}_p. \tag{8}$$

This curve is the Tate Normal Form of the pair (E,P) with the point $(0,0) \in E(b,c)$ the image of $P \in E$, thus of order N.

RADICAL ISOGENIES. Let E_A be a supersingular Montgomery curve over $\mathbb{F}_p$ with a point $P \in E_A$ of order N. Then $\langle P \rangle$ defines an isogeny $\varphi : E_A \to E_{A'}$. Let P' be a point on $E_{A'}$ of order N, with $P' \notin \ker \widehat{\varphi}$. Then (E_A, P) and

1 We explicitly do not focus on performance in clock cycles; a measurement in clock cycles could give the impression that the underlying field arithmetic is optimized, instead of the algorithmic performance.

$E(b', c')$ determine unique Tate normal forms $E(b, c)$ and $(E_{A'}, P')$, respectively. Castryck, Decru, and Vercauteren [92] prove the existence of a function $\varphi_N : (b, c) \mapsto (b', c')$, whose cost is roughly dominated by a single radical computation $\sqrt[N]{\rho}$, for some $\rho \in \mathbb{F}_p(b, c)$. By iteratively applying φ_N, this computes a chain of N-isogenies, without the need to sample new points of order N, as is required in 'classical' CSIDH. As a consequence, after finding the initial point $P \in E_A[N]$ and assuming we know φ_N, we can move from Montgomery models to Tate normal forms to compute a chain of N-isogenies.

APPLICATION IN CSIDH. For $N = \ell_i$, this implies we can compute the action $\mathfrak{l}_i^k \star E_A$ using a single ℓ_i-torsion point, as summarized in the following diagram.

$$\begin{array}{ccccccc} E_A & \xrightarrow{\text{Vélu}} & [\mathfrak{l}_i] \star E_A & \longrightarrow & \cdots & \longrightarrow & [\mathfrak{l}_i^k] \star E_A \\ \Big\downarrow & & & & & & \Big\uparrow \\ \text{To Tate normal form} & & & & & & \text{To Montgomery} \\ \Big\downarrow & & & & & & \Big| \\ E(b_0, c_0) & \xrightarrow[\varphi_N]{} & E(b_1, c_1) & \longrightarrow & \cdots & \longrightarrow & E(b_k, c_k) \end{array}$$

Castryck, Decru, and Vercauteren [92] give formulas for φ_N for all primes from 2 to 13, and additionally $N = 4$ and $n = 9$. These cases $N = 4$ and $N = 9$ are interesting: a *single* radical computation replaces the action of $[\mathfrak{l}_i^2]$ in these cases.

Remark 2. The N-th root of ρ is unique for any N coprime to $p - 1$, as the map $x \mapsto x^N$ is then a bijection. This holds in particular for any $N = \ell_i$ in CSIDH, as ℓ_i divides $p + 1$. The cases $N = 2$ and $N = 4$ require a deterministic choice of root.

§3 Projective Radical Isogenies

The formulas φ_N given in [92] are rational functions over $\mathbb{F}_p(b, c, \sqrt[N]{\rho})$. Although these are easy to give in projective coordinates naively, this would require us to compute the N-th root of a projective representation $(X : Z)$ of ρ, which costs *two* exponentiations $\sqrt[N]{X}$ and $\sqrt[N]{Z}$. The following lemma achieves this at the cost of only *one* exponentiation.

Lemma 1. Let $N \in \mathbb{N}$ coprime to $p - 1$ and let $\alpha \in \mathbb{F}_p$ be represented as $(X : Z)$ in projective coordinates, with $X, Z \in \mathbb{F}_p$. Then $\sqrt[N]{\alpha}$ can be compute as $(\sqrt[N]{XZ^{N-1}} : Z)$.

Proof. The result is non-trivial to find, but trivial once stated: the N-th root of Z^N is unique and thus equal to Z. Hence $(X : Z) = (XZ^{N-1} : Z^N)$ has the N-th root $(\sqrt[N]{XZ^{N-1}} : Z)$. □

Lemma 1 allows us to rewrite the formulas φ_N in projective coordinates for $N = 2, 3, 4, 5, 7, 9$, with some additional technicalities for $N = 2, 4$ due to the common divisor 2 of N and $p - 1$. The savings per degree are showing in the following table.

Table 2: Comparison between affine [92] and projective radical isogenies. The letters **E**, **M**, **S**, **a** and **I** denote exponentiation, multiplication, squaring, addition and inversion respectively. The last column expresses the cost ratio of projective versus affine in terms of $\mathbb{F}_p$-operations for a prime of 512 bits, assuming $\mathbf{S} = \mathbf{M}$ and $\mathbf{E} = \mathbf{I} = \log p \cdot \mathbf{M}$, ignoring **a**.

Degree	Affine [92]	Projective	Speed-up
2	$\mathbf{E} + 4\mathbf{M} + 6\mathbf{a} + \mathbf{I}$	$\mathbf{E} + 3\mathbf{M} + 5\mathbf{S} + 10\mathbf{a}$	50.4%
3	$\mathbf{E} + 6\mathbf{M} + 3\mathbf{a}$	$\mathbf{E} + 2\mathbf{M} + 10\mathbf{a}$	99.3%
4	$\mathbf{E} + 4\mathbf{M} + 3\mathbf{a} + \mathbf{I}$	$\mathbf{E} + 6\mathbf{M} + 4\mathbf{S} + 3\mathbf{a}$	50.5%
5	$\mathbf{E} + 7\mathbf{M} + 6\mathbf{a} + \mathbf{I}$	$\mathbf{E} + 8\mathbf{M} + 6\mathbf{S} + 18\mathbf{a}$	50.7%
7	$\mathbf{E} + 24\mathbf{M} + 20\mathbf{a} + \mathbf{I}$	$\mathbf{E} + 14\mathbf{M} + 4\mathbf{S} + 64\mathbf{a}$	50.5%
9	$\mathbf{E} + 69\mathbf{M} + 58\mathbf{a} + \mathbf{I}$	$\mathbf{E} + 61\mathbf{M} + 10\mathbf{S} + 202\mathbf{a}$	52.1%

Beyond projectivizing the φ_N formulas, we applied the same technique to the isomorphisms between different curve models required to use radical isogenies in CSIDH. These achieve similar speed-ups of roughly 50%. Thus, projective radical isogenies are almost twice as fast as affine radical isogenies in constant-time implementations.

§4 Hybrid Strategy

Our second improvement to radical isogenies is a *hybrid strategy* to evaluate the group action. The standard group action evaluation in CSIDH (see also Algorithm 1 in Chapter V) samples a random point $P \in E(\mathbb{F}_p)$, which is used to perform many isogenies of degree ℓ_i, but only one per degree, by pushing the point itself through all the isogenies generated by it. Thus, in constant-time implementations, we need as many points P as the largest bound m_i from which we sample secret exponents $e_i \in [-m_i, \ldots, m_i]$. Therefore, optimal bounds are relatively close together.

Radical isogenies require only a single point P of order ℓ_i to perform all e_i isogenies. Thus, the original implementation [92] separates these two approaches: Let $R = 2, 3, 5, 7$ denote the degrees for which we use radical isogenies, then for $\ell_i \in R$ chains of radical isogenies of length e_i are computed, and for all other degrees the 'traditional' CSIDH algorithm is used. Wherever we can use these chains, we can easily increase m_i. Thus, we get a skewed box, with large m_i whenever $\ell_i \in R$, and then a rather smooth range of m_i for the degrees with Vélu isogenies.

We observe that for when the 'traditional' CSIDH algorithm is used for all other degrees and we push the point P through all these isogenies, we are left with a point whose order is most-likely divisible by $\ell_i \in R$ This 'left-over' point P can still be used to perform Vélu isogenies of degree $\ell_i \in R$, and these Vélu isogenies are far cheaper than an individual radical isogeny, which cost at least an expontiation. It is therefore beneficial to perform both radical isogenies as well as Vélu isogenies for these degrees, using the 'left-over' points. Furthermore, the chain of radical isogenies, which requires a point of order ℓ_i, can also be initiated from such a left-over point.

HYBRID BOUNDS. We therefore write the bound m_i for those $\ell_i \in R$ per degree as $m_i = m_{i,v} + m_{i,r}$, where $m_{i,v}$ are the number of Vélu isogenies we perform per degree ℓ_i and $m_{i,r}$ the number of radical isogenies. This generalizes both the traditional CSIDH evaluation, taking $m_{i,r} = 0$ for all i, as well as the implementation of [92], taking $m_{i,v} = 0$.

An optimal estimate for $m_{i,v}$ is easy to make: we know the number of points T we require in a constant-time implementation of CSIDH, and we therefore expect $T_i = \frac{\ell_i - 1}{\ell_i} \cdot T$ left-over points to have order divisible by ℓ_i. Thus, for $\ell_i \in R$, we set $m_{i,v} = T_i - 1$ using the remaining last point to initiate the chain of radical isogenies of length $m_{i,r}$. Given $m_{i,v}$ we can then derive a close-to-optimal value for $m_{i,r}$ by computing cost c of performing $m_{i,r}$ isogenies, and the increase in key space b by increasing $m_{i,r}$.

§5 Implementation and Performance Benchmark

We implemented projective radical isogenies for $N = 2, 3, 4, 5, 7, 9$, including projective versions of the required isomorphisms between curve models. We furthermore implemented the hybrid strategy. The whole implementation is implemented with state-of-the-art techniques for constant-time CSIDH, including

efficient addition chains for the required exponentiations for the computation of the radicals $\sqrt[N]{\rho}$ and $\sqrt{\text{élu}}$ or $\ell_i > 101$.

To benchmark the performance of radical isogenies, we focus on two dummy-based constant-time approaches, MCR [274] and OAYT [301], for primes of 512-, 1024-, 2048-, 3072-, and 4096-bits. We compare the performance of **1.)** CSIDH with optimal bounds [213] to **2.)** CSIDH with radical 2/4-isogenies, which we denote CSURF[2], and **3.)** CSIDH with radical 2/4-isogenies and 3/9-isogenies, which we denote CRADS[3].

BENCHMARK RESULTS. Table 3 gives the cost of CSIDH, CSURF and CRADS for primes p of different bitlengths using the two different approaches to constant-time. We find that, although a small speed-up of roughly 2% to 3% can be achieved with radical isogenies for primes of 512 and 1024 bits, this quickly evaporates for larger sizes, as exponentiation, the core cost of a radical isogeny, scales much worse than Vélu isogenies.

Table 3: Benchmark results for 512- to 4096-bit primes. Results are given in millions of finite-field multiplications, averaged over 1024 runs assuming **S** = **M**. Numbers in bold are optimal results for that prime size.

Prime size (bits)	512	1024	1792	2048	3072	4096
CSIDH-OAYT	0.791	0.873	**0.999**	**1.039**	**1.217**	**1.361**
CSURF-OAYT	0.771	0.862	1.000	1.042	1.225	1.387
CRADS-OAYT	**0.765**	**0.861**	1.007	1.050	1.237	1.399
CSIDH-MCR	1.011	1.093	1.218	1.255	1.436	1.580
CSURF-MCR	0.980	1.074	1.211	1.253	1.443	1.594
CRADS-MCR	0.985	1.086	1.228	1.272	1.469	1.625

Remark 3. Very recently, CTIDH [20] introduced a third approach for constant-time CSIDH, achieving an almost twofold speed-up over OAYT-style CSIDH. Integrating radical isogenies into CTIDH seems much more difficult due to the batching strategy used in CTIDH. We therefore believe that the use of radical isogenies in constant-time CSIDH using CTIDH will cause a slow-down. A close analysis of this problem is future work.

2 Although CSURF [88] originally only uses 2-isogenies, it is clear that 4-isogenies will almost always perform better.

3 We do not use radical isogenies for any other degree, as we experimentally observed this gives no speed-up for any prime size.

We conclude that 'horizontally' expanding the degrees ℓ_i for large primes p is more efficient than 'vertically' increasing the bounds m_i using radical isogenies. Nevertheless, for small primes, our hybrid strategy for radical isogenies can speed-up CSIDH and requires only slight changes to a 'traditional' CSIDH implementation. However, before we can practically use such parameter sets, an improved understanding of the quantum security of CSIDH is necessary.

VII

OPTIMIZATIONS AND PRACTICALITY OF HIGH-SECURITY CSIDH

In this chapter, we assess the real-world practicality of CSIDH, an isogeny-based non-interactive key exchange. We provide the first thorough assessment of the practicality of CSIDH in higher parameter sizes for conservative estimates of quantum security, and with protection against physical attacks.

This requires a three-fold analysis of CSIDH. First, we describe two approaches to efficient high-security CSIDH implementations, based on SQALE and CTIDH. Second, we optimize such high-security implementations: on a high level by improving several subroutines, and on a low level by improving the finite-field arithmetic. Third, we benchmark the performance of high-security CSIDH in real-world applications.

As a stand-alone primitive, our implementations outperform previous results by a factor up to 2.53×. As a real-world use case considering network protocols, we use CSIDH in TLS variants that allow early authentication through a NIKE. Although our instantiations of CSIDH have smaller communication requirements than post-quantum KEM and signature schemes, even our highly-optimized implementations result in too-large handshake latency (tens of seconds), showing that CSIDH is only practical in niche cases.

This chapter is based on the paper

> Fabio Campos, Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Michael Meyer, Krijn Reijnders, Francisco Rodríguez-Henríquez Francisco, Peter Schwabe, and Thom Wiggers. "Optimizations and Practicality of High-Security CSIDH". in: *IACR Communications in Cryptology* 1.1 (2024).

I have improved the description of VeriFast by removing implicit assumptions.

§1 Introduction

The commutative isogeny-based key exchange protocol (CSIDH) was proposed by Castryck, Lange, Martindale, Panny, and Renes [95] at Asiacrypt 2018. Although it was proposed too late to be included as a candidate in the NIST post-quantum standardization effort, it has since received significant attention in the field of poast-quantum cryptography.

From a crypto-engineering point of view, this attention can be explained by two unique features of CSIDH: First, with the originally proposed parameters, CSIDH has remarkably small bandwidth requirements. Specifically, CSIDH-512, the parameter set targeting security equivalent to AES-128, needs to transmit only 64 bytes each way, more than 10 times less than Kyber-512, the KEM chosen for standardization by NIST [292]. Second, and more importantly, CSIDH is so far the only realistic option for post-quantum *non-interactive key exchange* (NIKE), meaning it can be used as a post-quantum drop-in replacement for Diffie–Hellman key exchange in protocols that combine ephemeral and static Diffie–Hellman key shares non-interactively. Such protocols include the Signal X3DH handshake [268] and early proposals for TLS 1.3 [245], known as OPTLS. The OPTLS authentication mechanism is still under consideration as an extension [328]. CSIDH is the only post-quantum NIKE that might enable these use cases, except for the recently-proposed Swoosh algorithm [183], which has too large public keys for use in TLS.

Unfortunately, quite soon after CSIDH was proposed, several security analyses called into question the claimed concrete security against quantum attacks achieved by the proposed parameters [60, 103, 307]. The gist of these analyses seems troublesome; Peikert [307] states that *"the cost of CSIDH-512 key recovery is only about* 2^{16} *quantum evaluations using* 2^{40} *bits of quantumly accessible classical memory (plus relatively small other resources)"*. Similarly, Bonnetain and Schrottenloher [60] claim a cost of 2^{19} quantum evaluations for attacking the same instance, and propose a quantum circuit requiring only $2^{52.6}$ T-gates per evaluation, which means the security would still be insufficient. Upon exploring the quantum cost of attacking larger instances but ignoring the cost per CSIDH quantum evaluation, instances may require 2048- to 4096-bit keys to achieve the security level originally claimed by CSIDH-512 [103].

Interestingly, although some of these concerns were raised as early as May 2018, at a time when [95] was only available as a preprint, most research on efficient implementations [20, 97, 273, 300], and phisical against CSIDH [78, 254], also explored in Chapters IV and V, continued to work with the original parameters. This can probably partly be explained by the fact that the software

implementation referenced in [95][1] implements only the smaller two of the three original parameter sets, i.e., CSIDH-512 and CSIDH-1024. However, another reason is that the concerns about the quantum security of CSIDH were, and to some extent still are, subject of debate. Most notably, Bernstein, Lange, Martindale, and Panny [48] point out that one issue with quantum attacks against CSIDH is the rather steep cost of implementing CSIDH on a quantum computer in the first place. They conclude that the cost of each query pushes the total attack cost over 2^{80}.

In this paper, we do not take any position in this ongoing debate but rather set out to answer the question of what it means for the performance and applicability of CSIDH if we choose more conservative parameters and aim for a security standard suitable for real-world application, including protection against physical attacks. We call such instantiations *high-security CSIDH*.

Contributions

The core contribution of this paper is an in-depth assessment of the real-world practicality of CSIDH.[2] On a high level, this assessment is divided into three parts. First, we instantiate CSIDH at high(er) security levels, suitable for real-world applications, and with protection against physical attacks. Second, we optimize the efficiency arithmetic of high-security CSIDH. Third, we test the practicality of high-security CSIDH. We present:

1. Efficient CSIDH instantiations, following two different approaches of implementing high-security CSIDH.
   a) The first approach aims at protection against physical attacks, and is based on SQALE [103]. In this approach, we eliminate randomness requirements and the use of dummy operations in CSIDH by restricting the key space to $\{-1, 1\}^n$, as proposed by [97]. We refer to this deterministic version of CSIDH as dCSIDH.
   b) The second approach optimizes purely for performance and uses the CTIDH batching techniques introduced in [20]. We refer to this variant of CSIDH as CTIDH. In particular, we extend the implementation from [20] to larger parameter sets.

1 Available from https://yx7.cc/code/csidh/csidh-latest.tar.xz
2 All our work follows the "constant-time" paradigm for cryptographic implementations and thus protects against timing attacks by avoiding secret-dependent branch conditions and memory indices.

2. Optimized implementations of dCSIDH and CTIDH.
   a) On a high level, we present faster key validation for large parameters, and add a small number of bits to public keys to improve shared-key generation in dCSIDH.
   b) On a low level, we improve the finite-field arithmetic. Our implementations use curves over large prime fields $\mathbb{F}_p$, where p ranges from 2048 to 9216 bits. We optimize arithmetic in these fields for 64-bit Intel processors, specifically the Skylake microarchitecture, using three different options for the underlying field arithmetic.
      - using the GNU Multiple Precision Library (GMP)
      - using MULX and the schoolbook approach (OpScan)
      - using MULX and the Karatsuba approach (Karatsuba)
3. Practicality benchmarks of dCSIDH and CTIDH.
   a) As a standalone primitive, we benchmark our optimized C/assembly implementations. Our dCSIDH implementation outperforms previous implementations by a factor up to 2.53×. Our CTIDH implementation is the first using large parameters, and, dropping determinism, three times as fast as dCSIDH.
   b) As a real-world use case, we benchmark both dCSIDH and CTIDH in real-world network protocols. We extend the Rustls library [57] to support OPTLS [245]. OPTLS is a variant of the TLS 1.3 handshake that heavily relies on a NIKE for authentication, and avoids handshake signatures. Signatures are especially large, cf. Dilithium [264], or hard to implement, cf. Falcon [316], in the post-quantum setting. We compare the performance of the resulting post-quantum OPTLS to post-quantum KEMTLS [341], an OPTLS-inspired protocol that uses KEMs for authentication to avoid handshake signatures. Our results show that dCSIDH and CTIDH are too slow for general-purpose use: a fully CSIDH-instantiated handshake protocol, though smaller in bandwidth requirements, is orders of magnitude slower than an equivalent based on signatures or KEMs.

Related work.

The impact of the CSIDH proposal on the cryptographic community can be assessed by the many papers that have been produced around this protocol.

Since Castryck, Lange, Martindale, Panny, and Renes [95] left open the problem of implementing CSIDH in constant-time, several papers have proposed different strategies for achieving this property.

The first constant-time implementation of CSIDH, by Bernstein, Lange, Martindale, and Panny [48], focused on assessing the quantum security level provided by CSIDH. For this purpose, the authors strive to produce not only a constant-time CSIDH instantiation but also a randomness-free implementation. Meyer, Campos, and Reith [273] (see also [275]) present a more efficient constant-time instantiation of CSIDH for practical purposes. They introduce several algorithmic techniques, including SIMBA and sampling secret keys from varying intervals, which are further improved by Onuki, Aikawa, Yamazaki, and Takagi [300]. Moreover, Moriya, Onuki, and Takagi [284], and Cervantes-Vázquez *et al.* [97], perform more efficient CSIDH isogeny computations using the twisted Edwards model of elliptic curves. The authors of [97] propose a more computationally demanding dummy-free variant of CSIDH. In exchange, this variant is arguably better suited to resist physical attacks from stronger adversaries, such as fault attacks.

A second wave of studies around CSIDH improve several crucial building blocks. First, a framework that adapts optimal strategies à la SIDH/SIKE to the context of CSIDH [107, 213]. Second, improving the computation of large-degree isogenies using an improved version of Vélu's formulas known as $\sqrt{\text{élu}}$ [44]. Several later variants of CSIDH [3, 20] use these improvements. Other works [87, 90] explore even more variants of CSIDH.

The introduction of CTIDH [20] achieved a breakthrough in the performance of constant-time CSIDH, resulting in an almost twofold speedup, using a new key space and accompanying constant-time algorithms to exploit the idea of batching isogeny degrees. However, they restrict their performance evaluation to 512- and 1024-bit primes. In contrast, SQALE [103] is the first CSIDH implementation at higher security levels, using 2000- to 9000-bit primes. The software we present here starts from their analysis and parameter sizes to reach NIST Security Level I (equivalent AES-128) and Level III (equivalent AES-192) under different assumptions about the efficiency of quantum attacks, yet goes much further in optimizing the parameters and implementation techniques than [103].

CSIDH is not the only attempt at building a post-quantum NIKE. Although the SIDH protocol [131, 222] was known to be insecure in the static-static scenario [189], Azarderakhsh, Jao, and Leonardi [16] suggested that a NIKE can still be obtained at the cost of many parallel executions of SIDH. However, recent attacks [86, 266, 329] completely break SIDH/SIKE, making this path to a

NIKE unfeasible. The only post-quantum NIKE *not* based on isogenies is based on (R/M)LWE and, according to Lyubashevsky, goes back to "folkore" [263]. In 2018, de Kock [152] first analyzed such a NIKE and the recently proposed Swoosh [183] is a more concrete instantiation of this approach. We discuss differences between CSIDH and Swoosh in more detail in Section 7.

Availability of software.

We place our CSIDH software into the public domain (CC0). All software described in this paper and all measurement data from the TLS experiments are available at

https://github.com/kemtls-secsidh/code.

Organization of this paper.

Section 2 presents the necessary background on isogeny-based cryptography and introduces CSIDH and its CTIDH instantiation. Section 3 explains how we instantiate dCSIDH and CTIDH and choose parameters for our optimized implementations. Section 4 introduces algorithmic optimizations that apply to our instantiations of dCSIDH and CTIDH. Section 5 details our optimization techniques for finite field arithmetic, in particular the efficient Karatsuba-based field arithmetic, and presents benchmarking results for the group action evaluation for dCSIDH and CTIDH. Section 6 describes our integration of dCSIDH and CTIDH into OPTLS and presents handshake performance results. Finally, Section 7 concludes the paper and sketches directions for future work.

§2 Preliminaries

Chapter II gives a general introduction to cryptographic primitives, including NIKEs (Definition II.1) and KEMs (Definition II.2) and Chapter II gives an introduction to curves and their isogenies, and a thorough introduction to CSIDH, and its variants SQALE and CTIDH. Thus, we simply recall that curves in CSIDH are represented as Montgomery curves, so E_A refers to the curve $y^2 = x^3 + Ax^2 + x$ with $A \in \mathbb{F}_p$, and move on with our improvements.

§3 Two Novel Instantiations of CSIDH

In this section, we describe how to instantiate and choose parameters for large-parameter CSIDH. We describe two different approaches to selecting parameters: a deterministic and dummy-free implementation[3] denoted dCSIDH, and an implementation optimizing the batching strategies proposed in [20], neither deterministic nor dummy-free, denoted CTIDH. This reflects the two extreme choices we can make between speed and security against physical attacks, with a requirement for constant-time behaviour seen as a bare minimum for any cryptographic implementation. We note that there are several choices in the middle ground, trading off physical security for speed. For comparability, both approaches are instantiated with the same primes p and prime fields $\mathbb{F}_p$, which we detail in Section 3.1.

3.1 THE CHOICE OF p

In this work, we take the conservative parameter suggestions from [103] at face value. In particular, we consider 2048- and 4096-bit primes for NIST Security Level I, 5120- and 6144-bit primes for NIST Security Level II, and 8192- and 9216-bit primes for NIST Security Level III. Each pair of bitsizes represents a choice between more "aggressive" assumptions, with attacker circuit depth bounded by 2^{60} and more "conservative" assumptions, with attacker circuit depth bounded by 2^{80}. As stressed in [103], this choice of parameters does not take into account the cost of calls to the CSIDH evaluation oracle on a quantum computer and is likely to underestimate security. However, our focus is to simply give performance results for these conservative parameters.

All our implementations use primes of the form $p = f \cdot \prod_{i=1}^{n} \ell_i - 1$, where ℓ_i are distinct odd primes, f is a large power of 2 and n denotes the number of such ℓ_i dividing $p+1$. For these sizes of p, it becomes natural to pick secret key exponents $e_i \in \{-1, +1\}$, as n can be chosen large enough to reach the desired key space size [97, 103]. In particular, to achieve a key space of b bits in CSIDH, we need to have at least $n = b$ primes ℓ_i dividing $p+1$.

For conservative instances, we base the key-space size on the classical meet-in-the-middle (MITM) attack considered in [95], requiring $b = 2\lambda$ for security

3 Our implementation does not take the recent physical attacks from Chapters IV and V into account, whose impact in the high-parameter range is future work. Heuristically, countermeasures against both attacks should not impact performance by much.

parameter λ. That is, $b = 256,\ 256,\ 384$ for p4096, p6144, p9216, respectively[4]. On the other hand, for aggressive instances we base the key-space size on the limited-memory van Oorschot-Wiener golden collision search [371] with the assumptions from [103], which leads to $b = 221,\ 234,\ 332$ for p2048, p5120, p8192, respectively.

Finally, we prefer cofactors f of the form $2^{64 \cdot k}$, since the arithmetic improvements discussed in Section 5 are optimized for such a shape. Hence, to find optimal primes for our implementation, we let $\ell_1, \ldots, \ell_b$ be the b smallest odd primes and then compute the cofactor f as the largest power of 2^{64} that fits in the leftover bitlength. This still leaves us with a bitlength slightly smaller than the target, and hence the leftover bits can be used to search for additional factors ℓ_i (ensuring $n > b$) so that $p = f \cdot \prod_{i=1}^{n} \ell_i - 1$ is prime. These extra factors go unused for dCSIDH, where they are viewed as part of the cofactor, but are exploited by the batching strategies of CTIDH to increase performance. We set a minimum requirement of 5 additional ℓ_i factors (that is, $n \geq b + 5$), decreasing f by a single factor of 2^{64} when not enough bits are left over. The results of this search are shown in Table 4.

Table 4: Parameters for reconstructing each prime $p = f \cdot \prod_{i=1}^{n} \ell_i - 1$. In each case the ℓ_i are assumed to be the first n odd primes, excluding the primes in the fourth column and including the primes in the fifth column, to ensure that p is prime. The last column indicates the NIST Security Level, assuming aggresive (A) or conservative (C) choices.

Prime bits	f	n	Excluded	Included	Key Space	NIST
p2048	2^{64}	226	1361	–	2^{221}	I (A)
p4096	2^{1728}	262	347	1699	2^{256}	I (C)
p5120	2^{2944}	244	227	1601	2^{234}	II (A)
p6144	2^{3776}	262	283	1693, 1697, 1741	2^{256}	II (C)
p8192	2^{4992}	338	401	2287, 2377	2^{332}	III (A)
p9216	2^{5440}	389	179	2689, 2719	2^{384}	III (A)

4 Since conservative instances do not take memory restrictions into account, NIST Security Level I and II have the same key space requirements. They only differ in the size of the prime due to quantum security concerns [103].

3.2 THE ROLE OF FULL-TORSION POINTS IN DCSIDH

The restriction of exponents e_i to $\{-1, +1\}$ allows us to easily make dCSIDH deterministic and dummy free [97, 103], as we always perform only one isogeny of each degree, with the only variable being the "direction" of each isogeny. Since isogenies in either direction require exactly the same operations, it is easy to obtain a constant-time implementation without using dummy operations.

Randomness appears in the traditional CSIDH implementation from the fact that performing isogenies of degree ℓ_i requires a point of order ℓ_i as input, and such a point is obtained by sampling random points on the current curve. Any random point can either be used for "positive" steps $\mathfrak{l}_i^{+1}$ or "negative" steps $\mathfrak{l}_i^{-1}$. Hence, a point of order ℓ_i can be used only once and only for a specific orientation. Therefore, with $|e_i| > 1$ we need to compute more than one isogeny degree ℓ_i, requiring us to sample new points midway. By restricting e_i to $\{-1, +1\}$, we compute only one isogeny per degree ℓ_i and completely avoid sampling random points. Instead, we provide a pair of points $T_+ \in E(\mathbb{F}_p)$ and $T_- \in E^T(\mathbb{F}_p)$ *beforehand* whose orders are divisible by all ℓ_i that we use in the computation of the group action. Thus, T_+ can be used for the positive steps $\mathfrak{l}_i$ with $e_i = 1$, and T_- for the negative steps $\mathfrak{l}_i^{-1}$, with $e_i = -1$. We refer to such points as *full-torsion* points[5], as they allow us to perform an isogeny of every degree ℓ_i when multiplied by the right scalar: To perform an ℓ_i-isogeny in the "plus" direction, we multiple T_+ by $[\frac{p+1}{\ell_i}]$ to obtain a point of order ℓ_i.

Since the probability for the order of a random point to have order divisible by ℓ_i is given by $\frac{\ell_i - 1}{\ell_i}$, sampling a (pair of) full-torsion point(s) can be expensive when small factors ℓ_i are used, as they dominate the probability $\prod_i \frac{\ell_i - 1}{\ell_i}$ of sampling a full-torsion point. Since the primes we use always have additional ℓ_i factors that are unused in dCSIDH (see Section 3.1), we make point sampling more efficient by always discarding the smallest primes rather than the largest ones, increasing the odds to sample a full-torsion point. For example, the prime p4096 has 262 ℓ_i factors but only needs a key space of 2^{256}, hence we can discard 6 primes. By discarding the 6 *smallest* ones, the probability to sample a full-torsion point goes up from $\prod_{i=1}^{256} \frac{\ell_i - 1}{\ell_i} \approx 0.151$ to $\prod_{i=7}^{262} \frac{\ell_i - 1}{\ell_i} \approx 0.418$, making it more than 2.7 times as easy to sample full-torsion points. Such a shift in primes causes a trade-off in the rest of the protocol, as higher-degree isogenies are more expensive. However, with recent improvements [44], the extra cost of using $\ell_{257}, \ldots, \ell_{262}$ instead of $\ell_1, \ldots, \ell_6$ is relatively small in comparison to

5 This is abuse of terminology, as these points do not have the maximal order $p + 1$. We stress that they do not require to have order divible by small primes ℓ_i that are not used.

the total cost of a group action computation. Thus, discarding the smallest ℓ_i is preferable as it significantly decreases the cost of sampling full-torsion points, and only increases the cost of computing $\mathfrak{a} * E$ by a marginal amount.

The points T_+, T_- on the starting curve E_0 can be precomputed and considered public parameters, but for the public-key curves they must be computed in real time. We include the computation of these points in the key generation, and include them in the public key, which makes the shared-secret derivation completely constant-time and deterministic. The key generation is then the only part that does not run in strictly constant wall-clock time, although implemented in constant-time, but still deterministic by sampling points in a pre-defined order. As we describe in Section 4, these points can be represented in a very compact form, which increases public-key sizes by only a few bits. We further emphasize that to avoid active attacks the shared-key computation must validate these transmitted points to be full-torsion points, at non-trivial cost.

3.3 PARAMETERS FOR CTIDH

As mentioned above, our implementation of dCSIDH is dummy-free and deterministic, in order to avoid potential issues with randomness and dummy operations. However, these choices induce significant computational overhead. Therefore, we also provide performance results for CTIDH, based on [20], the fastest available constant-time implementation of CSIDH when allowing randomness and dummy operations. The implementation of [20] only reports performance results for 512-bit and 1024-bit primes. We, however, implement CTIDH a the same (conservative) security levels as dCSIDH which allows us to compare the performance and learn the impact of requiring deterministic and dummy-free behaviour.

For the parameter sizes considered in this work, we thus use the same primes as we use for dCSIDH (see Table 4) and optimize the same finite-field arithmetic for both implementations (see Section 5). We acknowledge it is still unclear which parameters are optimal for CTIDH for such large primes: A larger number of small primes ℓ_i in the factorization of $p + 1$ can be beneficial, since the combinatorial advantage of CTIDH batching increases with the number of available prime degrees. On the other hand, this means including larger primes ℓ_i, and thus more expensive large degree isogenies. Furthermore, the choice of CTIDH parameters, i.e., batches and norm bounds, becomes more challenging at larger prime sizes. We thus leave a full exploration of optimal CTIDH parameters for large primes as future work.

For the given primes, we use the greedy algorithm from [20] to determine these additional parameters, adapted to the case of the cofactor $f > 4$. On input of the primes ℓ_i and a fixed number of batches, the algorithm searches for a locally optimal way to batch these primes, with norm bounds so that the expected number of field multiplications per group action evaluation is minimized. However, for the parameter sizes in this work, the greedy search becomes too inefficient: We can only run searches for a small set of potential batch numbers, especially for the largest parameters. We obtained these potential inputs by extrapolating data from smaller parameters, given in [20] and slightly larger parameters. For concrete parameter choices, we refer to our software. We leave a full exploration of optimal batches, and strategies to find such batches, as future work.

Apart from the parameters and batching setup, our CTIDH implementation uses the algorithms and strategies from [20]. We remark that CTIDH could *in theory* be implemented without dummy isogenies: [20] presents such an algorithm, but points out that the Matryoshka isogenies *internally* still require dummy operations. Thus, current techniques do not allow for a fully dummy-free implementation of CTIDH. Further, the design of a deterministic variant of CTIDH requires some adaptions, such as computing multiple isogenies per batch in a single round. We leave the design and analysis of such an implementation for future work.

§4 Optimizing dCSIDH and CTIDH

Given the parameter choices from Section 3, this section describes the high-level optimizations we apply for dCSIDH and CTIDH. We explain an improved public-key validation algorithm for large parameters, and present several improvements for dCSIDH. For CTIDH, we extend the implementation from [20] to the parameter sizes from Section 3.3.

4.1 Supersingularity verification

For the primes discussed in Section 3.1, we need to adapt supersingularity verification [95, Alg. 1 and Alg. 3], as both algorithms do not work for primes with cofactor $\log f > \frac{1}{2}\log p$ to test supersingularity of a public key E_A.

At its core, both tests verify whether $\#E_A(\mathbb{F}_p) = p+1$ by showing that there is a point P with large enough order $N \mid p+1$. Both algorithms start by sampling a random point P, followed by a multiplication by the cofactor $P \leftarrow [f]P$. These

algorithms then check if the resulting point has ℓ_i-torsion, by calculating if $Q \leftarrow [\frac{\prod \ell_j}{\ell_i}]P \neq \mathcal{O}$ and $[\ell_i]Q = \mathcal{O}$. If the random point P has ℓ_i-torsion for enough ℓ_i such that their product $\prod \ell_i \geq 4\sqrt{p}$, then in the Hasse interval $p+1-2\sqrt{p} \leq \#E_A(\mathbb{F}_p) \leq p+1+2\sqrt{p}$, the only possible multiple of the order of P is $p+1$ and thus $\#E_A(\mathbb{F}_p) = p+1$. Unfortunately, this approach cannot be applied to our setting, because for primes where $\log f > \frac{1}{2}\log p$, even a point with ℓ_i-torsion for all i used in the group action does not reach the threshold $4\sqrt{p}$, as our cofactors f are so large that $\log(\prod \ell_i) = \log p - \log f \leq \frac{1}{2}\log p$.

Luckily, for primes with a large cofactor $f = 2^k$, we can improve such supersingularity algorithms by using precisely this cofactor: Instead of verifying that the order of a random point P has enough ℓ_i-torsion, we verify P has 2^k-torsion. When $\log f = k > \frac{1}{2}\log p$, verifying that P has 2^k-torsion similarly implies that $p+1$ is the only multiple of the order of P in the Hasse interval, and thus E_A must be supersingular by the same logic as above. We name this approach to verify supersingularity VeriFast, given in Algorithm 3.

Algorithm 3 VeriFast: Supersingularity verification for primes with cofactor $2^k > 4\sqrt{p}$.

Input: $A \in \mathbb{F}_p$ defining a curve E_A
Output: true or false, verifying the supersingularity of E_A

$u \xleftarrow{\$} \mathbb{F}_p$, $x_P \leftarrow -u^2$ until $x_P^2 + Ax_P + 1$ is non-square
$v_2 \leftarrow 1$
$x_P \leftarrow [\frac{p+1}{f}]x_P$
while $x_P \neq \mathcal{O}$ and $v_2 < k$ **do**
    $x_P \leftarrow \text{xDBL}(x_P)$, $v_2 \leftarrow v_2 + 1$
if $v_2 > 1 + \log p/2$ and $x_P = \mathcal{O}$ **then**
    return true
return false

VeriFast can be performed deterministically or probabilistically: For Montgomery curves E_A, we can sample P directly from $E_A(\mathbb{F}_p) \setminus [2]E_A$ by picking a point with rational non-square x-coordinate [129]. When we work on the floor, this ensures we always sample a point P with maximum 2^k-torsion, and this removes the probabilistic aspect of the supersingularity verification. A similar improvement can be given for the surface, with minor changes. OtheIn our version, any random point is likely to have v_2 close to k, and hence still verifies supersingularity if the cofactor is a few bits larger than $4\sqrt{p}$. We simply pick $x_P = 2 \in \mathbb{F}_p$, hence $P = (2, -)$, for all supersingularity checks, which has

the advantage that multiplication by 2 can be performed as a simple addition, and hence, $x_P = 2$ optimizes the low-level arithmetic in the computation of $x_P \leftarrow [\frac{p+1}{f}]x_P$. Furthermore, the bound $4\sqrt{p}$ can be improved to $2\sqrt{p}$ as this still implies $p+1$ is the only multiple in the Hasse interval. VeriFast is faster than any of the analyzed algorithms in [24] and Chapter VIII, with a cost of $\mathcal{O}(\log p)$. More specifically, it requires a scalar multiplication by a scalar of $\log p - k$ bits and (at most) k point doublings, where $f = 2^k$ is the cofactor. In comparison to Doliskani's test [24, 161], also of complexity $\mathcal{O}(\log p)$, we have the advantage that we can stay over $\mathbb{F}_p$. The condition that $k > 1 + \frac{1}{2}\log p$ holds for our primes p5120 and beyond. More importantly, even with the probabilistic approach, for these primes the probability to sample a point that does *not* have large enough 2^z-torsion is lower than 2^{-256}. For the primes where $k \leq 1 + \frac{1}{2}\log p$, we can still use the 2^k-torsion, as in VeriFast, but we are required to also verify some ℓ_i-torsion to cross the bound $2\sqrt{p}$. A comparison of performance between VeriFast and previous methods is given in Table 5, showing VeriFast is 28 to 38 times faster for large primes. The hybrid method for $k \leq 1 + \frac{1}{2}\log p$ still achieves a significant speedup.

Table 5: Benchmarking results for supersingularity verification using VeriFast for primes with cofactor $k > \frac{1}{2}\log p$, and hybrid (marked with *) when $k < \frac{1}{2}\log p$. Results of [103] added for comparison. Numbers are median clock cycles (in gigacycles) of 1024 runs on a Skylake CPU.

	p2048*	**p4096***	**p5120**	**p6144**	**p8192**	**p9216**
VeriFast	0.16	0.50	0.53	0.81	1.88	2.54
SQALE [103]	0.30	1.86	14.90	27.65	67.79	96.99

4.2 OPTIMIZED DCSIDH PUBLIC KEYS

As described in Section 3.2, dCSIDH is dummy-free and deterministic by using secret key exponents $e_i \in \{-1, 1\}$, and public keys of the form (A, T_+, T_-), where T_+ and T_- are full-torsion points used to perform positive steps $\mathfrak{l}_i^{+1}$ and negative steps $\mathfrak{l}_i^{-1}$ respectively. For sampling suitable points T_+ and T_- for public keys during key generation, we use the Elligator map $(A, u) \mapsto (T_+, T_-)$ from [97], with Montgomery parameter $A \in \mathbb{F}_p$ and an Elligator seed $u \in \mathbb{F}_p$. The output of Elligator is exactly such a pair of points T'_+ and T'_-, although they might not be *full-torsion*, that is, their respective orders might not be divisible by *all* ℓ_i used in the group action.

Let P be either T_+ or T_-. To efficiently determine if P is a full-torsion point, we follow the usual product-tree approach that was also applied for supersingularity verification in [95] to verify $\left[\frac{p+1}{\ell_i}\right] P \neq \mathcal{O}$ for each ℓ_i. In order to obtain a deterministic algorithm, we try Elligator seeds from a pre-defined sequence $(u_1, u_2, \ldots)$ until we find full-torsion points T_+ and T_-. To determine which of the points $T_\pm$ is T_+ resp. T_-, Elligator requires a Legendre symbol computation. In the case of our proposed dCSIDH configuration with public inputs A and u, we can use a fast non-constant-time algorithm for the Legendre symbol computation as the one presented by Hamburg [206].

Thus, a dCSIDH public key consists of an affine Montgomery coefficient $A \in \mathbb{F}_p$, and an Elligator seed $u \in \mathbb{F}_p$ such that $\mathsf{elligator}(A, u)$ returns two full-torsion points T_+ and T_- on E_A. We choose small fixed values for u to get a public key (A, u) of only $\log_2(p) + \varepsilon$ bits for small $\varepsilon > 0$.

Finally, a user has to verify such a public key (A, u). For A, we verify E_A is supersingular as described in Section 4.2. For u, we verify that it generates two full-torsion points T_+ and T_-, by ensuring at the computation of each step $\mathfrak{l}_i^{\pm 1} * E$ that the correct multiple of *both* T_+ and T_- are not the point at infinity (i.e., both have order ℓ_i) regardless of which point we use to compute the step.[6]

Remark 1. An alternative to finding and including an Elligator seed $u \in \mathbb{F}_p$ in the public key is to find and include small x-coordinates x_+ and x_- that define full-torsion points $T_+ = (x_+, -)$ and $T_- = (x_-, -)$. Information-theoretically, u and the pair (x_+, x_-) share similar probabilities to generate full-torsion points and hence their bitlengths should be comparable. One advantage of x_+ and x_- is that they can be found individually, which should speed up their search. We choose, however, the more succinct approach using u and Elligator.

§5 Implementation

In this section, we describe the optimization steps at the level of field arithmetic to speed up both variants of CSIDH we consider. First and foremost, to enable a fair comparison, we implement a common code base for dCSIDH and CTIDH. Besides sharing the same field arithmetic, both instantiations of CSIDH share all the underlying functions required for computing the group action. However, some required parameters and the strategy within the group action strongly differ between dCSIDH and CTIDH. In the case of dCSIDH, the group action strategy and all the required parameters are based on the implementation

6 Later, in Chapter VIII, we explore improvements to verifying full-torsion points.

provided by [103]. In the case of CTIDH, we generate the batching and other parameters using the methods provided by [20].

5.1 LOW-LEVEL APPROACHES FOR THE FIELD ARITHMETIC LAYER

For the underlying field arithmetic, we implement three different approaches, named **GMP**, **OpScan**, and **Karatsuba** They all share the representation of integers in radix 2^{64} and use Montgomery arithmetic for efficient reductions modulo p.

1. **GMP:** To establish a performance baseline, our first method uses the low-level functions for cryptography ($\mathrm{mpn\text{“}sec\text{“}}$) of the GNU Multiple Precision Arithmetic Library (GMP). Modular multiplication uses a combination of $\mathrm{mpn\text{“}sec\text{“}mul}$ and $\mathrm{mpn\text{“}add\text{“}n}$ to implement Montgomery multiplication, i.e., interleaving multiplication with reduction.

2. **OpScan:** The second approach extends the optimized arithmetic from [95], using the MULX instruction, going from 512-bit and 1024-bit integers to the larger sizes we consider in this paper. Here, we also interleave multiplication with reduction; we generate code for all field sizes from a Python script.

3. **Karatsuba:** Our third strategy uses Karatsuba multiplication [233] together with the MULX optimizations used in our second approach. We describe this strategy, and in particular an optimized reduction for primes of 5120 bits and above, in more detail in Section 5.2.

We follow the earlier optimization efforts [20, 95, 103] for CSIDH and focus on optimizing our code primarily on Intel's Skylake microarchitecture. More specifically, we perform all benchmarks on one core of an Intel Core E3-1260L (Skylake) CPU with hyperthreading and TurboBoost disabled. An overview of (modular) multiplication performance of the three approaches for the different field sizes is given in Table 6. In the following, we will focus on describing the fastest of the three strategies mentioned above, i.e., **Karatsuba**, in more detail.

5.2 OPTIMIZED FIELD ARITHMETIC USING MULX AND KARATSUBA

We present scripts to generate optimized code using the **Karatsuba** approach, based on the **OpScan** approach. More precisely, compared to the **OpScan** approach, we achieve speedups for multiplication, squaring, and reduction.

Table 6: Benchmarking results for multiplication and reduction. Numbers are median clock cycles of 100 000 runs on a Skylake CPU. For the **OpScan** and the **GMP** approaches, we can only provide clock cycles for multiplication (mul) including reduction (red), due to the interleaved Montgomery reduction.

Prime	GMP	OpScan	Karatsuba		
	mul + red	mul + red	mul	red	mul + red
p2048	8662	4538	1442	2648	4090
p4096	34 030	20 318	4981	9777	14 758
p5120	51 671	33 676	8601	6528	15 129
p6144	74 338	53 746	10 210	9517	19 727
p8192	131 858	92 793	17 073	17 295	34 268
p9216	168 375	118 302	20 248	19 709	39 957

MULTIPLICATION. The implementation of **Karatsuba** follows careful considerations to optimize performance. To improve efficiency, we select a breakout level into a MULX-based schoolbook multiplication with a maximum of 9×9 limbs. By choosing this threshold, the implementation aims to strike a balance between utilizing the benefits of Karatsuba's divide-and-conquer strategy [233] and minimizing the overhead of stack operations. This leads to the following number of layers of Karatsuba: $2, 3, 4, 4, 4$, and 4 for the cases p2048, p4096, p5120, p6144, p8192, and p9216, respectively. To further enhance the speed of the implementation, the assembly code avoids function calls. By generating the assembly code dynamically, the implementation can adapt to different prime sizes and adjust the multiplication algorithm accordingly.

SQUARING. For squaring, we take advantage of the fact that some partial products ($a_i a_j$ such that $i \neq j$) only need to be calculated once, and are then accumulated/used twice. On the lowest level of **Karatsuba**, where the schoolbook multiplication takes place, we implement a squaring function with the corresponding savings based on the *lazy doubling* method [253] by adapting the assembly code of the squaring function of the GMP library. For a given n, the implemented method achieves the lower bound of $\frac{n^2-n}{2} + n$ required multiplications. Furthermore, we save additions on the higher levels of **Karatsuba** by reusing calculated values. However, as shown in Table 7, due to the chosen breakout into schoolbook multiplication and the number of available registers, the effort for dealing with the carry chains only leads to a maximum speedup of 17%. Adding a layer of Karatsuba to reduce the number of limbs for the

schoolbook multiplication leads to a speedup at this level. Overall, however, extra layers negate speed-ups gained from reducing limbs.

Table 7: Benchmarking results for multiplication (Mul.) and squaring (Sq.) for the **Karatsuba** approach. Numbers are median clock cycles of 100000 runs on a Skylake CPU.

Prime	p2048	p4096	p5120	p6144	p8192	p9216
Mul.	1442	4981	8601	10 210	17 073	20 248
Sq.	1230	4431	7990	9120	15 050	19 197

Montgomery reduction.

For the cases $p \in \{\text{p5120}, \text{p6144}, \text{p8192}, \text{p9216}\}$, the reduction is calculated according to the *intermediate Montgomery reduction* [17], as we use Montgomery-friendly primes of the form $p = f \cdot \prod_{i=1}^{n} \ell_i - 1$ with cofactor $f = 2^{e_2}$ where $e_2 \geq \log_2(p)/2$ (see Table 4).

As shown in Algorithm 4, the basic idea of this reduction is to perform two Montgomery-reduction steps modulo 2^{e_2} instead of n steps modulo 2^w as in the standard Montgomery reduction. Based on this reduction approach, we can further apply the available Karatsuba-based multiplication when calculating $q_0 \times \alpha$ and $q_1 \times \alpha$ (see Line 2 and 4 in Algorithm 4), leading to further speedups.

Algorithm 4 Intermediate Montgomery reduction for $p = 2^{e_2} \cdot \alpha - 1$ with $e_2 \geq \log_2(p)/2$

Input: $0 \leq a < 2^e p$
Output: $r = a2^{-2e_2} \mod p$ and $0 \leq r < p$
1: $q_0 \leftarrow a \mod 2^{e_2}$
2: $r_0 \leftarrow (a - q_0)/2^{e_2} + q_0 \times \alpha$ ▷ 1st reduction
3: $q_1 \leftarrow r_0 \mod 2^{e_2}$
4: $r \leftarrow (r_0 - q_1)/2^{e_2} + q_1 \times \alpha$ ▷ 2nd reduction
5: $r' \leftarrow r - p + 2^e$
6: **if** $r' \geq 2^e$ **then**
7: $\quad r \leftarrow r' \mod 2^e$
8: **return** r

For the cases $p \in \{\text{p2048}, \text{p4096}\}$, the respective primes do not, and cannot, fulfill the described requirements. Hence, we implement the *word version of*

the Montgomery reduction [17] for these cases. The complexity of Algorithm 5 is dominated by multiplications by α in Line 4. Compared to the standard Montgomery reduction, this approach reduces the number of limbs to be multiplied depending on the value of e_2. We show the results for the corresponding reduction in Table 6.

Algorithm 5 Word version of the Montgomery reduction if $p = 2^{e_2}\alpha - 1$

Input: $0 \leq a < p\beta^n$
Output: $r = a\beta^{-n} \mod p$ and $0 \leq r < p$
1: $r \leftarrow a$
2: **for** $i = 0$ **to** $n - 1$ **do**
3: $\quad r_0 \leftarrow r \mod \beta$
4: $\quad r \leftarrow (r - r_0)/\beta + r_0 \times \alpha 2^{e_2 - w}$
5: $r' \leftarrow r + (\beta^n - p)$
6: **if** $r' \geq \beta^n$ **then**
7: $\quad r \leftarrow r' - \beta$
8: **return** r

5.3 PERFORMANCE RESULTS

We demonstrate the performance increase due to the high-level improvements from Section 4 and the low-level improvements from Section 5.2 for dCSIDH and CTIDH in Table 8. We compare our results to [103], the only other available implementation of CSIDH for similar parameters listing performance numbers. For parameter sizes above p5120, our implementation of dCSIDH is between 55% and 60% faster than SQALE (dummy-free), and CTIDH consistently achieves a speed-up of almost 75% compared to SQALE (OAYT). This excludes the significant speedup from VeriFast, as shown in Table 5.

Longa [259] proposes a novel approach for the computation of sums of products over large prime fields achieving a significant performance impact. However, since the primes in our work support very fast reductions, applying the approach from [259] would not gain a significant advantage. A straight performance comparison is unfortunately rather difficult due to the different underlying fields.

Table 8: Benchmarks for a single group action evaluation for dCSIDH and CTIDH, excluding key validation. SQALE denotes the dummy-free version of [103] and SQALE* denotes the non-deterministic version using dummy isogenies and OAYT-style [301] evaluation. Numbers are median clock cycles (in gigacycles) of 1024 executions on a Skylake CPU.

	p2048	**p4096**	**p5120**	**p6144**	**p8192**	**p9216**
dCSIDH	7.48	34.64	31.80	47.47	127.57	219.09
SQALE	—	39.35	73.57	117.57	322.57	475.64
CTIDH	2.21	11.11	11.26	17.13	43.65	68.78
SQALE*	—	23.21	44.56	74.88	199.15	292.41

§6 Non-Interactive Key Exchange in Protocols

Diffie–Hellman key exchange, as displayed in Figure I.3, is probably the most well-known example of a NIKE protocol, even if it is often used as a "simple" interactive key exchange. One such example is TLS, where ephemeral Diffie–Hellman key exchange is authenticated via a signature. This key exchange can be replaced with a KEM, as shown in [65]. Experiments by Google and Cloudflare [70, 249, 252] used the same approach.

However, in two scenarios the inherently interactive character of a KEM creates issues for protocol designers. When used with long-term keys, a NIKE allows a user Alice to send an authenticated ciphertext to an *offline* user Bob. Signal's X3DH handshake [268] is a notable example using this feature of NIKEs. Indeed, [72] shows that a naive replacement of the Diffie–Hellman operations by KEMs does not work.

In the early stages of the development of TLS 1.3, Krawczyk and Wee proposed OPTLS [245], a variant that uses Diffie–Hellman key exchange not only for ephemeral key exchange, but also for authentication. Many elements of this proposal, made it into the eventual RFC8446 [327]. Though the standard reverted to handshake signatures, the idea lives on in an Internet Draft [328].

As Kuhnen pointed out, OPTLS does use the non-interactive property of Diffie–Hellman [246]. As part of the ephemeral key exchange, the client sends their ephemeral Diffie–Hellman public key. For authentication, the server takes this ephemeral key share and combines it with their long-term Diffie–Hellman key. The obtained shared secret is used to compute a MAC which is used in place of the signature in the CertificateVerify message. This computation proves the server's possession of the long-term secret key corresponding to the public

key in the certificate. The client computes the same shared secret by combining its ephemeral secret Diffie–Hellman key with the certified public key, thus verifying the MAC.

6.1 POST-QUANTUM TLS WITHOUT SIGNATURES

In a naive instantiation of an OPTLS-like protocol with KEMs, we require an additional round-trip. To compute the authentication message, the server needs to first receive the ciphertext that was encapsulated against the long-term public key held in its certificate—which the client can not send before having received it from the server. The KEMTLS proposal by Schwabe, Stebila, and Wiggers avoids this issue partially by letting the client already transmit data immediately after computing and sending the ciphertext to the server [341]. This relies on the fact that any keys derived from the shared secret encapsulated to the server's long term key are *implicitly authenticated*. KEMTLS has the advantage of not having to compute any post-quantum signatures during the handshake protocol, which are typically either expensive or large. Only the variant that assumes the client already has the server's public key, for example through caching, can achieve a protocol flow that is similar to OPTLS and TLS 1.3 [339]. In that flow, the server can send authenticated data immediately on their first response to the client.

However, as CSIDH *does* provide a post-quantum NIKE, we can use it to instantiate post-quantum OPTLS and avoid any online post-quantum signatures. Because OPTLS immediately confirms the server's authenticity, its handshake has the same number of transmissions of messages as TLS 1.3 and there is no need to rely on implicit authentication. Integrating our implementations in OPTLS gives us an understanding of how CSIDH affects the performance of real-world network protocols, which will typically feature similar cryptographic operations and transmissions.

6.2 BENCHMARKING SET-UP

INTEGRATION INTO RUSTLS. To investigate the performance of OPTLS with CSIDH, we integrate our optimized implementations into the framework of KEMTLS [340]. We add OPTLS to the same modified version of Rustls [57] used to implement KEMTLS, which allows us to straightforwardly compare to KEMTLS and TLS 1.3 handshakes instantiated with post-quantum primitives.

CACHING EPHEMERAL KEYS. A CSIDH-OPTLS handshake requires a large number of group acton in each handshake: First, in the generation of the ephemeral key of both the client and the server. Second, in the computation of both the ephemeral shared secret as well as the authentication shared secret, again for both the client and the server.

Unfortunately, due to the order of the handshake messages and the requirements for handshake encryption, most of these computations need to be done in-order and can not really be parallelized. However, we can avoid the cost of CSIDH key generation by implementing caching of ephemeral keys. This reduces the forward secrecy but emulates a best-case scenario for CSIDH-based OPTLS in which the keys are generated "offline", outside the handshake context. We therefore exclude all first TLS handshakes from clients and servers from our measurements to exclude key generation time: in the *pregen* OPTLS instances, all subsequent handshakes use the same public-key material. In the *ephemeral* OPTLS instances, we generate ephemeral keys in each handshake.

Note that OPTLS combines the ephemeral and static keys, thus all need to use the same algorithm and we can not use a faster KEM for ephemeral key exchange.

MEASUREMENT SETUP. We run all TLS handshake experiments on a server with two Intel Xeon Gold 6230 CPUs, each featuring 20 physical cores. This gives us 80 hyperthreaded cores in total. For these experiments, we do *not* disable hyperthreading or frequency scaling, as these features would also be enabled in production scenarios. We run 80 servers and clients in parallel, as each pair of client and server roughly interleave their execution. We collect 8000 measurements per experiment. Every 11 handshakes, we restart the client and server, so that we measure many ephemeral keys even in the scenarios that use ephemeral-key caching. We exclude the first handshake from the measurements to allow for cache warm-up and ephemeral-key generation in the caching scenario.

As in the KEMTLS papers [339, 341], we measure the performance of the different TLS handshakes when run over two network environments: a low-latency 30.9 ms round-trip time (RTT), 1000 Mbps and a high-latency 195.5 ms RTT, 10 Mbps network connection. The latency of the former represents a continental, high-bandwidth connection, while the latter represents a transatlantic connection.

6.3 BENCHMARKING RESULTS

In Table 9, we compare OPTLS with dCSIDH and CTIDH with the performance of instantiations of TLS 1.3 and KEMTLS.

Table 9: Transmission sizes in bytes and timing results in seconds until client receives and sends Finished messages for OPTLS, TLS 1.3 and KEMTLS. All instantiations use Falcon-512 for the certificate authority; the CA public key is not transmitted. Authentication (Auth) bytes includes 666 bytes for the Falcon-512 CA signature on the server's certificate.

		Transmission (bytes)		**Handshake latencies** (RTT, link speed) **30.9 ms, 1000 Mbps**		**195.5 ms, 10 Mbps**	
		KEX	**Auth**	**SFIN recv**	**CFIN sent**	**SFIN recv**	**CFIN sent**
OPTLS (pregen)	dCSIDH p2048	544	938	24.468	24.468	24.288	24.288
	CTIDH p2048	512	922	7.346	7.346	7.203	7.203
	CTIDH p4096	1024	1178	36.321	36.321	36.299	36.299
	CTIDH p5120	1280	1306	28.701	28.701	28.580	28.580
OPTLS (ephem.)	dCSIDH p2048	544	938	43.642	43.642	43.486	43.486
	CTIDH p2048	512	922	10.042	10.042	9.882	9.882
	CTIDH p4096	1024	1178	50.039	50.039	49.951	49.951
	CTIDH p5120	1280	1306	42.383	42.383	42.163	42.163
TLS 1.3	Kyber512–Falcon512	1568	2229	0.064	0.064	0.428	0.428
	Kyber512–Dilithium2	1568	4398	0.063	0.063	0.519	0.519
	Kyber768–Falcon1024	2272	3739	0.065	0.065	0.497	0.497
KEMTLS	Kyber512	1568	2234	0.094	0.063	0.593	0.396
	Kyber768	2272	2938	0.094	0.063	0.597	0.400

Comparing the sizes of the handshakes, OPTLS requires fewer bytes on the wire, as it only needs to transmit two ephemeral public keys and one static public key (and the CA signature). KEMTLS requires an additional ciphertext, and TLS an additional signature.

In OPTLS, like in TLS 1.3, the client receives the server's handshake completion message ServerFinished (SFIN) first and before returning a ClientFinished (CFIN) message and its request immediately after. In KEMTLS, SFIN and full server authentication is received a full round-trip after CFIN is received. Nevertheless, it is clear that the runtime requirements of dCSIDH are almost insurmountable, even for the smallest parameters (p2048). Even CTIDH is orders of magnitude slower than the KEMTLS and OPTLS instances. If the more conservative p4096 prime is required for NIST Level I security, even CTIDH handshakes do not complete in under 30 seconds. Due to a better reduction al-

gorithm, the p5120 prime performs roughly on par with p4096, while providing NIST Level II security in the aggressive analysis (see Table 4).

As discussed by Gonzalez and Wiggers [197], for TLS and KEMTLS in constrained environments and certain scenarios, such as 46 kbps IoT networks, the transmission size can become the dominant factor instead of computation time. However, with the results from this chapter, we expect the environments in which CSIDH-based OPTLS instances are competitive to be very niche: To overcome 7 seconds of computational latency, the network needs to take more than 7 seconds to transmit the additional data required for e.g. TLS 1.3 with Dilithium which corresponds to link speeds of less than 1 kilobyte per second. Additionally, these environments often rely on microcontrollers that are much less performant than the Intel CPUs on which we run our implementations.

Interestingly, the CSIDH experiments run on the high-latency, low-bandwidth networks show slightly lower latencies than those on the high-bandwidth, low-latency network. We suspect that this is due to the TCP congestion control algorithm's transmission windows.

§7 Conclusion and Future Work

This chapter presents low-level and high-level optimizations for CSIDH at larger parameter sets, with a deterministic and dummy-free implementation called dCSIDH, and a non-deterministic implementation optimized for speed called CTIDH. These optimizations achieve impressive results on their own: dCSIDH is almost twice as fast as the state-of-the-art, and CTIDH, dropping determinism, is three times as fast as dCSIDH. Further optimizations of the field arithmetic, i.e., utilizing vector processing capabilities of modern processors, might push these speed-ups further.

Nevertheless, when integrated into the latency-sensitive TLS variant OPTLS, both implementations still have too-large handshake latency in comparison to TLS or KEMTLS using lattice-based KEMs. We conclude that the reduced number of roundtrips, through the non-interactive nature of CSIDH, does not make up for the performance hit. For *truly non-interactive* settings, where latency has the least priority and we cannot replace NIKEs by KEMs, the performance of CSIDH may be sufficient even at high-security levels. This includes, for example, using CSIDH in X3DH [268] for post-quantum Signal, as it would incur a delay of seconds only when sending the first message to another user who might be offline, thus ruling out KEM-based interactive approaches.

Unless significant performance improvements occur for CSIDH in large parameters, or if the quantum-security debate shifts in favor of 512- to 1024-bit parameters, we conclude that CSIDH is unlikely to be practical in real-world applications, outside of those that specifically require NIKEs. It will be interesting to investigate how CSIDH and SWOOSH, the only two current proposals for a post-quantum NIKE, compare in a protocol context. Unfortunately, there is no full implementation of SWOOSH yet: the cycle counts reported in [183] are for the passively-secure core component only. Based on the available numbers, it seems likely that SWOOSH outperforms large-parameter CSIDH in terms of latency, but that key sizes are much smaller for CSIDH.

VIII

EFFECTIVE PAIRINGS IN ISOGENY-BASED CRYPTOGRAPHY

In this chapter, we study pairings and their applications in isogeny-based cryptography. Pairings are useful tools in isogeny-based cryptography and have been used in SIDH/SIKE and other protocols. As a general technique, pairings can be used to move problems about points on curves to elements in finite fields. Until now however, their applicability was limited to curves over fields with primes of a specific shape and pairings seemed too costly for the type of primes that are nowadays often used in isogeny-based cryptography.

We remove this roadblock by optimizing pairings for highly-composite degrees such as those encountered in CSIDH and SQIsign. This makes the general technique viable again: We apply our low-cost pairing to problems of general interest, such as supersingularity verification and finding full-torsion points, and show that we can outperform current methods, in some cases up to four times faster than the state-of-the-art. Furthermore, we analyze how pairings can be used to improve deterministic and dummy-free CSIDH. Finally, we provide a constant-time implementation (in Rust) that shows the practicality of these algorithms.

This chapter is based on the paper

> Krijn Reijnders. "Effective Pairings in Isogeny-based Cryptography". In: *International Conference on Cryptology and Information Security in Latin America*. Springer. 2023, pp. 109–128.

I have only made minor editorial changes to smoothen notation. I have explicitly left the statements in this chapter in the terms they were originally posed, without the renewed understanding gained for pairings in Part 2. Reformulations in those terms are left as a (fun!) exercise for the reader. I thank Michael Scott for his support in optimizing pairing computations.

§1 Introduction

In the event that quantum computers break current cryptography, post-quantum cryptography will provide the primitives required for digital security. Isogeny-based cryptography is a field with promising quantum-secure schemes, offering small public keys in key exchanges using CSIDH [95], and small signatures using SQIsign [146]. The main drawback of isogeny-based cryptography is speed, as it requires heavy mathematical machinery in comparison to other areas of post-quantum cryptography. In particular, to ensure security against physical attacks, the requirements for constant-time and leakage-free implementations cause a significant slowdown. Trends in current research in isogenies are, therefore, analyzing side-channel threats [78, 255], including Chapters IV and V and looking at new ideas to improve constant-time performance [20, 23, 98, 102, 107, 213, 274, 276, 301] including Chapter VII.

Surprisingly, although pairings were initially considered in SIDH and SIKE to improve the cost of key compression [130, 131], they have received little attention for optimizing CSIDH or later isogeny-based protocols. There are clear obstructions that heavily affect the performance of pairings: we have little control over the Hamming weight of p for the base fields (in CSIDH or SQIsign), we are likely to compute pairings of highly-composite degree, and many optimizations in the pairing-based literature require different curve models than the ones we consider in isogeny-based cryptography. Nevertheless, the field of pairing-based cryptography is rich in ideas and altogether many small improvements can make practical pairings even on unpractical curves. As a general technique, we can use pairings to analyze certain properties of *points on elliptic curves* by a pairing evaluation as *elements of finite fields*. In this way, a single pairing can be used to solve a curve-theoretical problem with only field arithmetic, which is much more efficient. Hence, even a relatively expensive pairing computation can become cost-effective if the resulting problem is much faster to solve "in the field" than "on the curve".

Pairings have been used constructively since the 2000s [208, 226]. The literature is rich, but the main focus has mostly been on pairings of prime degree. Although proposals using composite degree pairings have been explored, analysis such as Guillevic [205] shows that prime degrees are favorable. Composite-degree pairings have thus received little attention compared to prime-degree pairings.

Aside from these issues, CSIDH is well-suited for pairings: CSIDH mostly works with points of order $N \mid p+1$ on supersingular curves $E/\mathbb{F}_p$ whose x-coordinate lives in $\mathbb{F}_p$. This allows for fast x-only arithmetic on the Kummer

line $\mathbb{P}(\mathbb{F}_p)$, but also implies an embedding degree of 2 for pairings of degree $N \mid p+1$. The result $\zeta = t_N(P,Q)$ of a pairing is thus a norm-1 element in $\mathbb{F}_{p^2}$, and ζ contains useful information on P and Q, which is precisely the information that CSIDH often requires to perform certain isogenies. Hence, pairings are a natural tool to study properties of the pair (P,Q). Norm-1 elements in $\mathbb{F}_{p^2}$ allow for very fast $\mathbb{F}_p$-arithmetic (compared to $\mathbb{F}_{p^2}$-arithmetic). A final advantage is that for any *generic* supersingular curve E_A we choose over $\mathbb{F}_p$, the result of a pairing always ends up in the *specific* field $\mathbb{F}_{p^2}$. Thus, we can fix public values in $\mathbb{F}_{p^2}$, which we cannot do for curves: it would require fixing a value for *all* supersingular curves E_A, independent of $A \in \mathbb{F}_p$.

Contributions

We combine optimized pairings with highly-efficient arithmetic in $\mu_r \subseteq \mathbb{F}_{p^2}^*$ to solve isogeny problems faster. To achieve this, we first optimize pairings for CSIDH in Section 3 and then apply this low-cost pairing to move specific problems from curves to finite fields in Sections 4 and 5. Specifically,

1. we first reduce the cost of subroutines Dbl and Add in MillersAlgorithm and then reduce the total number of these subroutines using non-adjacent forms and windowing techniques,
2. we analyze the asymptotic and concrete cost of single and multi-pairings, in particular for supersingular curves over p512 (the prime used in CSIDH-512),
3. we apply these low-cost pairings to develop alternative algorithms for supersingularity verification, verification of full-torsion points, and finding full-torsion points, using highly-efficient arithmetic available for pairing evaluations,
4. we discuss the natural role these algorithms have when designing 'real-world' isogeny-based protocols, in particular, CSIDH-variants that are deterministic and secure against side-channel attacks, as discussed before in Chapter VII, and
5. we provide a full implementation of most of these algorithms in Rust, following the "constant-time" paradigm, that shows such algorithms can immediately be used in practice to speed up deterministic variants of CSIDH.

   Our implementation is available at:

https://github.com/Krijn-math/EPIC

RELATED WORK. This chapter can partly be viewed as a natural follow-up to [98, 102] and in particular Chapter VII, which inspired this chapter. Independent work by Lin, Wang, Xu, and Zhao [258] applies pairings to improve the performance of SQIsign [146]. This shows the potential of pairings in isogeny-based cryptography and we believe this chapter can contribute to improving SQIsign performance even further.

Organization of the paper.

Section 2 introduces the mathematical tools in pairing-based and isogeny-based cryptography required in the rest of the chapter. Section 3 analyzes optimization of pairings to the setting used in isogeny-based cryptography, which allows us to apply pairings to optimize general problems in Section 4. In Section 5, we show that these pairing-based algorithms speed up current variants of deterministic, dummy-free CSIDH and can be used to construct ideas beyond those in use now.

§2 PRELIMINARIES

Chapter II gives an introduction to curves, their isogenies and pairings, and furthermore introduces CSIDH. Here, we only introduce finer arithmetical details on pairings.

Recall that a (supersingular) elliptic curve E_A is assumed to be in Montgomery form

$$E_A : y^2 = x^3 + Ax^2 + x, \quad A \in \mathbb{F}_p,$$

although this chapter applies to other curve forms (most notably Edwards) too. We denote the order of a point $P \in E_A(\mathbb{F}_p)$ by $\mathrm{Ord}(P)$, and define its *missing torsion* by $\mathrm{Miss}(P) = (p+1)/\mathrm{Ord}(P)$. Furthermore, note that in the CSIDH setting μ_{p+1} can be seen as $\mathbb{F}_{p^2}^*/\mathbb{F}_p^*$, the elements in $\mathbb{F}_{p^2}$ of norm 1, and μ_r for $r \mid p+1$ is a subgroup of μ_{p+1}. We use a cost model of $1\mathbf{S} = 0.8\mathbf{M}$ and $1\mathbf{A} = 0.01\mathbf{M}$ to compare performance in terms of finite-field multiplications.

THE PRIME p. We specifically look at supersingular elliptic curves over $\mathbb{F}_p$, where $p = h \cdot \prod_{i=1}^{n} \ell_i - 1$, with h is a suitable cofactor and the ℓ_i are small odd primes. We refer to these ℓ_i as *Elkies primes* [95]. We denote the set of Elkies

primes as L_χ[1] and write $\ell_\chi = \prod_{\ell_i \in L_\chi} \ell_i$. Hence, $\log p = \log h + \log \ell_\chi$. If h is large, the difference in bit-size between ℓ_χ and $p+1$ can be significant, and this can impact performance whenever an algorithm takes either $\log \ell_\chi$ or $\log p$ steps. For p512, h is only 4 and so we do not differentiate between the two.

2.1 ISOGENY-BASED CRYPTOGRAPHY

TORSION POINTS. Let E be a supersingular elliptic curve over $\mathbb{F}_p$, then E has $p+1$ rational points. Such points $P \in E(\mathbb{F}_p)$ therefore have order $N \mid p+1$. When $\ell_i \mid N$, we say P has ℓ_i-torsion. When P is of order $p+1$, we say P is a *full-torsion* point. The twist of E over $\mathbb{F}_p$ is denoted by E^t, and E^t is also supersingular. Rational points of E^t can also be seen as $\mathbb{F}_{p^2}$-points in $E[p+1]$ of the form (x, iy) for $x, y \in \mathbb{F}_p$. Using x-only arithmetic, we can do arithmetic on *both* $E(\mathbb{F}_p)$ and $E^t(\mathbb{F}_p)$ using only these rational x-coordinates. Full-torsion points are useful to compute the class group action in CSIDH, as explained in Chapter II.

DETERMINISTIC CSIDH. The probabilistic nature of the evaluation of $\mathfrak{a} * E$, stemming from the random sampling of points P, Q, causes several issues, as randomness makes deterministic or constant-time implementations difficult as discussed in Chapter VII. One way to avoid this random nature of $\mathfrak{a} * E$ is to ensure that both P and Q are *full-torsion points*, e.g., we have $\mathrm{Ord}(P) = \mathrm{Ord}(Q) = p+1$. CSIDH strategies that require only two full-torsion points $T_+ \in E(\mathbb{F}_p)$ and $T_- \in E^t(\mathbb{F}_p)$ are discussed in [98] and implemented and improved by [102]. We have explored and improved such strategies in more detail in Chapter VII. All these implementations restrict secret-key coefficients t_i to $-1, +1$ to remove randomness and dummy operations.

2.2 BUILDING BLOCKS IN ISOGENY-BASED CRYPTOGRAPHY

We list several general routines in isogeny-based cryptography that will be analyzed in more detail in later sections. These routines are posed as general problems, with their role in CSIDH specified afterward.

1. Given $P \in E(\mathbb{F}_p)$, find the order of P.
2. Given $A \in \mathbb{F}_p$, verify E_A is supersingular.
3. Given $P \in E(\mathbb{F}_p)$, $Q \in E^t(\mathbb{F}_p)$, verify both have order $p+1$.

1 pronounced "ell-kie"

4. Given E, find $P \in E(\mathbb{F}_p)$, $Q \in E^t(\mathbb{F}_p)$ both of order $p+1$.

These problems are inter-related. For example, verifying supersingularity is usually done by showing that there is a $\mathbb{F}_p$-rational point of order $N \geq 4\sqrt{p}$, which implies that $\#E_A(\mathbb{F}_p) = p+1$. All variants of CSIDH use a supersingularity verification in order to ensure a public key E_A is valid. As introduced in Chapter VII, dCSIDH includes (an identifier of) full-torsion points P, Q in the public key to speed up the shared-secret computation. This requires both *finding* full-torsion points in key generation and, given the public key, *verifying* the points P, Q are both full-torsion before deriving the shared secret. Without this verification, a user is vulnerable to physical attacks.

2.3 PAIRING-BASED CRYPTOGRAPHY

One of the goals in pairing-based cryptography is to minimize the cost of computing a Weil or Tate pairing, as defined in Section 7. This section requires a more computatonal understanding, for which we must furthermore assume a basic familiarity with pairings up to the level of Costello's tutorial [126]. Other great resources are Galbraith [188] and Scott [347]. We focus only on the reduced Tate pairing, as it is more efficient for our purposes. We build on top of the fundamental works [31, 32, 279, 374].

THE REDUCED TATE PAIRING. In this work, we are specifically focused on the reduced Tate pairing of degree r for supersingular elliptic curves with embedding degree $k = 2$, which can be seen as a bilinear pairing

$$t_r : E[r] \times E(\mathbb{F}_{p^2})/rE(\mathbb{F}_{p^2}) \to \mathbb{F}_{p^2}^*/(\mathbb{F}_{p^2}^*)^r.$$

In this *reduced* Tate pairing[2], the result $\zeta = t_r(P, Q)$ is raised to the power $k = (p^2-1)/r$, which ensures ζ^k is an r-th root of unity in μ_r. In this work, we want to evaluate the Tate pairing on points $P \in E(\mathbb{F}_p)$ and $Q \in E^t(\mathbb{F}_p)$ of order r. For supersingular curves over $\mathbb{F}_p$ and $r \mid p+1$, such points generate all of $E[r]$. From the point of view of pairings, $E(\mathbb{F}_p)$ is the *base-field subgroup* and $E^t(\mathbb{F}_p)$ is the *trace-zero subgroup* of $E[p+1]$. Using the bilinear properties of the Tate pairing, we can compute t_r from its restriction to $E(\mathbb{F}_p) \times E^t(\mathbb{F}_p)$.

2 The unreduced Tate pairing simply returns *some* representant of the class in $\mathbb{F}_{p^2}^*/(\mathbb{F}_{p^2}^*)^r$, we require a unique representant.

COMPUTING THE TATE PAIRING. There are multiple ways to compute the Tate pairing [260, 262, 279, 358]. Most implementations evaluate t_r in three steps.

1. compute the Miller function $f_{r,P}$, satisfying $\mathrm{div}(f_{r,P}) = r(P) - r(\mathcal{O})$,
2. evaluate $f_{r,P}$ on an appropriate divisor D_Q, and
3. raise $f_{r,P}(D_Q)$ by $k = \frac{p^2-1}{r}$.

The last step is also known as the *final exponentiation*.

In practice, $f_{r,P}$ is a function in x and y of degree r, where r is cryptographically large, and therefore infeasible to store or evaluate. Miller's solution is a bitwise computation and direct evaluation of $f_{r,P}$ on D_Q to compute $f_{r,P}(D_Q)$ in $\log(r)$ steps. Barreto, Kim, Lynn, and Scott [32, Theorem 1] show that we can choose $D_Q = Q$, which simplifies computations. The Hamming weight of r is a large factor in the cost of computing $f_{r,P}(Q)$: to compute $f_{r,P}$ we go over each bit of r, and the computational cost if the bit is 1 is close to twice as expensive as when the bit is 0. For our purposes, we need either $r = p + 1$ or $r = \ell_\chi$. As p is chosen such that the isogeny computations on CSIDH are effective, we have little control over the Hamming weight of r. Algorithm 6 describes MillersAlgorithm before applying any optimizations, where $l_{T,T}$ and $l_{T,P}$ denote the required line functions (see [126, § 5.3]). We refer to the specific subroutines in Line 3 as Dbl and Line 5 as Add, which are described in more detail in Section A.

Algorithm 6 MillersAlgorithm

Input: $P \in E(\mathbb{F}_p)$, $Q \in E^t(\mathbb{F}_p)$, r of embedding degree $k = 2$, with $r = \sum_{i=0}^{t} t_i \cdot 2^i$
Output: The reduced Tate pairing $t_r(P, Q) \in \mu_r$

1: $T \leftarrow P$, $f \leftarrow 1$
2: **for** i from $t - 1$ to 0 **do**
3: $\quad T \leftarrow 2T$, $f \leftarrow f^2 \cdot l_{T,T}(Q)$ ▷ Dbl
4: $\quad$ **if** $t_i = 1$ **then**
5: $\quad\quad T \leftarrow T + P$, $f \leftarrow f \cdot l_{T,P}(Q)$ ▷ Add
6: **return** $f^{\frac{p^2-1}{r}}$

More generally, the value f is updated according to the formula

$$f_{(n+m),P} = f_{n,P} \cdot f_{m,P} \cdot \frac{l}{v} \tag{9}$$

where l and v are the lines that arise in the geometric interpretation of the addition of $[n]P$ and $[m]P$. MillersAlgorithm uses $n = m$ to double $T = [n]P$, or $m = 1$ to add $T = [n]P$ and P.

2.4 FIELD ARITHMETIC

As the norm of $\alpha \in \mathbb{F}_{p^2}^*$ is $\alpha \cdot \alpha^p = \alpha^{p+1}$, the result of the reduced Tate pairing for $r = p + 1$ are precisely the norm-1 elements of $\mathbb{F}_{p^2}^*$. We will use two algorithms from finite-field arithmetic: GaussAlgorithm to find primitive roots and Lucas exponentiation.

GAUSS'S ALGORITHM. An algorithm attributed to Gauss [270, p. 38] to find primitive roots of a certain order in a finite field is given in Algorithm 7, specialized to the case of finding a generator α for a finite field $\mathbb{F}_q$. It assumes a subroutine Ord computing the order of any element in the finite field.

Algorithm 7 GaussAlgorithm.

Input: A prime power $q = p^k$.
Output: A generator α for $\mathbb{F}_q^*$.

1: $\alpha \xleftarrow{\$} \mathbb{F}_q^*$, $t \leftarrow \mathrm{Ord}(\alpha)$
2: **while** $t \neq q - 1$ **do**
3: $\quad \beta \xleftarrow{\$} \mathbb{F}_q^*$, $s \leftarrow \mathrm{Ord}(\beta)$
4: $\quad$ **if** $s = q - 1$ **then return** β
5: $\quad$ **else**
6: $\quad\quad$ Find $d \mid t$ and $e \mid s$ with $\gcd(d, e) = 1$ and $d \cdot e = \mathrm{lcm}(t, s)$
7: $\quad\quad$ Set $\alpha \leftarrow \alpha^{t/d} \cdot \beta^{s/e}$, $t \leftarrow d \cdot e$
8: **return** α

GaussAlgorithm is easy to implement and finds generators quickly. The main cost is computing the orders. We can adapt GaussAlgorithm to elliptic curves to find generators for $E(\mathbb{F}_p)$, simply by replacing α, β by rational points P, P' until P reaches $\mathrm{Ord}(P) = p + 1$. Intuitively, one could say we "add" the torsion that P is missing using the right multiple of P'.

LUCAS EXPONENTIATION. Lucas exponentiation provides fast exponentiation for $\zeta \in \mathbb{F}_{p^2}$ of norm 1. This has been used in cryptography since 1996 [228, 229], and specifically applied to pairings by Scott and Barreto [348]. We follow their notation.

Let $\zeta = a + bi \in \mathbb{F}_{p^2}$ be an element of norm 1, i.e. $a^2 + b^2 = 1$, then ζ^k can be efficiently computed using only $a \in \mathbb{F}_p$ for every $k \in \mathbb{N}$ using Lucas sequences, based on simple laddering algorithms. We denote these sequences by $V_k(a)$ and $U_k(a)$ but often drop a for clarity. The central observation is

$$\zeta^k = (a + bi)^k = V_k(2a)/2 + U_k(2a) \cdot bi, \quad \text{for } \zeta \in \mu_{p+1},$$

where

$$\begin{aligned} V_0 = 2, \quad V_1 = a, \quad V_{k+1} = a \cdot V_k - V_{k-1}, \\ U_0 = 0, \quad U_1 = 1, \quad U_{k+1} = a \cdot U_k - U_{k-1}. \end{aligned}$$

Given V_k, we can compute U_k by $(a \cdot V_k - 2 \cdot V_{k-1})/(a^2 - 4)$. An algorithmic description is given in [348, App. A]. For this work, we only require the value of $V_k(2a)$. As such, exponentiation of norm-1 elements is much more efficient than general exponentiation in $\mathbb{F}_{p^2}^*$: the former requires $1\mathbf{S} + 1\mathbf{M}$ per bit of k, whereas the latter requires roughly $2\mathbf{S} + \frac{5}{2}\mathbf{M}$ per bit of k, assuming the Hamming weight of k is $\log(k)/2$. In our cost model, this is an almost 60% improvement. We denote the cost of exponentiation per bit for norm-1 elements by C_{Lucas}.

This arithmetical speed-up is key to the applications in this work: as the required pairings have evaluations of norm 1, we can apply Lucas exponentiation to the results to get very fast arithmetic. In comparison to x-only arithmetic on the curve, we are between five and six times faster per bit. Hence, if the cost of the pairing is low enough, the difference in cost between curve arithmetic and Lucas exponentiation is so large that it makes up for the cost of the pairing.

§3 Optimizing Pairings for Composite Order

In this section, we apply several techniques to decrease the cost of MillersAlgorithm, specifically for pairings of degree $r \mid p + 1$, and points $P \in E(\mathbb{F}_p)$, $Q \in E^t(\mathbb{F}_p)$, with E supersingular and $p = h \cdot \ell_\chi - 1$. This is a different scenario than pairing-based literature usually considers: we have no control over the Hamming weight of r, which is furthermore composite.

We first give an abstract view and then start optimizing MillersAlgorithm. In Section 3.2, we decrease the cost per subroutine Dbl/Add with known optimizations that fit our scenario perfectly. In Section 3.3, we decrease the number of subroutines Dbl/Add using non-adjacent forms and (sliding) window

techniques, inspired by finite-field exponentiation and elliptic-curve scalar-multiplication algorithms.

3.1 AN ABSTRACT VIEW ON PAIRINGS

Silverman [355] views the reduced Tate pairing as a threefold composition

$$E[r] \to \mathrm{Hom}(E[r], \mu_r) \to \mathbb{F}_{p^2}^* / \mathbb{F}_{p^2}^{*r} \xrightarrow{z \mapsto z^{(p^2-1)/r}} \mu_r \tag{10}$$

similar to the one described in Section 2.3. Namely, for $r = p+1$, we can reduce the first map to $E(\mathbb{F}_p) \to \mathrm{Hom}(E^t(\mathbb{F}_p), \mu_r)$ as

$$\Psi : E(\mathbb{F}_p) \to \mathrm{Hom}(E^t(\mathbb{F}_p), \mu_r), \quad P \mapsto t_r(P, -),$$

which can be made instantiated by a Miller function $P \mapsto f_{r,P}$. By composing with its evaluation on Q, we get $f_{r,P}(Q) = t_r(P, Q)$ (unreduced). To $f_{r,P}(Q)$, we apply the final exponentiation $z \mapsto z^{(p^2-1)/r}$. In the case of $r = p+1$, we thus get the reduced Tate pairing $t_{p+1}(P, Q)$ as $\zeta = f_{p+1,P}(Q)^{p-1}$.

From this point of view, identifying full-torsion points $P \in E(\mathbb{F}_p)$ is equivalent to finding points P that map to isomorphisms $f_{r,P} \in \mathrm{Hom}(E^t(\mathbb{F}_p), \mu_r)$. We make this precise in the following lemma.

Lemma 1. Let E be a supersingular elliptic curve over $\mathbb{F}_p$. Let $P \in E(\mathbb{F}_p)$ and $r = p+1$. Then $f_{r,P}$ as a function $E^t(\mathbb{F}_p) \to \mu_r$ has kernel

$$\ker f_{r,P} = \{Q \in E^t(\mathbb{F}_p) \mid \mathrm{Ord}(P) \text{ divides } \mathrm{Miss}(Q)\}.$$

Hence, $|\ker f_{r,P}| = \mathrm{Miss}(P)$. Thus, if P generates $E(\mathbb{F}_p)$, the $\ker f_{r,P}$ is trivial.

Proof. Recall that $\mathrm{Ord}(P) \cdot \mathrm{Miss}(P) = p+1$. The reduced Tate pairing maps precisely to a primitive $p+1$-th root of unity if and only if P and Q are a torsion basis for $E[p+1]$ [355]. Note that if $P \in E(\mathbb{F}_p)$ does not have order $p+1$, we can write $P = [\mathrm{Miss}(P)]T_+$ for some specific point $T_+ \in E(\mathbb{F}_p)$ of order $p+1$, and similarly $Q = [\mathrm{Miss}(Q)]T_-$ for some $T_- \in E^t(\mathbb{F}_p)$. Hence $t_{p+1}(T_+, T_-)$ is an exact $p+1$-th root of unity, and

$$\zeta = t_{p+1}(P, Q) = t_{p+1}(T_+, T_-)^{\mathrm{Miss}(P) \cdot \mathrm{Miss}(Q)}$$

is a $p+1$-th root of unity with $\mathrm{Miss}(\zeta) = \mathrm{lcm}(\mathrm{Miss}(P), \mathrm{Miss}(Q))$. The points $Q \in E^t(\mathbb{F}_p)$ such that $f_{r,P}(Q) = 1$ are then precisely the points with $\zeta =$

1, and thus $\operatorname{lcm}(\operatorname{Miss}(P) \cdot \operatorname{Miss}(Q)) = p+1$. Whenever $\operatorname{Ord}(P)$ divides $\operatorname{Miss}(Q)$, we know that $p+1$ divides $\operatorname{Miss}(P) \cdot \operatorname{Miss}(Q)$ and so we must have $\operatorname{lcm}(\operatorname{Miss}(P), \operatorname{Miss}(Q)) = p+1$. As $\operatorname{Ord}(P) \mid \operatorname{Miss}(Q)$ implies $\operatorname{Ord}(Q) \mid \operatorname{Miss}(P)$, we can generate all such points Q by a single point R of order $\operatorname{Miss}(P)$, giving us $\ker f_{r,P} = \langle R \rangle$ of order $\operatorname{Miss}(P)$. □

This result essentially described what information about the curve points P and Q remains after using the Tate pairing to move to the finite field, and will therefore be crucial for all our algorithms. For example, given a full-torsion point $P \in E(\mathbb{F}_p)$, we can identify full-torsion points $Q \in E^t(\mathbb{F}_p)$ as points where $t_r(P,Q)$ is a primitive root in μ_r. More generally, using Lemma 1, we can try to tackle the problems sketched in Section 2.2 using properties of $f_{r,P}$, $f_{r,P}(Q)$ and $\zeta = f_{r,P}(Q)^{p-1}$. For example, we can find $\ker f_{r,P}$ by evaluating $f_{r,P}$ on multiple points Q_i, and finding the orders of the resulting ζ_i. In the language of pairing-based cryptography, we compute multiple pairings $t_r(P,Q_i)$ for the same point P. Hence, beyond our previous goals, we furthermore need to minimize the cost of several evaluations of the Tate pairing for fixed P but different Q_i.

3.2 REDUCING THE COST PER SUBROUTINE OF MILLER'S LOOP

In this section, we optimize the cost per Dbl and Add in MillersAlgorithm. We assume that $P \in E(\mathbb{F}_p)$ is given by $\mathbb{F}_p$-coordinates $x_P, y_P \in \mathbb{F}_p$, and $Q \in E^t(\mathbb{F}_p)$ is given by $\mathbb{F}_p$-coordinates $x_Q, y_Q \in \mathbb{F}_p$, implicitly thinking of Q as $(x_Q, i \cdot y_Q)$. Some of these techniques were used before in SIDH and SIKE [130, 131], in a different situation: in SIDH and SIKE, these pairings were specifically applied for $p = 2^{e_2} \cdot 3^{e_3} - 1$, whereas we assume $p = h \cdot \ell_\chi - 1$. Thus, we have much more different $\ell_i \mid p+1$ to work with, and we cannot apply most of their techniques.

REPRESENTATIONS. For the T value in MillersAlgorithm, we use projective coordinates to avoid costly inversions when doubling T, adding P and computing $\ell_{T,T}$ and $\ell_{T,P}$. We can leave Q affine, as we only evaluate Q in $\ell_{T,T}$ and $\ell_{T,P}$. We represent $\mathbb{F}_{p^2}$-values $f = a + bi$ projectively as $(a : b : c)$, with $a, b, c \in \mathbb{F}_p$ and c as the denominator. Although x-only pairings exist [185], they seem unfit for this specific scenario.

THE FINAL EXPONENTIATION. As established before, after computing $f_{r,P}(Q) \in \mathbb{F}_{p^2}$ we perform a final exponentiation by $p-1$. This is beneficial for two reasons:
1.) Raising to the power p is precisely applying Frobenius $\pi : z \mapsto z^p$ in $\mathbb{F}_{p^2}$, and so $\pi : a + bi \mapsto a - bi$. Hence we can compute $z \mapsto z^{p-1}$ as $z \mapsto \frac{\pi(z)}{z}$. The Frobenius part is 'free' in terms of computational cost. In $\mathbb{F}_{p^2}$, inversion is simply $(a+bi)^{-1} = \frac{(a-bi)}{a^2+b^2}$. Hence, the $\mathbb{F}_p$-inversion of a^2+b^2 is the dominating cost of the final exponentiation. In constant-time, this costs about $\log p$ multiplications. When a and b are public, we use faster non-constant-time inversions. In comparison to prime pairings, this final exponentiation is surprisingly efficient.
2.) Raising z to the power $p-1$ for $\mathbb{F}_{p^2}$-values gives the same as $\alpha \cdot z$ for $\alpha \in \mathbb{F}_p^*$, as $(\alpha \cdot z)^{p-1} = \alpha^{p-1} \cdot z^{p-1} = z^{p-1}$. Hence, when we compute $f_{r,P}(Q) \in \mathbb{F}_{p^2}$, we can ignore or multiply by $\mathbb{F}_p$-values [32], named "denominator elimination" in pairing literature. That is, we ignore the denominator c in the representation $(a : b : c)$ of f in the Miller loop, and similarly in evaluating $l_{T,T}(Q)$ and $l_{T,P}(Q)$, saving several $\mathbb{F}_p$-operations in Dbl/Add [99, 344].

REUSING INTERMEDIATE VALUES. In Dbl, computing $T \leftarrow 2T$ shares many values with the computation of $l_{T,T}$. In Add, computing $T \leftarrow T + P$ shares values with $l_{T,P}$. Reusing such values saves several $\mathbb{F}_p$-operations in Dbl/Add.

IMPROVED DOUBLING FORMULAS. As shown in [130, §4.1], the subroutine Dbl in MillersAlgorithm is more efficient when using a projective representation of $T \in E(\mathbb{F}_p)$ as (X^2, XZ, Z^2, YZ), although this requires a slight adjustment of the formulas used in Add. Overall, this reduces the cost for Dbl to $5\mathbf{S} + 15\mathbf{M}$ and the cost for Add becomes $4\mathbf{S} + 20\mathbf{M}$, for an average of $7\mathbf{S} + 25\mathbf{M}$ per bit.

3.3 REDUCING THE NUMBER OF ADDS IN MILLER'S LOOP

Next to reducing the cost for a single Dbl or Add, we apply techniques to reduce the total number of Adds. Usually in pairing-based cryptography, we do so by using primes p of low Hamming weight. Here we do not have this freedom, thus we resort to techniques inspired by exponentiation in finite fields.

NON-ADJACENT FORM. With no control over the Hamming weight of $p+1$, we assume half of the bits are 1. However, in MillersAlgorithm, it is as easy to add $T \leftarrow T + P$ as it is to subtract $T \leftarrow T - P$ (which we denote Sub), with the

only difference being a negation of y_P. Hence, we use *non-adjacent forms* [324] (NAFs) to reduce the number of Add/Subs. A NAF representation of $p+1$,

$$p+1 = \sum_{i=0}^{n} t_i \cdot 2^i, \quad t_i \in \{-1, 0, 1\},$$

reduces the Hamming weight from $\log(p)/2$ to $\log(p)/3$, and thus decrease the number of expensive Add/Subs in MillersAlgorithm by $\log(p)/6$. We get an average cost of $6\frac{1}{3}\mathbf{S} + 21\frac{2}{3}\mathbf{M}$ per bit, a saving of about 10%.

Algorithm 8 gives a high-level overview of the Miller loop with the improvements so far, for general p. As the output $\zeta \in \mathbb{F}_{p^2}$ will have norm 1, we can represent ζ using only its real part. See Section A for the specific algorithms Dbl, Add and Sub, which are implemented in bignafmiller.rs in our Rust implementation.

Algorithm 8 Miller's algorithm, using NAFs

Input: $x_P, y_p, x_Q, y_Q \in \mathbb{F}_p$, $p+1 = \sum_{i=0}^{t} t_i \cdot 2^i$
Output: The real part of $t_r(P,Q) \in \mu_{p+1}$

$T = (X^2, XZ, Z^2, YZ) \leftarrow (x_P^2, x_P, 1, y_P)$
$f \leftarrow (1, 0)$
for i from $t-1$ to 0 **do**
    $(T, f) \leftarrow \mathsf{Dbl}(T, f, x_Q, y_Q)$
    if $t_i = 1$ **then**
        $(T, f) \leftarrow \mathsf{Add}(T, f, x_P, y_P, x_Q, y_Q)$
    if $t_i = -1$ **then**
        $(T, f) \leftarrow \mathsf{Sub}(T, f, x_P, y_P, x_Q, y_Q)$
$a \leftarrow f[0]$, $b \leftarrow f[1]$
$\zeta \leftarrow \frac{a^2 - b^2}{a^2 + b^2}$ ▷ The final exponentiation
return ζ

For specific primes.

For specific primes p, such as p512, we can improve on the NAF representation by using windowing techniques [238, 239]. This allows us to decrease even further the times we need to perform Add or Sub, at the cost of a precomputation of several values.

In short, windowing techniques allow us to not only add or subtract P but also multiples of P during the loop. To do so, we are required to precompute

several values, namely $[i]P, [-i]P, f_{i,P}$ and $f_{-i,P}$: We need the multiples $[\pm i]P$ to perform $T \leftarrow T \pm [i]P$, and the line functions $f_{i,P}$ to set $f \leftarrow f \cdot f_{\pm i,P} \cdot \ell_{T,[i]P}(Q)$ in Add/Sub. We first precompute the required $[i]P$ in projective form, and we keep track of $f_{i,P}$. We use Montgomery's trick [280] to return the points $[i]P$ in affine form at the cost of a single inversion and some multiplications. Using affine form decreases the cost of $T \leftarrow T \pm [i]P$ during Add/Sub.

Note that $[i]P$ gives $[-i]P$ for free, simply by negating $y_{[i]P}$. Furthermore, from $f_{i,P} = a + bi$ we can obtain $f_{-i,P}$ as $f_{i,P}^{-1} = \frac{a-bi}{a^2+b^2}$. However, as $a^2 + b^2 \in \mathbb{F}_p^*$, we can ignore these, thanks to the final exponentiation, and simply set $f_{-i,P} = a - bi$. Altogether, these sliding-window techniques reduce the number of Add/Subs from $\log(p)/3$ down to about $\log(p)/(w+1)$. See bigwindowmiller.rs for the implementation of this algorithm.

For the prime p512, we found the optimum at a window size $w = 5$. This requires a precomputation of $\{P, [3]P, \ldots, [21]P\}$. Beyond $w = 5$, the cost of additional computation does not outweigh the decrease in Add/Subs. Altogether this gives another saving of close to 10% for p512.

Remark 1. A reader who is familiar with curve arithmetic might be tempted to suggest (differential) addition chains at this point, specialized for $p + 1$, to decrease even further the number of Add/Subs. However, an addition chain requires either keeping track of the different points in the chain in projective form, hence performing projective additions/subtractions, or, converting them to affine points and performing affine additions/subtractions. Both approaches are not cost-effective; the amount of additions/subtractions hardly decreases, and the cost per addition/subtraction increases significantly. The crucial difference is that the precomputed points can be mapped into affine form using a single shared inversion and a few multiplications.

3.4 MULTIPLE PAIRING EVALUATIONS

The previous sections attempt to optimize a single pairing $t_r(P, Q)$. However, in many scenarios, including the ones in Section 4, it is beneficial to optimize the cost of multiple pairings, in particular multiple pairings $t_r(P, Q_1), \ldots t_r(P, Q_k)$ for the same point P. This is known as the "one more pairing" problem in pairing literature. We quickly sketch two methods to do so. First, we assume the set $\{Q_1, \ldots, Q_k\}$ is known in advance and we minimize the overall cost of these k pairings. Second, we assume we want to compute additional pairings $t_r(P, Q_{k+1})$ *after* having already computed $t_r(P, Q_1), \ldots, t_r(P, Q_k)$.

Evaluating a fixed set $Q_1, Q_2, \ldots, Q_k$.

Optimizing k pairings $t_r(P, Q_i)$ for an already known set $Q_1, \ldots, Q_k$ is easy: only f depends on Q_i, hence we can easily adapt Algorithm 8 for an array of points $[Q_1, \ldots, Q_k]$ to keep track of a value $f^{(i)}$ per Q_i, and we return an array $\zeta^{(1)}, \ldots, \zeta^{(k)}$. All evaluations share (per bit) the computations of T, $l_{T,T}$ and $l_{T,P}$. Our additional cost per extra point Q_i thus comes down to the evaluations $l_{T,T}(Q_i)$, $l_{T,P}(Q_i)$ and updating $f^{(i)}$. In total, this is $7\mathbf{M}$ per Dbl, and $5\mathbf{M}$ per Add/Sub, plus $2\mathbf{S} + \mathbf{I}$ to compute $\zeta^{(i)}$ given $f^{(i)}$ (the final exponentiation) per point Q_i. See bigmultimiller.rs for the implementation of this algorithm.

Evaluating an additional pairing with Q'.

It is more difficult to effectively compute $t_r(P, Q')$ *after* the computation of $t_r(P, Q)$. That is, in some applications we compute $t_r(P, Q)$ first, and, based on this evaluation, compute another $t_r(P, Q')$. In practice, this seems to require another full pairing computation.

However, Scott [346] observed that one can achieve a time/memory trade-off, by dividing MillersAlgorithm into three distinct subalgorithms: one to compute $f_{r,P}$, one to evaluate $f_{r,P}(Q_i)$ per Q_i and one for the final exponentiation. Paradoxically, this brings us back to the original three-step process from Section 2.3, where we argued that the degree of $f_{r,P}$ is too large to store $f_{r,P}$ in full. However, Scott notes that $f_{r,P}(Q)$ can be computed from the set of all line functions $l_{T,T}$ and $l_{T,P}$ and Q. Up to $\mathbb{F}_p$-invariance, all such line functions l can be written as

$$l(x, y) = \lambda_x \cdot x + \lambda_y \cdot y + \lambda_0, \quad \lambda_i \in \mathbb{F}_p,$$

and we get a line function per Dbl, Add, and Sub. At a memory cost of

$$\underbrace{(\log(p) + 1/3\log(p))}_{\text{\# subroutines}} \cdot \underbrace{3 \cdot \log(p)}_{\text{bits per } l} = 4 \cdot \log(p)^2$$

we can thus store a representation of $f_{r,P}$ as an array of line functions.[3] Hence, we can split up Algorithm 8 into three subroutines Construct, Evaluate, and Exponentiate, which coincide precisely with the decomposition of the Tate pair-

3 The factor $1/3$ can be further reduced using windowing techniques.

ing given in Equation (10). We refer to the composition of these subalgorithms as Scott-Miller's algorithm[4]. See Section B for an algorithmic description.

3.5 SUMMARY OF COSTS

We summarize Section 3 in terms of $\mathbb{F}_p$-operations for pairings of degree $r = p + 1$.

GENERAL PRIMES. MillersAlgorithm has $\log p$ steps and each step performs 1 Dbl. Using the techniques from Section 3.3 to decrease the number of Add/Subs[5]:

1. each Dbl costs $15\mathbf{M} + 5\mathbf{S} + 7\mathbf{A}$, we always perform $\log p$ Dbls
2. each Add or Sub costs $20\mathbf{M} + 4\mathbf{S} + 9\mathbf{A}$,
   a) in a naïve approach, we perform $\frac{1}{2}\log p$ Adds and Subs
   b) using NAFs, we perform $\frac{1}{3}\log p$ Adds and Subs
   c) using windowing, we perform $\frac{1}{w+1}\log p$ Adds and Subs

FOR CSIDH-512. For p512, Table 10 gives the number of $\mathbb{F}_p$-operations to compute a pairing, and shows the effectiveness of the optimizations: a reduction of 40% compared to unoptimized pairings.

	M	**S**	**A**	Total
Original MillersAlgorithm	28 498	2621	39 207	30 987
Optimized step (Sec 3.2)	12 740	3569	12 230	15 717
Using NAFs (Alg. 8)	11 152	3254	11 125	13 866
Using windows ($w = 5$)	9963	2960	10 592	12 436
Additional pairing	4410	2	5704	4468

Table 10: Concrete cost of the Miller loop for p512 for a pairing of degree $r = p + 1$. 'Total' gives the number of $\mathbb{F}_p$-operations, with cost model $1\mathbf{S} = 0.8\mathbf{M}$ and $1\mathbf{A} = 0.01\mathbf{M}$.

4 I noticed only after publication of this chapter that Miller's original article [277] presents a similar approach as Scott does. This seems to have been lost (to some) throughout the many years and improvements applied to Miller's algorithm.

5 These techniques are inspired by finite-field exponentiation and scalar multiplication, as analyzed similarly in [238, 239].

For an additional pairing, if the points are known beforehand, we require only slightly more memory cost for each additional pairing. If we need to compute a multipairing for variable points, this takes $4 \cdot \log(p)^2$ bits of memory to store the representation of $f_{r,P}$, using Scott-Miller's algorithm (Section 3.4).

Remark 2. In Table 10, we consider the cost for a pairing of degree $r = p + 1$. An alternative for primes $p = h \cdot \ell_\chi - 1$ with large cofactor h is to consider pairings of degree $r = \ell_\chi$ on $E[\ell_\chi]$. In many applications, such as described in Chapter VII, this contains all the necessary torsion information needed. The loop, then, has $\log \ell_\chi = \log p - \log h$ steps. The cost of such a pairing can be deduced from the given estimates.

§4 Applications of Pairings to Isogeny Problems

In this section, we apply the optimized pairings from Section 3 to the isogeny problems described in Section 2.1. The core design idea is clear: the pairing is now cheap enough to move isogeny problems from curves to finite fields, where we have efficient Lucas exponentiation. Lemma 1 captures what information about the original points remains after the pairing.

4.1 Verification of full-torsion points

We start with the following problem: Given $P \in E(\mathbb{F}_p)$ and $Q \in E^t(\mathbb{F}_p)$, verify that both P and Q have full-torsion, where E is a supersingular curve over $\mathbb{F}_p$ [6].

current methods. Current methods to verify full-torsion points compute $[\frac{p+1}{\ell_i}]P \neq \mathcal{O}$ to conclude p has ℓ_i-torsion for every $\ell_i \mid p+1$ and hence is a full-torsion point. In a naïve way, this can be done per ℓ_i for the cost of a scalar multiplication of size $\log p$, using either Montgomery ladders or differential addition chains, at a total complexity of $\mathcal{O}(n \log p)$. Concretely, this comes down to a cost of $C_{\text{curve}} \cdot n \log p$ where C_{curve} is the cost of curve arithmetic per bit.

Using product trees, this drops to $\mathcal{O}(\log n \log p)$, although taking $\mathcal{O}(\log n)$ space. Product-tree-based order verification is currently the method used in our deterministic and dummy-free implementation dCSIDH from Chapter VII to verify a given basis (P, Q). Performing this verification for both P and Q comes down to approximately $2 \cdot C_{\text{curve}} \cdot \log n \log p$ operations in $\mathbb{F}_p$.

6 We delay *finding* full-torsion points until Section 4.3.

TORSION BASES. We can improve on the previous verification approach using two easy observations. First, whenever a pair of points $P \in E(\mathbb{F}_p)$ and $Q \in E^t(\mathbb{F}_p)$ generates all of $E[p+1] \subset E(\mathbb{F}_{p^2})$, the pair (P, Q) is a torsion basis for the $(p+1)$-torsion. As described in SIDH/SIKE [130, 131], we can verify such a torsion basis using the result that $\zeta = t_{p+1}(P, Q) \in \mu_{p+1}$ must be a primitive $(p+1)$-th root of unity in $\mathbb{F}_{p^2}$. Second, this situation is ideal for our pairing: P has only rational coordinates, and Q has rational x and purely imaginary y-coordinate. As noted before, ζ is an element of norm 1 in $\mathbb{F}_{p^2}$, which allows us to apply fast Lucas exponentiation to compute ζ^k. The following lemma is our key building block.

Lemma 2. Let $P \in E(\mathbb{F}_p)$ and $Q \in E^t(\mathbb{F}_p)$. Let $\zeta = t_{p+1}(P, Q) \in \mu_{p+1}$. Then

$$\zeta^{\frac{p+1}{\ell_i}} \neq 1 \Leftrightarrow [\frac{p+1}{\ell_i}]P \neq \mathcal{O} \text{ and } [\frac{p+1}{\ell_i}]Q \neq \mathcal{O}.$$

Proof. This is a direct application of Lemma 1. □

Hence, instead of verifying that both P and Q have ℓ_i-torsion, we verify that the powers $\zeta^{\frac{p+1}{\ell_i}}$ do not vanish. Furthermore, as ζ and its powers have norm 1, we simply verify that $\mathrm{Re}(\zeta^k) \neq 1$ which implies $\zeta^k \neq 1$. In terms of Lucas sequences, this is equal to $V_k(2a) \neq 2$ for $\zeta = a + bi$. Per bit, Lucas exponentiation is much more efficient than curve arithmetic, which allows us to compute every $\zeta^{\frac{p+1}{\ell_i}}$ (again using product trees) very efficiently at a similar complexity $\mathcal{O}(\log n \log p)$, and a concrete cost [7] of $1 \cdot C_{\text{Lucas}} \cdot \log n \log p$. Algorithm 9 summarizes this approach, where Order is a product-tree based function that computes the order of ζ (equivalently, verifies for which ℓ_i we have $\zeta^{\frac{p+1}{\ell_i}} \neq 1$) using Lucas exponentiation. Order is the field analogue of algorithms such as OrderRec [23] and validate_rec in CTIDH [20]. See bigpairingfo.rs for an implementation of Algorithm 9.

FURTHER IMPROVEMENTS. As stated before, working in μ_{p+1} has the added benefit that we work in same subgroup of $\mathbb{F}_{p^2}$, independent of the curve E_A. This can be used to speed up the cost of torsion basis verification even more, at the cost of $\log p$ additional bits next to the pair (P, Q), as follows.

The scalars $\Lambda := \mathbb{Z}_{p+1}^*$ of invertible elements in $\mathbb{Z}_{p+1}$ act faithfully and free on both the group of full-torsion points of E_A, as well as on the primitive

7 To compute using Lucas exponentiation we use a constant-time laddering approach. Interesting future work would be to use (differential) addition chains to reduce costs.

Algorithm 9 Verification of torsion basis

Input: $P \in E(\mathbb{F}_p)$, $Q \in E^t(\mathbb{F}_p)$
Output: **True** or **False**
1: $\zeta \leftarrow \mathrm{Re}(t_{p+1}(P, Q))$
2: $m \leftarrow \mathsf{Order}(\zeta)$
3: **if** $m = p + 1$ **then**
4: **return True**
5: **else**
6: **return False**

roots in $\mu_{p+1} \subset \mathbb{F}_{p^2}$. This means that we can write any full-torsion point T relative to another full-torsion point T' as $T = [\lambda]T'$ with $\lambda \in \Lambda$. Similarly for primitive roots, this implies we can pick a system parameter ζ_0 as the "standard" primitive root, and can find $\lambda \in \Lambda$ such that for $\zeta = t_{p+1}(P, Q)$ we get $\zeta = \zeta_0^\lambda$.

By including λ next to (P, Q), we do not have to verify the complete order of ζ. Instead, we simply verify that $\lambda \in \Lambda$, compute $\zeta = t_{p+1}(P, Q)$ and verify that $\zeta = \zeta_0^\lambda$ using only a single Lucas exponentition to verify that ζ is an exact $(p + 1)$-th root of unity. Compared to the algorithm sketched before the cost decreases from $\mathcal{O}(\log n \log p)$ to $\mathcal{O}(\log p)$ to verify the order of ζ.

Remark 3. The addition of the discrete log λ such that $\zeta = t_{p+1}(P, Q) = \zeta_0^\lambda$ might be unnecessary depending on the specific application. Namely, for a pair (P, Q), we get another pair $(P, \lambda^{-1}Q)$ that is a torsion basis, with

$$t_{p+1}(P, \lambda^{-1}Q) = t_{p+1}(P, Q)^{\lambda^{-1}} = \zeta^{\lambda^{-1}} = \zeta_0.$$

As ζ_0 is a public parameter, verification requires no extra λ. Simply providing $(P, \lambda^{-1}Q)$ as the torsion basis then only requires the computation of $t_{p+1}(P, \lambda^{-1}Q)$ to verify the basis. However, the choice of P and Q might have been performed carefully, e.g. to make sure both have small x-coordinates to reduce communication cost. It thus depends per application if a modified torsion basis $(P, \lambda^{-1}Q)$ reduces communication cost.

Remark 4. The above algorithm not only verifies that P and Q are full-torsion points, but includes the supersingularity verification of E_A, as it shows E_A has points of order $p + 1$. This can be useful for the dCSIDH implementation from Chapter VII.

4.2 PAIRING-BASED SUPERSINGULARITY VERIFICATION

Supersingularity verification asks us to verify that E_A is a supersingular curve for a given $A \in \mathbb{F}_p$. A sound analysis of the performance of different algorithms was made by Banegas, Gilchrist, and Smith [23]. They examine

a. a product-tree-based approach to find a point of order $N \geq 4\sqrt{p}$,

b. Sutherland's algorithm [361] based on isogeny volcanoes, and

c. Doliskani's test [161] based on division polynomials.

They conclude that Doliskani's test is best for Montgomery models over $\mathbb{F}_p$, as it requires only a single scalar multiplication over $\mathbb{F}_{p^2}$ of length $\log p$ followed by $\log p$ squarings in $\mathbb{F}_{p^2}$. The algorithms we propose resemble the product-tree approach, but move the computation of the orders of points from the curve to the field. There are two ways to apply the pairing approach here:

Approach 1. Sample, using Elligator [46], random points $P \in E(\mathbb{F}_p)$ and $Q \in E^t(\mathbb{F}_p)$, compute $\zeta = t_r(P, Q)$ for $r = p + 1$, and compute $\mathrm{Ord}(\zeta)$ using a product-tree up until we have verified $\mathrm{Ord}(\zeta) \geq 4\sqrt{p}$.

Approach 2. Divide L_χ into two lists L_1 and L_2 such that $\ell^{(1)} := \prod_{\ell_i \in L_1} \ell_i$ is slightly larger than $4\sqrt{p}$. Sample again two random points $P \in E(\mathbb{F}_p)$ and $Q \in E^t(\mathbb{F}_p)$, and multiply both by $\frac{p+1}{\ell^{(1)}} = h \cdot \ell^{(2)}$ so that $P, Q \in E[\ell^{(1)}]$. Then, compute the pairing of degree $r = \ell^{(1)}$ and verify that $\zeta \in \mu_r$ has $\mathrm{Ord}(\zeta) \geq 4\sqrt{p}$.

Approach 1 is essentially Algorithm 9, where we cut off the computation of $\mathrm{Ord}(\zeta)$ early whenever we have enough torsion. Approach 2 uses the fact that we do not have to work in all of μ_{p+1} to verify supersingularity. This reduces the number of steps in the Miller loop roughly by half, $\frac{1}{2}\log p$ compared to $\log p$, but requires two Montgomery ladders of $\log(p)/2$ bits to kill the $\ell^{(2)}$-torsion of P and Q. Note that we must take L_1 a few bits larger than $4\sqrt{p}$ to ensure with high probability that random points P, Q have enough torsion to verify or falsify supersingularity. In practice, the fastest approach is highly dependent on the prime p and the number of factors ℓ_i, as well as the size of the cofactor 2^k. Approach 2 is summarized in Algorithm 10. See the folder `supersingularity` for the implementations of these algorithms.

Algorithm 10 Verification of supersingularity

Input: $A \in \mathbb{F}_p$, where $p = h \cdot \ell^{(1)} \cdot \ell^{(2)} - 1$
Output: **True** or **False**

1: $(P, Q) \xleftarrow{\$} E(\mathbb{F}_p) \times E^t(\mathbb{F}_p)$
2: $P \leftarrow [h \cdot \ell^{(2)}]P, \ Q \leftarrow [h \cdot \ell^{(2)}]Q$
3: $\zeta \leftarrow \mathrm{Re}(t_{\ell^{(1)}}(P, Q))$
4: $m \leftarrow \mathsf{Order}(\zeta)$
5: **if** $m \geq 4\sqrt{p}$ **then**
6: **return True**
7: **else**
8: **else** repeat

Remark 5. Line 4 of Algorithm 10 computes the order of ζ using a product-tree approach to verify **(a)** $\zeta' := \zeta^{\frac{p+1}{\ell_i}} \neq 1$ and **(b)** $\zeta'^{\ell_i} = 1$. If at any moment, **(a)** holds but **(b)** does not, this implies the order of ζ does not divide $p+1$, which implies the curve is *ordinary*. In such a case, as in the original algorithm [95, Alg. 1], we return **False** immediately as we know the curve is not supersingular.

Remark 6. Note that in an approach where torsion points (P, Q) are given, together with a discrete log λ, so that $t_{p+1}(P, Q) = \zeta_0^{\lambda}$, or even the variant where $(P, \lambda^{-1}Q)$ is given as in Remark 3, the cost of supersingularity verification is essentially that of a single pairing computation, or that of a pairing computation together with a Lucas exponentiation of length $\log \lambda \approx \log p$. For CSIDH-512, this beats Doliskani's test as we will see in Section 4.4

4.3 FINDING FULL-TORSION POINTS

Finding full-torsion points is more tricky. In Chapter VII we propose simply sample random points and compute their order until we find a full-torsion point. Although the probability of finding a full-torsion point is not low, this approach is rather inefficient as the cost of computing the order per point is similar to the one sketched in Section 4.1. The curve-equivalent of GaussAlgorithm can be given as follows: Given a point $P \in E(\mathbb{F}_p)$ with $\mathrm{Ord}(P) \neq p+1$, sample a random point $P' \in E(\mathbb{F}_p)$. Set $P' \leftarrow [\mathrm{Ord}(P)]P'$ and compute $\mathrm{Ord}(P')$. Set $P \leftarrow P + P'$ and $\mathrm{Ord}(P) \leftarrow \mathrm{Ord}(P) \cdot \mathrm{Ord}(P')$. Repeat until P is a full-torsion point. This already improves on the naïve approach, yet still requires a lot of curve arithmetic. We apply pairings again to improve performance. We

specifically need Scott-Miller's algorithm (Section 3.4) to compute a variable number of pairings for a given P.

Abstract point-of-view.

From the abstract point of view, sketched in Section 3.1, we want to identify a full-torsion point P as an isomorphism $f_{r,P} : E^t(\mathbb{F}_p) \to \mu_r$, using Lemma 1. Scott-Miller's algorithm allows us to compute a representation of $f_{r,P}$ and to evaluate $f_{r,P}$ efficiently on points $Q \in E^t(\mathbb{F}_p)$.

Starting from random points $P_1 \in E(\mathbb{F}_p)$ and $Q_1 \in E^t(\mathbb{F}_p)$, we compute $\zeta_1 := f_{r,P_1}(P_1, Q_1)^{p-1}$ and $\mathrm{Ord}(\zeta_1)$. The missing torsion $m_1 = \mathrm{Miss}(\zeta_1)$ is then equal to $\mathrm{lcm}(\mathrm{Miss}(P), \mathrm{Miss}(Q)))$. If $m_1 = 1$, then we know both P_1 and Q_1 are full-torsion points. If $m_1 \neq 1$, we continue with a second point Q_2. Compute ζ_2 and $m_2 = \mathrm{Miss}(\zeta_2)$. Let $d = \gcd(m_1, m_2)$. If $d = 1$, that is, m_1 and m_2 are co-prime, then P_1 is a full-torsion point, and we apply GaussAlgorithm to compute a full-order point Q, given Q_1 and Q_2.

For $d > 1$, it is most likely that $d = |\ker f_{rP_1}| = \mathrm{Miss}(P_1)$, or, if unlucky, both Q_1 and Q_2 miss d-torsion. The probability that both Q_1 and Q_2 miss d-torsion is $\frac{1}{d^2}$. Hence, if d is small, this is unlikely but possible. If d is a large prime, we are almost certain P_1 misses d-torsion. In the former case, we sample a third point Q_3 and repeat the same procedure. In the latter case, we use Q_1 and Q_2 to compute a full-torsion Q. Using Q, we compute $f_{r,Q}$ and apply the same procedure to points P_i to create a full-torsion point P (reusing P_1).

Distinguishing between these cases is highly dependent on the value of d, which in turn depends on L_χ and p. We leave these case-dependent details to the reader. See bigfastfinding.rs for the implementation of this algorithm.

Remark 7. To distinguish between the cases dependent on d, it is favorable to have no small factors ℓ_i in $p + 1$. This coincides with independent analysis [20, 102] that small $\ell_i \mid p + 1$ might not be optimal for deterministic variants of CSIDH. We have seen similar intuition in Chapter VII.

Remark 8. One can improve on randomly sampling P_1 and Q_1, as we can sample points directly in $E \setminus [2]E$ [130] by ensuring the x-coordinates are not quadratic residues in $\mathbb{F}_p$. Similar techniques from p-descent apply for $\ell_i > 2$.

Remark 9. The above approach is very efficient to find a second full-torsion point $Q \in E^t(\mathbb{F}_p)$ given a first full-torsion point $P \in E(\mathbb{F}_p)$, which may be of independent interest in other situations.

4.4 CONCRETE COST FOR CSIDH-512

We have implemented and evaluated the performance of the algorithms in Section 4 for p512, the prime used in CSIDH-512. Table 11 shows the performance in $\mathbb{F}_p$-operations, compared to well-known or state-of-the-art algorithms.

	Source	**M**	**S**	Total
Product-tree torsion verif.	[77]	51 318	29 388	75 562
Pairing-based torsion verif.	Alg. 9	13 693	6838	19 293
Pairing-based (given λ)	Sec. 4.1	10 472	3471	13 364
CSIDH-Supersingularity verif.	[95]	13 324	7628	19 617
Doliskani's test	[23, 161]	13 789	2	14 097
Approach 1 (pairing-based)	Sec. 4.2	11 081	4112	14 500
Approach 2 (pairing-based)	Alg. 10	9589	5801	14 334

Table 11: Concrete cost of the algorithms in this section using the prime in CSIDH-512. 'Total' gives the number of $\mathbb{F}_p$-operations, with cost model $1\mathbf{S} = 0.8\mathbf{M}$ and $1\mathbf{A} = 0.01\mathbf{M}$.

VERIFYING TORSION POINTS. We find that Algorithm 9 specifically for p512 takes about 19293 operations, with 12426 operations taken up by the pairing, hence order verification of ζ using Lucas exponentiation requires only 6867 operations, closely matching the predicted cost $C_{\text{Lucas}} \cdot \log n \cdot \log p$. In comparison, the currently-used method [77] to verify full-torsion points requires 75562 operations, hence we achieve a speed-up of 75%, due to the difference in cost per bit between C_{curve} and C_{Lucas}. If we include a system parameter ζ_0 and a discrete log λ, our cost drops down to 13364 operations, increasing the speedup from 75% to 82%.

SUPERSINGULARITY VERIFICATION. We find that Doliskani's test is still slightly faster, but our algorithms come within 2% of performance. Saving a single M or S in Dbl or Add would push Algorithm 10 below Doliskani's test for p512. When we include λ, as mentioned above, we outperform Doliskani's test by 6%.

FINDING TORSION POINTS. Although the cost of this algorithm depends highly on divisors ℓ_i of $p+1$, heuristics for p512 show that we usually only require 2 points $P_1, P_2 \in E(\mathbb{F}_p)$ and two points $Q_1, Q_2 \in E^t(\mathbb{F}_p)$, together with

pairing computations $t_r(P_1, Q_1)$ and $t_r(P_1, Q_2)$ to find full-torsion points P and Q. We leave out concrete performance numbers, as this varies too much per case.

§5 Applications of Pairing-based Algorithms

The pairing-based algorithms from Section 4 are of independent interest, but also find natural applications in (deterministic) variants of CSIDH.

Applying pairing-based algorithms.

In all versions of CSIDH, supersingularity verification is required on public keys. We estimate that, depending on the shape and size of the prime p, either Doliskani's test or one of the pairing-based algorithms (Section 4.2) is optimal.

For deterministic variants of CSIDH, such as dCSIDH, including (an Elligator seed for) a torsion basis of E_A in the public key is only natural. This requires verification of such a torsion basis. For this verification, Algorithm 9 clearly outperforms curve-based approaches. Furthermore, the verification of such a torsion basis also verifies the supersingularity of E_A, which would otherwise have cost an additional $\mathcal{O}(\log p)$ operations, using either Doliskani's test or one of our pairing-based algorithms.

Including torsion-point information in the public key also requires a party to *find* such a torsion basis in key generation. The pairing-based approach described in Section 4.3 heuristically beats current approaches based on random sampling.

Remark 10. One might think that including a torsion basis (P, Q) resulting from the pairing-based approach in a public key would cost $2 \log p$ bits, as it requires a description of the x-coordinates of both points. However, it is possible to use points P_i and Q_i with very small x-coordinates, and to describe the construction of P (resp. Q) as a combination of the P_i (resp. Q_i) in a few bits.

Constant-time versions.

Both Algorithms 9 and 10 are easy to implement in constant-time, given constant-time curve and field arithmetic. However, constant-time verification is usually not required, as both E_A and (P, Q) should be public.

For a constant-time approach to finding full-torsion points, a major roadblock is finding a constant-time version of GaussAlgorithm. This is both mathematically

interesting as well as cryptographically useful, but seems to require a better understanding of the distribution of the x-coordinates of full-torsion points for curves, or the $(p+1)$-th primitive roots for fields.

Beyond current implementations.

Current deterministic variants of CSIDH, such as dCSIDH, are limited to exponents $e_i \in \{-1, 0, +1\}$. Going beyond such exponents requires sampling new points during the class group action evaluation on an intermediate curve E'. Therefore, to not leak any information on E' in a deterministic implementation of CSIDH, we need a constant-time basis-sampling algorithm as sketched above. This would allow approaches with $e_i \geq 1$ for multiple small ℓ_i to reduce the number of ℓ_i-isogenies for large ℓ_i, which is deemed favorable in constant-time probabilistic approaches [20, 107, 274].

For ordinary CSIDH and CTIDH, using full-torsion points in every round would have the further improvement that the number of rounds is constant, and we have no trial-and-error approaches in the group action computation, providing a stronger defence against physical attacks, such as the one described in Chapter V. In particular for CTIDH, we can use full-torsion points to remove the "coin flip" that decides if a batch is performed. This improves performance and design properties. A full analysis is future work.

A final remark should be made about the use of theta functions to compute pairings [260, 262]. Such an approach might yield speed-ups in comparison to the methods used in this work. Furthermore, the use of theta functions provides a more general framework for higher-dimensional isogeny-based cryptography.

§A Subalgorithms of Miller's Algorithm

The following algorithms give an algorithmic description of the subroutines Dbl, Add and Sub as used in the optimizations of the Miller loop.

Algorithm 11 Subroutine Dbl in the Miller loop

Input: $T \in \mathbb{F}_p^4, f \in \mathbb{F}_p^2, x_Q, y_Q \in \mathbb{F}_p, A \in \mathbb{F}_p$
Output: (T, f) corresponding to $T \leftarrow T + T$, $f \leftarrow f^2 \cdot \ell_{T,T}(Q)$
1: $(T, \ell) \leftarrow \mathsf{DblAndLine}(T, A)$ ▷ Doubles T and computes $\ell_{T,T}$
2: $(\alpha, \beta) \leftarrow \mathsf{Eval}(\ell, x_Q, y_Q)$ ▷ Evaluates $\ell_{T,T}(Q)$
3: $f \leftarrow f^2$
4: $f \leftarrow f \cdot (\alpha + \beta \cdot i)$
5: **return** T, f

Algorithm 12 Subroutine Add in the Miller loop

Input: $T \in \mathbb{F}_p^4, f \in \mathbb{F}_p^2, x_P, y_P, x_Q, y_Q \in \mathbb{F}_p, A \in \mathbb{F}_p$
Output: (T, f) corresponding to $T \leftarrow T + P$, $f \leftarrow f \cdot \ell_{T,P}(Q)$
1: $(T, \ell) \leftarrow \mathsf{AddAndLine}(T, x_P, y_P, A)$ ▷ Adds $T + P$ and computes $\ell_{T,P}$
2: $(\alpha, \beta) \leftarrow \mathsf{Eval}(\ell, x_Q, y_Q)$ ▷ Evaluates $\ell_{T,P}(Q)$
3: $f \leftarrow f \cdot (\alpha + \beta \cdot i)$
4: **return** T, f

Algorithm 13 Subroutine Sub in the Miller loop

Input: $T \in \mathbb{F}_p^4, f \in \mathbb{F}_p^2, x_P, y_P, x_Q, y_Q \in \mathbb{F}_p, A \in \mathbb{F}_p$
Output: (T, f) corresponding to $T \leftarrow T - P$, $f \leftarrow f \cdot \ell_{T,-P}(Q)$
1: $(T, f) \leftarrow \mathsf{Add}(T, f, x_P, y_P, x_Q, y_Q, A)$
2: **return** T, f

§B Subalgorithms of Scott-Miller's Algorithm

We describe here Scott-Miller's subalgorithms Construct, Evaluate and Exponentiate.

Algorithm 14 Scott-Miller's subalgorithm Construct

Input: $x_P, y_p \in \mathbb{F}_p$, $p+1 = \sum_{i=0}^{t} t_i \cdot 2^i$
Output: A representation of $f_{r,P}$ as an array of line functions

$T = (X^2, XZ, Z^2, YZ) \leftarrow (x_P^2, x_P, 1, y_P)$
$f \leftarrow [\,]$
for i from $t-1$ to 0 **do**
    $T, l \leftarrow \mathsf{Dbl}(T)$
    Append l to f
    if $t_i = 1$ **then**
        $T, l \leftarrow \mathsf{Add}(T, f, x_P, y_P)$
        Append l to f
    if $t_i = -1$ **then**
        $T, l \leftarrow \mathsf{Sub}(T, f, x_P, y_P)$
        Append l to f
return f

Algorithm 15 Scott-Miller's subalgorithm Evaluate

Input: A representation of $f_{r,P}$ as an array of line functions, and $x_Q, y_Q \in \mathbb{F}_p$
Output: The unreduced Tate evaluation $f_{r,P}(Q) \in \mathbb{F}_{p^2}$

$f_0 \leftarrow (1, 0)$
for l in f **do**
    if l is a doubling **then** $f_0 \leftarrow f_0^2$
    $f_0 \leftarrow f_0 \cdot \mathsf{Eval}(l, x_Q, y_Q)$
return f_0

Algorithm 16 Scott-Miller's subalgorithm Exponentiate

Input: The unreduced Tate pairing $f_0 = a + bi$ as a pair $f = (a, b)$
Output: The reduced Tate pairing $\zeta \in \mu_r$
1: $a \leftarrow f[0]$, $b \leftarrow f[1]$
2: $\zeta \leftarrow \frac{a^2-b^2}{a^2+b^2}$
3: **return** ζ

The composition of these three algorithms is referred to as Scott-Miller's algorithm.

A VIEW TO THE HORIZON

Chapter VII draws a rather bleak picture of the applicability of CSIDH. However, cryptographic research can be surprising and we may never know what lies ahead of us, especially with the speed at which isogeny-based cryptography is currently moving. We describe a few movements that may impact the horizon of CSIDH.

Radical Isogenies.

In Chapter VI, we have shown that radical isogenies, as introduced by Castryck, Decru, and Vercauteren [92], were uneffective to improve the performance of constant-time CSIDH. Since then, several works [89, 91, 154, 302, 317] have improved the performance and degrees of radical isogenies. It is an interesting question to try to combine these newer radical isogenies with a state-of-the-art implementation of constant-time CSIDH, where the techniques proposed in Chapter VI can be re-used to achieve projective formulas. However, many of the arguments raised in Chapter VI still stand, and we see no straightforward solution. On the contrary, CTIDH [20] significantly improved the key space for CSIDH and the usage of radical isogenies goes directly against this improvement in key space.

Improved Keyspaces.

The new key space introduced by CTIDH [20] is far better than the original key space introduced in CSIDH, mainly enabled by a clever use of Matryoshka isogenies [49]. Further explorations and improvements to this key space certainly seem possible, in particular for large-prime instantiations of CSIDH. We started an initial exploration of such large-prime instantiations in Chapter VII and future work suggests positive results in this direction.

Other Class Groups.

We can achieve an *effective* group action using CSI-FiSh [54] for CSIDH-512, or, more scalable, SCALLOP [143] and SCALLOP-HD [104], or even any class group using Clapoti(s) [304]. However, most of these approaches do not seem

to outperform CSIDH in terms of speed, and it seems that we are a still a breakthrough (or two) away from using such class group actions in post-quantum NIKEs. Beyond NIKEs, an exploration of their usefulness in other primitives [54, 144, 150, 155, 250] would be very interesting.

Supersingularity Verification.

The improved supersingularity verification algorithm VeriFast always seemed of relatively little importance to me beyond large-paramter CSIDH, due to the requirement of more than $\log p/2$ bits of rational $2^\bullet$-torsion. However, the higher-dimensional breakthrough requires similar amounts of rational $2^\bullet$-torsion in its primes, and so, many protocols based on higher-dimensional techniques, such as SQIsignHD [141] or SQIsign2D [34, 288], also use primes with this requirement. It is simple to adapt VeriFast to these cases, or, using pairing-based techniques, to similar cases where the rational $2^\bullet$-torsion has a slightly different structure. Thus, VeriFast may surprisingly appear in higher-dimensional isogeny-based cryptography in future applications that require supersingularity verification.

We may therefore draw the following conclusion for this part of the thesis: CSIDH, deservedly, received a major amount of research attention in recent years. Yet, we must still classify CSIDH as 'unpractical' for real-world scenarios, mainly due to its debated quantum security and the resulting slow performance of large parameter sets. CSIDH-based KEMs seem to require a higher-dimensional miracle, such as we have seen recently for isogeny-based signatures, or an improved understanding of the plausability of quantum threats to be classified as 'practical'.

Part 2

ISOGENIES (SQISIGN)

THE LANDSCAPE OF SQISIGN

Don't you ever say,
Torsion is okay,
They will always hurt you.

The SICQ.

At the start of this thesis, SQIsign simply did not exist yet.

However, the crucial building blocks were there already, introduced in the KLPT algorithm [242] and the GPS signature scheme [186]. Soon enough, De Feo, Kohel, Leroux, Petit, and Wesolowski [147] introduced the basis of the signature scheme SQIsign. With an impressively small signature and public-key size, SQIsign is an ideal candidate for post-quantum signatures, if only signing and verification were not so slow. Establishing SQIsign required the development of many new mathematical tools and techniques, especially in quaternion algebras, and so, it took a while before research in SQIsign picked up its pace. Two improved variants [149, 258] appeared, both improving signing and verification times. This created more interest in SQIsign: if verification was only *fast enough*, many applications could suffer the slow signing, in particular in use cases that require many devices to verify the same signature, such as TLS certificates.

In Chapter IX, we ask ourselves how fast verification could potentially be for SQIsign, by pushing prime characteristics to its limits and exploring size-speed trade-offs that may be worthwhile for several important applications. We use extension-field arithmetic to enable signing for such primes, and thoroughly analyse and improve the verification procedure of SQIsign by significantly improving several imporant subroutines, resulting in a SQIsign-variant we call AprèsSQI. We show that the synergy between these high-level and low-level improvements gives significant verification improvements, with speed-ups between 2.07 times and 3.41 times for AprèsSQI depending on size-speed trade-offs, compared to the state of the art, without majorly degrading the performance of signing.

The advent of higher-dimensional isogenies [86, 266, 329, 330] ushered in an HD-variant: SQIsignHD [141]. The higher-dimensional method allows for

much faster signing times, but verification suffers: the already relatively-slow verification of SQIsign becomes unpracticaly slow in SQIsignHD for most applications, mainly because SQIsignHD requires verification using 4-dimensional isogenies. Nevertheless, the potential was there to allow 2-dimensional verification, if the stars aligned correctly. With an new technique by Nakagawa and Onuki [287], three separate works were ablo to achieve 2-dimensional verification: SQIsign2D-West [34], SQIsign2D-East [288], and SQIprime [166]. As a result, 2D-SQIsign achieves fast-ish verification with much better signing times than SQIsign, in particular compared to the results in Chapter IX. However, AprèsSQI lacks a 'proper' implementation and so comparing the performance of one-dimensional and two-dimensional SQIsign is a difficult task.

In Chapter X, we aim to answer precisely that question. By providing highly-optimized field arithmetic for Intel and Cortex-M4 architectures and improved subroutines for one-dimensional SQIsign verification, we are able to achieve verification times for compressed SQIsign in 8.56 MCycles, and uncompressed SQIsign in 5.62 MCycles. Furthermore, one-dimensional SQIsign verification seems ideally suited for parallisation and we achieve parallelised compressed verification in 2.05 MCycles and parallelised uncompressed verification in 1.33 MCycles.

This implies that, at the moment of writing this thesis, SQIsign-2D has (much) better performance for signing, yet compressed verification seems comparable between both variants. Furthermore, uncompressed and parallelised verification can give one-dimensional SQIsign a definite edge in applications where signing time and signature size are of lesser importance than verification time. Recent developments [303] show potentially significant improvements for one-dimensional signing, although it is unlikely that these results can match two-dimensional signing anytime soon. Improvements from this chapter can be used to improve two-dimensional SQIsign too.

Lastly, in Chapter XI, we explore an esoteric idea: Costello [125] shows that the one-dimensional isogeny between elliptic curves over $\mathbb{F}_{p^2}$ in SIDH can also be computed as a two-dimensional isogeny between Kummer surfaces over $\mathbb{F}_p$, using Scholten's construction [338]. Similarly, the one-dimensional verification of SQIsign over $\mathbb{F}_{p^2}$ might potentially be computed as a two-dimensional isogeny over $\mathbb{F}_p$. To achieve this, we develop several essential algorithms for genus-2 cryptography and, in turn, advance pairing-based techniques to enable these algorithms efficiently. With SQIsign as an example and/or goal, we explore the rich, vast and bewildering literature of Kummer surfaces and their $(2,2)$-isogenies and expand the machinery of isogeny-based cryptography in genus 2. Contrary to (our own) expection, the resulting SQIsign verification

on Kummer surfaces is nearly as efficient as the verification on elliptic curves, even though many of the introduces techniques are completely new and have the potential to be further optimized.

IX

APRÈSSQI: FAST VERIFICATION FOR SQISIGN

In this chapter, we optimise verification of the SQIsign signature scheme. By using field extensions during signing, we are able to significantly increase the amount of available rational 2-power torsion in verification, which allows for a significant speed-up. Moreover, this allows for several other speed-ups on the level of curve arithmetic. We show that the synergy between these high-level and low-level improvements gives significant improvements, making verification 2.07 times faster, or up to 3.41 times when using size-speed trade-offs, compared to the state of the art, without majorly degrading the performance of signing.

§1 Introduction

Research has shown that large-scale quantum computers will break current public-key cryptography, such as RSA or ECC, whose security relies on the hardness of integer factorization or the discrete logarithm [351]. Post-quantum cryptography seeks to thwart the threat of quantum computers by developing cryptographic primitives based on alternative mathematical problems that cannot be solved efficiently by quantum computers. In recent years, lattice-based

This chapter is based on the paper

> Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Krijn Reijnders. "AprèsSQI: extra fast verification for SQIsign using extension-field signing". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2024, pp. 63–93.

I have only removed a few tables in the appendix describing in which field extension torsion lives, as these take up a major amount of space for little additional understanding. Notably, I have not changed the results on basis generation to include the improved understanding of the role of the Tate pairing developed in later chapters. This is left as a (fun!) exercise for the reader.

cryptography has developed successful post-quantum schemes for essential primitives such as key encapsulation mechanisms (KEMs) and digital signatures that will be standardized by NIST. Lattice-based signatures are able to provide fast signing and verification, but have to resort to larger key and signature sizes than were previously acceptable in pre-quantum signatures. For applications where the amount of data transmitted is crucial, these lattice-based schemes may not be practical. NIST is therefore looking for other signature schemes with properties such as smaller combined public-key and signature sizes to ensure a smooth transition to a post-quantum world [357].

A potential solution to this problem is provided by the sole isogeny-based candidate in NIST's new call for signatures: SQIsign [148]. This is currently the candidate that comes closest to the data sizes transmitted, i.e. the combined size of the signature and the public key, in pre-quantum elliptic-curve signatures [224, 225]. SQIsign is most interesting in scenarios that require small signature sizes and fast verification, particularly in those applications where the performance of signing is not the main concern. A few common examples include long-term signatures, specifically public-key certificates, code updates for small devices, authenticated communication with embedded devices, or other microcontrollers that solely run verification. For such use cases it is imperative to explore the lower limits of the cost of verification as much as possible.

PERFORMANCE BOTTLENECKS IN SQISIGN. The bottleneck in SQIsign verification is the computation of an isogeny of fixed degree 2^e with $e \approx (15/4)\log(p)$, with p a prime of roughly 2λ bits, so that $\log(p) \approx 256$ for NIST Level I security. However, the rational 2-power torsion is limited, since we work with supersingular elliptic curves over $\mathbb{F}_{p^2}$. In this case, as the rational 2-power torsion is equal to the largest f such that $2^f \mid p+1$, we find an upper bound of $f = \log p$. Therefore, the verifier needs to perform $\lceil \frac{e}{f} \rceil$ *blocks*, isogenies of degree 2^f, to complete the full isogeny of degree 2^e. Each of these blocks involves costly steps such as computing a torsion basis for $E[2^f]$ and computing the kernel generator for a block. Hence, in general, a smaller number of blocks will not decrease the cost for pure isogeny computations, but *should* decrease the number of times we require computing torsion bases or kernel generators, and thus improve the overal performance of verification.

However, we cannot simply increase f to $\log p$. The bottleneck in signing is the computation of several T-isogenies for odd smooth $T \approx p^{5/4}$. Current implementations of SQIsign therefore require $T \mid (p-1)(p+1)$, such that

$\mathbb{F}_{p^2}$-rational points are available for efficient T-isogeny computations. The performance of this step is dominated by the smoothness of T, i.e., its largest prime factor. In theory, this requirement on p theoretically limits the maximal f we can achieve to roughly $3/4 \log p$. In practice, current techniques for finding such *SQIsign-friendly* primes suggest that achieving $f = 3/4 \log p$ with acceptable smoothness of T is infeasible [75, 100, 124, 134, 148]. In particular, the NIST submission of SQIsign uses a prime with $f = 75$ and 1973 as the largest factor of T. Since $e \approx (15/4) \cdot 256 = 960$, the verifier has to perform $\lceil e/f \rceil = 13$ costly isogeny blocks. Increasing the 2^f-torsion beyond 75 is difficult as it decreases the probability of finding a smooth and large enough T for current implementations of SQIsign, severely penalizing signing.

Contributions

In this work, we deploy a range of techniques to increase the 2^f-torsion and push the SQIsign verification cost far below the state of the art. Alongside these technical contributions, we aim to give an accessible description of SQIsign, focusing primarily on verification, which solely uses elliptic curves and isogenies and does not require knowledge of quaternion algebras.

Next to targeting faster verification, a major contribution is signing with field extensions. This allows us to get a much weaker requirement on the prime p, which in turn enables us to increase the size of the 2^f-torsion. Focusing on NIST Level I security, we study the range of possible 2^f-torsion to its theoretical maximum $f = \log p$, and measure how the size of f correlates to verification time through an implementation that uses an equivalent to the number of field multiplications as cost metric. Compared to the state of the art, increasing the 2^f-torsion alone makes verification almost 1.7 times faster. Furthermore, we implement our new signing procedure as a proof of concept in SageMath and show that signing times using field extensions are in the same order of magnitude as signing times using only $\mathbb{F}_{p^2}$-operations.

For verification, in addition to implementing some known general techniques for improvements compared to the reference implementation provided in the NIST submission of SQIsign, we show that increasing the 2^f-torsion also opens up a range of optimisations that were previously not possible. For instance, large 2^f-torsion allows for an improved challenge-isogeny computation and improved basis and kernel generation. Furthermore, we show that certain size-speed trade-offs, previously explored in [148], become especially worthwhile for large 2^f-torsion. When pushing the 2^f-torsion to its theoretical maximum,

this even allows for *uncompressed* signatures, leading to significant speed-ups at the cost of roughly doubling signature size.

For two specific primes, with $f = 145$ and $f = 246$, we combine all these speed-ups, and measure the performance of verification in terms of finite-field operations. Compared to the implementation of the SQIsign NIST submission [100], we reach a speed-up up to a factor 2.70 at NIST Level I when keeping the signature size of 177 bytes. When using our size-speed trade-offs, we reach a speed-up by a factor 3.11 for signatures of 187 bytes, or a factor 4.46 for uncompressed signatures of 322 bytes. Compared to the state of the art [258], these speed-ups are factors 2.07, 2.38 and 3.41 respectively.

Related work.

De Feo, Kohel, Leroux, Petit, and Wesolowski [148] introduced SQIsign and provided a first implementation, later supersededy by [149]. Subsequently, Lin, Wang, Xu, and Zhao [258] introduced several improvements for this implementation. The NIST submission of SQIsign [100] features a new implementation that does not rely on any external libraries. Since this is the latest and best documented implementation, we will use it as a baseline for performance comparison, and refer to it as SQIsign(NIST). Since the implementation by Lin, Wang, Xu, and Zhao [258] is not publicly available, we included their main improvement for verification in SQIsign(NIST), and refer to this as SQIsign(LWXZ).

Dartois, Leroux, Robert, and Wesolowski [141] recently introduced SQIsignHD, which massively improves the signing time in SQIsign, in addition to a number of other benefits, but at the cost of a significant slowdown in verification. This could make SQIsignHD an interesting candidate for applications that prioritise the combined cost of signing and verification over the sole cost of verification.

Recent work by Eriksen, Panny, Sotáková, and Veroni [168] explored the feasibility of computing the Deuring correspondence (see Section 5) for *general* primes p using extension fields. We apply the same techniques tailored to *specialised* primes for use in our extension-field signing procedure of SQIsign.

Overview.

The rest of the paper is organised as follows. Section 2 recalls the necessary background, including a high-level overview of SQIsign. Section 3 describes how the effect of field extensions in signing and the subsequent relaxation of the requirements on the prime. Section 4 analyses how the size of the 2^f-torsion correlates to verification time. Section 5 presents optimisations enabled by the

increased 2^f-torsion, while Section 6 gives further optimisations enabled by increased signature sizes. Finally, Section 7 gives some example parameters, and measures their performance compared to the state of the art.

Availability of software.

We make our Python and SageMath software publically available under the MIT licence at

https://github.com/TheSICQ/ApresSQI.

§2 Preliminaries

Chapter II gives an introduction to curves and their isogenies, and an introduction on SQIsign. Here, we recall the basics of the protocol and explain in more detail the steps in verification, SQIsign-friendly primes p, and computational details on isogenies with irrational kernel points.

Throughout, we will exploit the fact that every isomorphism class of supersingular curves has a model over $\mathbb{F}_{p^2}$, such that the p^2-power Frobenius π is equal to the multiplication-by-$(-p)$ map. Such curves E are $\mathbb{F}_{p^2}$-isogenous to curves defined over $\mathbb{F}_p$, and satisfy

$$E(\mathbb{F}_{p^{2k}}) = E\left[p^k - (-1)^k\right] \cong \mathbb{Z}/\left(p^k - (-1)^k\right)\mathbb{Z} \oplus \mathbb{Z}/\left(p^k - (-1)^k\right)\mathbb{Z}, \quad (11)$$

while their quadratic twist over $\mathbb{F}_{p^{2k}}$, which we will denote E_k^t, satisfies

$$E_k^t(\mathbb{F}_{p^{2k}}) = E\left[p^k + (-1)^k\right] \cong \mathbb{Z}/\left(p^k + (-1)^k\right)\mathbb{Z} \oplus \mathbb{Z}/\left(p^k + (-1)^k\right)\mathbb{Z}. \quad (12)$$

For such curves, for any positive integer $N \mid p^k \pm 1$, the full N-torsion group $E[N]$ is defined over $\mathbb{F}_{p^{2k}}$, either on the curve itself, or on its twist.

The SQIsign protocol.

A high-level description of the SQIsign Sigma protocol is given below (see also Figure 15).

SETUP: Fix a prime number p and supersingular elliptic curve $E_0/\mathbb{F}_{p^2}$ with known endomorphism ring.

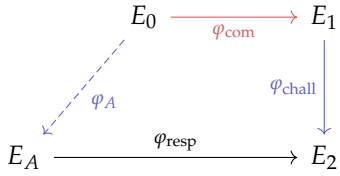


Figure 15: The SQIsign protocol with three phases: commitment φ_{com}, challenge φ_{chall}, and response φ_{resp}.

KEY GENERATION: Compute a secret key $\varphi_A : E_0 \to E_A$, giving the prover knowledge of $\mathrm{End}(E_A)$, with corresponding public verification key E_A.

COMMIT: The prover generates a random commitment isogeny $\varphi_{\text{com}} : E_0 \to E_1$, and sends E_1 to the verifier.

CHALLENGE: The verifier computes a random challenge isogeny $\varphi_{\text{chall}} : E_1 \to E_2$, and sends φ_{chall} to the prover.

RESPONSE: The prover uses the knowledge of φ_{com} and φ_{chall} to compute $\mathrm{End}(E_2)$, allowing the prover to compute the response isogeny $\varphi_{\text{resp}} : E_A \to E_2$, by translating an ideal computed using the generalised KLPT algorithm.

The verifier needs to check that φ_{resp} is an isogeny from E_A to E_2.[1] Assuming the hardness of the endomorphism ring problem, the protocol is sound: if the prover is able to respond to two different challenges φ_{chall}, φ'_{chall} with φ_{resp} and φ'_{resp}, the prover knows an endomorphism of the public key E_A, namely $\hat{\varphi}'_{\text{resp}} \circ \varphi'_{\text{chall}} \circ \hat{\varphi}_{\text{chall}} \circ \varphi_{\text{resp}}$. Proving zero-knowledge is harder and relies on the output distribution of the generalised KLPT algorithm. Note that KLPT is needed for computing the response:[2] while setting $\varphi_{\text{resp}} = \varphi_{\text{chall}} \circ \varphi_{\text{com}} \circ \hat{\varphi}_A$ gives an isogeny from E_A to E_2, this leaks the secret φ_A.[3] For a further discussion on zero-knowledge, we refer to the original SQIsign articles [148, 149].

1 Additionally, $\hat{\varphi}_{\text{chall}} \circ \varphi_{\text{resp}}$ needs to be cyclic. Observe that otherwise, the soundness proof might return a scalar endomorphism.

2 Alternatively, one can replace the connecting ideal with the shortest equivalent ideal, and translate it by embedding it in an isogeny between higher-dimensional abelian varieties, as shown in SQIsignHD [141].

3 This is also not a valid response, as the composition with $\hat{\varphi}_{\text{chall}}$ is not cyclic.

Remark 1. The best-known attacks against SQIsign are the generic attacks against the endomorphism ring problem. Their run time depends only on the size of p and, with high probability, do not recover the original secret isogeny, but rather a different isogeny between the same curves. Therefore, their complexity should be unaffected by the changes we introduce to the protocol in Section 3: for these attacks it does not matter whether the original secret isogeny is rational or irrational. In short, the changes in this work do not affect the security of SQIsign.

Verification.

Using the Fiat–Shamir transform [178], the SQIsign Sigma protocol is transformed into a signature scheme. This means that a signature on the message msg is of the form $\sigma = (\varphi_{\mathrm{resp}}, \mathsf{msg}, E_1)$. For efficiency, φ_{resp} is compressed, and E_1 is replaced by a description of φ_{chall}. Thus, given the signature σ and public key E_A, the verifier recomputes the response isogeny $\varphi_{\mathrm{resp}} : E_A \to E_2$ and the (dual of the) challenge isogeny $\hat{\varphi}_{\mathrm{chall}} : E_2 \to E_1$, and then verifies that the hash $\mathsf{H}(\mathsf{msg}, E_1)$ indeed generates φ_{chall}.

The isogeny φ_{resp} is of degree 2^e with $e = \lceil \frac{15}{4} \log(p) \rceil + 25$ (see [100, §7.2.3] for details), where 2^e corresponds to the output size of the generalised KLPT algorithm. The bottleneck in verification is the re-computation of φ_{resp} in $\lceil e/f \rceil$ blocks of size 2^f. Accelerating each block, and decreasing the number of blocks will be the focus of this paper.

2.1 SQISIGN-FRIENDLY PRIMES

Next, we give more details on the parameter requirements in SQIsign. We refer to the original SQIsign works [100, 148, 149] for a detailed description of their origins.

SQIsign prime requirements.

The main bottlenecks in SQIsign are related to the computation of isogenies, not the quaternion algebra computations. Recall from Equations (11) and (12) that, when working with supersingular elliptic curves $E/\mathbb{F}_{p^2}$, we have $E(\mathbb{F}_{p^2}) = E[p \pm 1]$. Thus, to use x-only arithmetic over $\mathbb{F}_{p^2}$, SQIsign restricts to computing isogenies of *smooth* degree $N \mid (p^2 - 1)$. Finding SQIsign-friendly primes thus reduces to finding primes p, with $p^2 - 1$ divisible by a large, smooth number. More explicitly, for a security level λ, the following parameters are needed:

- A prime p of bitsize $\log_2(p) \approx 2\lambda$ with $p \equiv 3 \mod 4$.
- The torsion group $E[2^f]$ as large as possible, that is $2^f \mid p+1$.
- A smooth odd factor $T \mid (p^2-1)$ of size roughly $p^{5/4}$.
- The degree of φ_{com}, denoted $D_{\text{com}} \mid T$, of size roughly $2^{2\lambda} \approx p$.
- The degree of φ_{chall}, denoted $D_{\text{chall}} \mid 2^f T$, of size roughly $2^\lambda \approx p^{1/2}$.
- Coprimality between D_{com} and D_{chall}.

To achieve NIST Level I, III, and V security, we set the security parameter as $\lambda = 128, 192, 256$, respectively. Concretely, this means that, for each of these security parameters, we have $\log p \approx 256, 384, 512$, and $\log T \approx 320, 480, 640$. The smoothness of T directly impacts the signing time, and the problem of finding primes p with a large enough T that is reasonably smooth is difficult. We refer to recent work on this problem for techniques to find suitable primes [75, 100, 124, 134, 148, 149].

The crucial observation for this work is that T occupies space in p^2-1 that limits the size of f, hence current SQIsign primes must sacrifice f to increase the smoothness of T.

Remark 2. SQIsign (NIST) further requires $3^g \mid p+1$ such that $D_{\text{chall}} = 2^f \cdot 3^g \geq p^{1/2}$ and $D_{\text{chall}} \mid p+1$. While this is not a strict requirement in the theoretical sense, it facilitates efficiency of computing φ_{chall}. From this point on, we ensure that this requirement is always fulfilled.

Remark 3. Since the curves E and their twists E^t satisfy

$$E(\mathbb{F}_{p^2}) \cong \mathbb{Z}/(p \pm 1)\mathbb{Z} \oplus \mathbb{Z}/(p \pm 1)\mathbb{Z},$$

and we work with both simultaneously, choosing T and f is often incorrectly described as choosing divisors of p^2-1. There is a subtle issue here: even if 2^f divides p^2-1, the kernel $E[2^f]$ may not exist as a subgroup of $\langle E(\mathbb{F}_{p^2}), \rho^{-1}(E^t(\mathbb{F}_{p^2}))\rangle \subseteq E(\mathbb{F}_{p^4})$, where $\rho : E \to E^t$ is the twisting isomorphism. While this does not usually matter in the case of SQIsign, where we pick 2^f as a divisor of $p+1$ and T is odd, this becomes a problem when working over multiple extension fields. In Section 3.2, we make this precise and reconcile it using Theorem 2.

2.2 RATIONAL ISOGENIES FROM IRRATIONAL GENERATORS

To facilitate signing with field extensions, we recall the techniques for computing $\mathbb{F}_{p^2}$-rational isogenies, that is, isogenies defined over $\mathbb{F}_{p^2}$, generated by *irrational* kernel points, that is, not defined over $\mathbb{F}_{p^2}$. In the context of this chapter, we again stress that such isogenies will only be computed by the signer; the verifier only works with points in $\mathbb{F}_{p^2}$.

The main computational task of most isogeny-based cryptosystems, including SQIsign, lies in evaluating isogenies given the generators of their kernels. Explicitly, given an elliptic curve $E/\mathbb{F}_q$, a point $K \in E(\mathbb{F}_{q^k})$ such that $\langle K \rangle$ is *defined over* $\mathbb{F}_q$,[4] and a list of points $(P_1, P_2, \ldots, P_n)$ in E, we wish to compute the list of points $(\varphi(P_1), \varphi(P_2), \ldots, \varphi(P_n))$, where φ is the separable isogeny with $\ker \varphi = \langle K \rangle$. Since we work with curves E whose p^2-Frobenius π is equal to the multiplication-by-$(-p)$ map, *every* subgroup of E is closed under the action of $\mathrm{Gal}(\bar{\mathbb{F}}_{p^2}/\mathbb{F}_{p^2})$, hence every isogeny from E can be made $\mathbb{F}_{p^2}$-rational, by composing with the appropriate isomorphism.

Computing isogenies of smooth degree.

Recall from Chapter II that an composite-degree isogeny factors as a composition of small prime degree isogenies, which we compute using Vélu-style algorithms. For simplicity, for the rest of the section, we therefore assume that $\langle K \rangle$ is a subgroup of order $\ell > 2$, where ℓ is a small prime.

At the heart of these Vélu-style isogeny formulas is evaluating the *kernel polynomial*. Pick any subset $S \subseteq \langle K \rangle$ such that $\langle K \rangle = S \sqcup -S \sqcup \{\mathcal{O}_E\}$. Then the kernel polynomial can be written as

$$f_S(X) = \prod_{P \in S} (X - x(P)). \tag{13}$$

Here, the generator K can be either a rational point, i.e., lying in $E(\mathbb{F}_q)$, or an irrational point, i.e., lying in $E(\mathbb{F}_{q^k})$ for $k > 1$, but whose group $\langle K \rangle$ is defined over $\mathbb{F}_q$. We briefly discuss how to solve the problem efficiently in the latter case.

4 That is, the group $\langle K \rangle$ is closed under the action of $\mathrm{Gal}(\bar{\mathbb{F}}_q/\mathbb{F}_q)$.

Irrational kernel generators.

For $K \notin E(\mathbb{F}_q)$ of order ℓ, we can speed up the computation of the kernel polynomial using the action of Frobenius. This is used in two recent works [22, 168], though the general idea is used even earlier [367].

As $\langle K \rangle$ is defined over $\mathbb{F}_q$, we know that the q-power Frobenius π acts as an endomorphism on $\langle K \rangle \subseteq E(\mathbb{F}_{p^k})$ and thus maps K to a multiple $[\gamma]K$ for some $\gamma \in \mathbb{Z}$. This fully determines the action on $\langle K \rangle$, i.e., $\pi|_{\langle K \rangle}$ acts as $P \mapsto [\gamma]P$ for all $P \in \langle K \rangle$. For the set S as chosen above, this action descends to an action on its x-coordinates $X_S = \{x(P) \in \mathbb{F}_{q^k} \mid P \in S\}$ and thus partitions X_S into orbits $\{x(P), x([\gamma]P), x([\gamma^2]P), \ldots\}$ of size equal to the order of γ in $(\mathbb{Z}/\ell\mathbb{Z})^\times/\{1, -1\}$.

If we pick one representative $P \in S$ per orbit, and call this set of points S_0, we can compute the kernel polynomial (13) as a product of the minimal polynomials $\mu_{x(P)}$ of the $x(P) \in \mathbb{F}_{q^k}$ for these $P \in S_0$, with each $\mu_{x(P)}$ defined over $\mathbb{F}_q$, as

$$f_S(X) = \prod_{P \in S_0} \mu_{x(P)}(X), \tag{14}$$

where μ_β denotes the minimal polynomial of β over $\mathbb{F}_q$.

To evaluate f_S on $\alpha \in \mathbb{F}_q$, we only require the smaller polynomial $f_{S_0}(X)$ to compute $\mathrm{Norm}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(f_{S_0}(\alpha))$ by [22]

$$\prod_{\pi \in G} \pi(f_{S_0}(\alpha)) = \prod_{P \in S_0} \prod_{\pi \in G} (\alpha - \pi(x(P))) = \prod_{P \in S_0} \mu_{x(P)}(\alpha),$$

where $G = \mathrm{Gal}(\mathbb{F}_{q^k}/\mathbb{F}_q)$. This allows us to compute the image under f_S of x-values of points in $E(\mathbb{F}_q)$, but only when $\alpha \in \mathbb{F}_q$. To evaluate $f_S(\alpha)$ for general $\alpha \in \overline{\mathbb{F}}_p$, i.e. to compute the image of a point in $E(\overline{\mathbb{F}}_p)$, we instead use the larger polynomial $f_S(X)$, which we compute, as in Equation (14), as a product of the minimal polynomials $\mu_{x(P)}$, where we use Shoup's algorithm [352] to compute each $\mu_{x(P)}$ given $x(P)$. Computing $f_S(X)$ requires a total of $O(\ell k) + \widetilde{O}(\ell)$ operations, with k such that each $x(P) \in \mathbb{F}_{q^k}$. Evaluation f_S at α takes $\widetilde{O}(\ell k')$ operations, with k' the smallest value such that $\alpha \in \mathbb{F}_{q^{k'}}$ [168, Section 4.3].

Remark 4. The biggest drawback to using this technique is that $\sqrt{\text{élu}}$ is no longer effective, as we would need to work in the smallest field where both the isogeny generator and the x-value of the point we are evaluating are defined in.

§3 Signing with Extension Fields

By allowing torsion T from extension fields, we enable more flexibility in choosing SQIsign primes p, thus enabling a larger 2^f-torsion. Such torsion T requires us to compute rational isogenies with kernel points in extension fields $\mathbb{F}_{p^{2k}}$. This section describes how to adapt SQIsign's signing procedure to enable such isogenies, and the increased cost this incurs. In particular, we describe two approaches for T: allowing torsion T from a particular extension field $\mathbb{F}_{p^{2k}}$, or from all extension fields $\mathbb{F}_{p^{2n}}$ for $1 \leq n \leq k$. The first approach means that we can look for T dividing an integer of bit size $\Theta(k \log p)$, and the second approach allows for $\Theta(k^2 \log p)$. Both allow for a significant increase in 2^f-torsion, and in Section 4, we then explore how this increased 2^f-torsion affects verification.

3.1 Changes in the signing procedure

Recall from Chapter II that the signing operation in SQIsign requires us to work with both elliptic curves and quaternion algebras, and to translate back and forth between these worlds. Note that the subroutines that work solely with objects in the quaternion algebra $\mathcal{B}_{p,\infty}$, including all operations in KLPT and its derivatives, are indifferent to what extension fields the relevant torsion groups lie in. Hence, a large part of signing is unaffected by torsion from extension fields. In fact, the only subroutines that are affected by moving to extension fields are those we rely on to translate ideals back to isogenies.

We describe the such an ideal-to-isogeny algorithm in Algorithm 17, which we denote $\mathsf{IdealToIsogeny}_D$. This algorithm translates $\mathcal{O}_0$-ideals I of norm dividing D to their corresponding isogenies φ_I. $\mathsf{IdealToIsogeny}_D$ is not used during verification, and is used only in the following parts of signing:

COMMITMENT: The signer translates a random ideal of norm D_{com} to its corresponding isogeny, using one execution of $\mathsf{IdealToIsogeny}_{D_{\text{com}}}$.

RESPONSE: The signer translates an ideal of norm 2^e to its corresponding isogeny, requiring $2 \cdot \lceil e/f \rceil$ executions of $\mathsf{IdealToIsogeny}_T$.[5]

Remark 5. We choose parameters such that $2^f \mid p+1$ and $D_{\text{chall}} \mid p+1$, so that $E[2^f]$ and $E[D_{\text{chall}}]$ are defined over $\mathbb{F}_{p^2}$. As a result, the verifier only works in $\mathbb{F}_{p^2}$ and the added complexity of extension fields applies only to the signer.

5 The technical details are given in [149].

Algorithm 17 $\mathsf{IdealToIsogeny}_D(I)$

Input: I a left $\mathcal{O}_0$-ideal of norm dividing D
Output: φ_I

1: Compute α such that $I = \mathcal{O}_0\langle \alpha, \mathrm{nrd}(I)\rangle$
2: Let $\mathbf{A} = [1, i, \frac{i+j}{2}, \frac{1+k}{2}]$ denote a basis of $\mathcal{O}_0$
3: Compute $\mathbf{v}_{\bar{\alpha}} := [x_1, x_2, x_3, x_4]^T \in \mathbb{Z}^4$ such that $\mathbf{A}\mathbf{v}_{\bar{\alpha}} = \bar{\alpha}$
4: **for** $\ell^e \mid\mid D$ **do**
5: $\quad \bar{\alpha}|_{\langle P_{\ell^e}, Q_{\ell^e}\rangle} \leftarrow x_1\mathbf{I} + x_2(i|_{\langle P_{\ell^e}, Q_{\ell^e}\rangle}) + x_3(\frac{i+j}{2}|_{\langle P_{\ell^e}, Q_{\ell^e}\rangle}) + x_4(\frac{1+k}{2}|_{\langle P_{\ell^e}, Q_{\ell^e}\rangle})$
6: $\quad$ Let a, b, c, d be integers such that $\bar{\alpha}|_{\langle P_{\ell^e}, Q_{\ell^e}\rangle} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$
7: $\quad K_{\ell^e} \leftarrow [a]P_{\ell^e} + [c]Q_{\ell^e}$
8: $\quad$ **if** $\mathrm{ord}(K_{\ell^e}) < \ell^e$ **then**
9: $\quad\quad K_{\ell^e} \leftarrow [b]P_{\ell^e} + [d]Q_{\ell^e}$
10: Set φ_I to be the isogeny generated by the points K_{ℓ^e}.
11: **return** φ_I

To facilitate signing with field extensions, we slightly adapt $\mathsf{IdealToIsogeny}_D$ so that it works with prime powers separately. The additional cost of this is negligible compared to the cost of computing the isogeny from the generators because finding the action of the relevant endomorphisms consists of simple linear algebra. In Line 5 of Algorithm 17, the notation $\beta|_{\langle P_{\ell^e}, Q_{\ell^e}\rangle}$ refers to the action of an endomorphism β on a fixed basis P_{ℓ^e}, Q_{ℓ^e} of $E[\ell^e]$. This action is described by a matrix in $M_2(\mathbb{Z}/\ell^e\mathbb{Z})$. These matrices can be precomputed, hence the only operations in which the field of definition of $E[\ell^e]$ matters are the point additions in Lines 7 and 9, and isogenies generated by each K_{ℓ^e} in Line 10.

3.2 INCREASED TORSION AVAILABILITY FROM EXTENSION FIELDS

This section describes the two approaches to allow torsion groups from extension fields, either from a single extension field $\mathbb{F}_{p^k}$ or all $\mathbb{F}_{p^k}$ for $1 \leq k \leq n$. Both permit more flexibility in choosing the final prime p and thus allow us to increase the 2^f-torsion.

Working with a single field extension of $\mathbb{F}_{p^2}$.

Although the choice of solely working in $\mathbb{F}_{p^2}$ occurs naturally,[6] there is no reason *a priori* that this choice is optimal. Instead, we can choose to work in the field $\mathbb{F}_{p^{2k}}$. We emphasise that this does *not* affect signature sizes; the only drawback is that we now perform more expensive $\mathbb{F}_{p^{2k}}$-operations during signing in IdealToIsogeny. The upside, however, is a relaxed prime requirement: we are no longer bound to $E[T] \subseteq \langle E(\mathbb{F}_{p^2}), \rho^{-1}(E^t(\mathbb{F}_{p^2}))\rangle$ and can instead use

$$E[T] \subseteq \langle E(\mathbb{F}_{p^{2k}}), \rho^{-1}(E^t(\mathbb{F}_{p^{2k}}))\rangle.$$

By Equations (11) and (12), we have $E(\mathbb{F}_{p^{2k}}) \cong E[p^k \pm 1]$ and $E^t(\mathbb{F}_{p^{2k}}) \cong E[p^k \mp 1]$, thus we simply get

$$E[T] \subseteq E\left[\frac{p^{2k}-1}{2}\right],$$

since $\langle E[A], E[B]\rangle = E[\mathrm{lcm}(A, B)]$. Hence, by using torsion defined over $\mathbb{F}_{p^{2k}}$, we increase $T \mid (p^2-1)/2$ to $T \mid (p^{2k}-1)/2$. This implies there are $2k \log p$ bits available to find T with adequate smoothness, instead of $2 \log p$ bits.

Working with multiple field extensions of $\mathbb{F}_{p^2}$.

Instead of fixing a single higher extension field $\mathbb{F}_{p^{2k}}$, we can also choose to work with multiple field extensions, in particular all fields $\mathbb{F}_{p^{2k}}$, where $1 \leq k \leq n$. The torsion group we can access by this relaxed requirement is described by the following definition.

Definition 1. Let E be a supersingular elliptic curve over $\mathbb{F}_{p^2}$ and let E_k^t denote an arbitrary quadratic twist of E over $\mathbb{F}_{p^{2k}}$ with associated twisting isomorphism $\rho_k : E \to E_k^t$. We define the *n-available torsion* of E to be the group generated by $E(\mathbb{F}_{p^{2k}})$ and $\rho_k^{-1}(E_k^t(\mathbb{F}_{p^2}))$ for $1 \leq k \leq n$.

Any point P in the n-available torsion can thus be written as a sum

$$P = \sum_{i=1}^{n} (P_i + \rho_k^{-1}(P_i^t))$$

6 It is the smallest field over which every isomorphism class of supersingular elliptic curves has a model.

of points $P_i \in E(\mathbb{F}_{p^{2k}})$ and $P_i^t \in E_k^t(\mathbb{F}_{p^{2k}})$. Since ρ_k keeps the x-coodinate fixed, the computation of this isomorphism can be ignored when using x-only arithmetic, and we simply obtain a sum of points whose x-coordinates lie in $\mathbb{F}_{p^{2k}}$ for $1 \leq k \leq n$. This justifies the name n-available torsion, as we do not have to go beyond $\mathbb{F}_{p^n}$ to do arithmetic with P by working with the summands separately.

The structure of the n-available torsion is completely captured by the following result.

Theorem 2. Let $p > 2$ be a prime, and let $E/\mathbb{F}_{p^2}$ be a supersingular curve with $\mathrm{tr}(\pi) = \pm 2p$, where π is the Frobenius endomorphism. Then the n-available torsion is precisely the group $E[N]$ with

$$N = \prod_{k=1}^{n} \Phi_k(p^2)/2,$$

where Φ_k denotes the k-th cyclotomic polynomial.

Before we prove Theorem 2, we require the following technical lemma.

Lemma 3. For any integer $m \geq 2$, we have the following identity

$$\mathrm{lcm}\left(\{m^k - 1\}_{k=1}^{n}\right) = \prod_{k=1}^{n} \Phi_k(m),$$

where Φ_k denotes the k-th cyclotomic polynomial.

Proof. We show that any prime power dividing either side of the equation, also divides the other side.

For any prime ℓ and $e > 0$, if ℓ^e divides the left-hand side, then, by definition, it divides $m^i - 1 = \prod_{d|i} \Phi_d(m)$ for some $1 \leq i \leq n$. Hence, it also divides the right-hand side. Conversely, if ℓ^e divides the right-hand side, then ℓ^e also divides the left-hand side. To show this, we need to know when $\Phi_i(m)$ and $\Phi_j(m)$ are coprime. We note that

$$\gcd(\Phi_i(m), \Phi_j(m)) \mid R$$

where R is the resultant of $\Phi_i(X)$ and $\Phi_j(X)$, and a classic result by Apostol [10, Theorem 4.], tells us that

$$\mathrm{Res}(\Phi_i(X), \Phi_j(X)) > 1 \Rightarrow i = jd$$

for $i > j$ and some integer d. Using this, if ℓ^e divides the right-hand side, then it will also divide the product

$$\prod_{k=1}^{\lfloor n/d \rfloor} \Phi_{dk}(m),$$

for some integer d, and this product divides the left-hand side, as it divides $m^{d\lfloor n/d \rfloor} - 1$. □

We can now conclude the proof of Theorem 2.

Proof. From the structure of $E(\mathbb{F}_{p^{2k}})$, as in Remark 3, where E is as in the statement, the n-available torsion can be seen as the group generated by the full torsion groups $E[p^k \pm 1]$ for $1 \leq k \leq n$. Using the fact that $\langle E[A], E[B] \rangle = E[\mathrm{lcm}(A, B)]$, we see that the n-available torsion is $E[N]$ where

$$N = \mathrm{lcm}\left(\{p^k - 1\}_{k=1}^n \cup \{p^k + 1\}_{k=1}^n\right) = \mathrm{lcm}\left(\{p^{2k} - 1\}_{k=1}^n\right)/2,$$

where the last equality only holds for $p > 2$. Applying Lemma 3 with $m = p^2$, we obtain

$$N = \prod_{k=1}^{n} \Phi_k(p^2)/2.$$

□

We find that using all extension fields $\mathbb{F}_{p^{2k}}$, for $1 \leq k \leq n$, increases $T \mid p^2 - 1$ to $T \mid N$, with N as given by Theorem 2. Given that

$$\log(N) = \sum_{k=1}^{n} \log(\Phi_k(p^2)/2) \approx 2 \sum_{k=1}^{n} \phi(k) \log(p),$$

where ϕ denotes Euler's totient function, and the fact that $\sum_{k=1}^{n} \phi(k)$ is in the order of $\Theta(n^2)$, we find that $T \mid N$ gives roughly $n^2 \log(p)$ more bits to find T with adequate smoothness, compared to the $2\log(p)$ bits in the classical case of working over $\mathbb{F}_{p^2}$, and $2k\log(p)$ bits in the case of working over a single field extension $\mathbb{F}_{p^{2k}}$. Due to this, we only consider working in multiple field extensions from this point on.

3.3 COST OF SIGNING USING EXTENSION FIELDS

In SQIsign, operations over $\mathbb{F}_{p^2}$ make up the majority of the cost during signing [149, Section 5.1]. Hence, we get a rough cost of signing by ignoring purely quaternionic operations, in which case the bottleneck of the signing procedure becomes running $\mathsf{IdealToIsogeny}_T$ as many times as required by the IdealToIsogenyEichler algorithm [149, Algorithm 5] in the response phase. In other words, we estimate the total signing cost from the following parameters:

- f, such that $2^f \mid p+1$,
- T, the chosen torsion to work with, and
- for each $\ell_i^{e_i} \mid T$, the smallest k_i such that $E[\ell_i^{e_i}]$ is defined over $\mathbb{F}_{p^{2k_i}}$.

Since Algorithm 17 works with prime powers separately, we can estimate the cost of a single execution by considering the cost per prime power.

COST PER PRIME POWER. For each $\ell_i^{e_i} \mid T$, let k_i denote the smallest integer so that $E[\ell_i^{e_i}] \subseteq E(\mathbb{F}_{p^{2k_i}})$, and let $M(k_i)$ denote the cost of operations in $\mathbb{F}_{p^{2k_i}}$ in terms of $\mathbb{F}_{p^2}$-operations. Computing the generator $K_i := K_{\ell_i^{e_i}}$ consists of a few point additions in $E[\ell_i^{e_i}]$, hence is $O(M(k) \cdot e_i \log \ell_i)$, while the cost of computing the isogeny generated by K_i comes from computing e_i isogenies of degree ℓ_i at a cost of $O(\ell_i k) + \tilde{O}(\ell_i)$, using the techniques from Section 2.2.

To compute the whole isogeny, we need to push the remaining generators K_j, through this isogeny generated by K_i for $j \neq i$. To minimize the total cost, we pick the greedy strategy of always computing the smaller ℓ_i first. This bounds the cost of evaluating K_j in *other* isogenies by $O(M(k) \cdot \ell_j)$.

TOTAL COST OF SIGNING. Based on the analysis above, we let

$$\mathrm{Cost}_p(\ell^e) = c_1 M(k_i) e \log \ell + c_2 e \ell k_i + c_3 e \ell \log(\ell) + c_4 M(k_i) \ell$$

where k_i, and $M(k)$ are as before, and c_i are constants corresponding to the differences in the asymptotic complexities. Since we can estimate the total cost of executing $\mathsf{IdealToIsogeny}_T$ by summing the cost of each maximal prime power divisor of T, and observing that signing consists of executing $\mathsf{IdealToIsogeny}_{D_{\mathrm{com}}}$

one time, and $\mathsf{IdealToIsogeny}_T$ a total of $2 \cdot \lceil e/f \rceil$ times, we get a rough cost estimate of signing as

$$\textsc{SigningCost}_p(T) = (2 \cdot \lceil e/f \rceil + 1) \cdot \sum_{\ell_i^{e_i} | T} \textsc{Cost}_p(\ell_i^{e_i}).$$

In Section 7, we use this function to pick p and T minimising this cost. While this cost metric is very rough, we show that our implementation roughly matches the times predicted by this function. Further, we show that this cost metric suggests that going to extension fields gives signing times within the same order of magnitude as staying over $\mathbb{F}_{p^2}$, even when considering the additional benefit of using $\sqrt{\text{élu}}$ to compute isogenies in the latter case.

§4 Effect of Increased Torsion on Verification

In Section 3, we showed that signing with extension fields gives us more flexibility in choosing the prime p, and, in particular, allows us to find primes with rational 2^f-torsion for larger f. In this section, we analyse how such an increase in 2^f-torsion affects the cost of SQIsign verification, e.g., computing φ_{resp} and $\hat{\varphi}_{\text{chall}}$, in terms of $\mathbb{F}_p$-multiplications,[7] taking the SQIsign (NIST) implementation (with no further optimisations) as the baseline for comparison.

We start with a detailed description of all steps in verification, on which most of our later analysis is based.

4.1 Detailed description of verification

Recall that a SQIsign signature σ for a message msg created by a signer with secret signing key $\varphi_A : E_0 \to E_A$ proves knowledge of an endomorphism on E_A by describing an isogeny $\varphi_{\text{resp}} : E_A \to E_2$ of degree 2^e (see Figure 15). A given message msg is hashed on E_1 to a point K_{chall} of order D_{chall}, hence represents an isogeny $\varphi_{\text{chall}} : E_1 \to E_2$. A signature is valid if the composition of φ_{resp} with $\hat{\varphi}_{\text{chall}}$ is cyclic of degree $2^e \cdot D_{\text{chall}}$.

Thus, to verify a signature σ, the verifier must **(a)** recompute φ_{resp}, **(b)** compute the dual of φ_{chall}, to confirm that both are well-formed, and finally **(c)** recompute the hash of the message msg to confirm the validity of the signature.

In SQIsign, the size of the sample space for φ_{chall} impacts soundness, a key security property for signature schemes. In SQIsign (NIST), to obtain negligible

7 Recall, we denote multiplications by **M**, squarings by **S**, and additions by **a**.

soundness error (in the security parameter λ) the message is hashed to a cyclic isogeny φ_{chall} of degree $D_{\text{chall}} = 2^f \cdot 3^g > 2^\lambda$. It follows that, when $f \geq \lambda$, we can simply set $D_{\text{chall}} = 2^\lambda$.

The signature σ consists of a compressed description of the isogenies φ_{resp} and $\hat{\varphi}_{\text{chall}}$. For $f < \lambda$ and $D_{\text{chall}} = 2^f \cdot 3^g$ it is of the form

$$\sigma = (b, s^{(1)}, \ldots, s^{(n)}, r, b_2, s_2, b_3, s_3)$$

with $s^{(j)}, s_2 \in \mathbb{Z}/2^f\mathbb{Z}$, $s_3 \in \mathbb{Z}/3^g\mathbb{Z}$, $r \in \mathbb{Z}/2^f 3^g\mathbb{Z}$, and $b, b_2, b_3 \in \{0,1\}$. If $f \geq \lambda$, we set $D_{\text{chall}} = 2^f$ and have $s_2 \in \mathbb{Z}/2^\lambda\mathbb{Z}$ and $r \in \mathbb{Z}/2^f\mathbb{Z}$, while b_3, s_3 are omitted. Algorithmically, the verification steps **(a)-(c)** mostly requires three subroutines.

FINDBASIS: Given a curve E, find a deterministic basis (P, Q) of $E[2^f]$.

FINDKERNEL: Given a curve E with basis (P, Q) for $E[2^f]$ and $s \in \mathbb{Z}/2^f\mathbb{Z}$, compute the kernel generator $K = P + [s]Q$.

COMPUTEISOGENY: Given a curve E and a kernel generator K, compute the isogeny $\varphi : E \to E/\langle K \rangle$ and $\varphi(Q)$ for some $Q \in E$.

Below we detail each of the three verification steps **(a)-(c)**.

STEP (A). Computing φ_{resp} is split up into $n-1$ *blocks* $\varphi^{(j)} : E^{(j)} \to E^{(j+1)}$ of size 2^f, and a last block of size 2^{f_0}, where $f_0 = e - (n-1) \cdot f$. For every $\varphi^{(j)}$, the kernel $\langle K^{(j)} \rangle$ is given by the generator $K^{(j)} = P^{(j)} + [s^{(j)}]Q^{(j)}$ for a deterministic basis $(P^{(j)}, Q^{(j)})$ of $E^{(j)}[2^f]$.

In the first block, after sampling $(P^{(1)}, Q^{(1)})$ via FindBasis, the bit b indicates whether $P^{(1)}$ and $Q^{(1)}$ have to be swapped before running FindKernel. For the following blocks, the verifier pushes $Q^{(j)}$ through the isogeny $\varphi^{(j)}$ to get a point $Q^{(j+1)} \leftarrow \varphi^{(j)}(Q^{(j)})$ on $E^{(j+1)}$ of order 2^f above $(0,0)$.[8] Hence, for $j > 1$ FindBasis only needs to find a suitable point $P^{(j)}$ to complete the basis $(P^{(j)}, Q^{(j)})$. Furthermore, $K^{(j)}$ is never above $(0,0)$ for $j > 1$, which ensures cyclicity when composing $\varphi^{(j)}$ with $\varphi^{(j-1)}$. In all cases we use $s^{(j)}$ from σ to compute the kernel generator $K^{(j)}$ via FindKernel and $\varphi^{(j)}$ via ComputeIsogeny.

The last block of degree 2^{f_0} uses $Q^{(n)} \leftarrow [2^{f-f_0}]\varphi^{(n-1)}(Q^{(n-1)})$ and samples another point $P^{(n)}$ as basis of $E^{(n)}[2^{f_0}]$. In the following, we will often assume

8 A point P is said to be *above* a point R if $[k]P = R$ for some $k \in \mathbb{N}$ dividing the order of P.

$f_0 = f$ for the sake of simplicity.[9] An algorithmic description of a single block of SQIsign (NIST) is given in Algorithm 18 in Section B.

STEP (B). Computing $\hat{\varphi}_{\text{chall}}$ requires a single isogeny of smooth degree $D_{\text{chall}} \approx 2^\lambda$. For the primes given in SQIsign (NIST), we have $E_2[D_{\text{chall}}] \subseteq E_2(\mathbb{F}_{p^2})$. Thus, we compute $\hat{\varphi}_{\text{chall}}$ by deterministically computing a basis (P, Q) for $E_2[D_{\text{chall}}]$ and finding the kernel $\langle K \rangle$ for $\hat{\varphi}_{\text{chall}} : E_2 \to E_1$. For $f < \lambda$, we have $D_{\text{chall}} = 2^f \cdot 3^g$, and split this process into two parts.

Given the basis (P, Q) for $E_2[D_{\text{chall}}]$, we compute $(P_2, Q_2) = ([3^g]P, [3^g]Q)$ as basis of $E_2[2^f]$, and use $K_2 = P_2 + [s_2]Q_2$, where b_2 indicates whether P_2 and Q_2 have to be swapped prior to computing K_2. We compute $\varphi_2 : E_2 \to E_2'$ with kernel $\langle K_2 \rangle$, and $P_3 = [2^f]\varphi_2(P)$ and $Q_3 = [2^f]\varphi_3(Q)$ then form a basis of $E_2'[3^g]$. Again, b_3 indicates a potential swap of P_3 and Q_3, while $K_3 = P_3 + [s_3]Q_3$ is the kernel generator of the isogeny $\varphi_3 : E_2' \to E_1$. Thus, we have $\hat{\varphi}_{\text{chall}} = \varphi_3 \circ \varphi_2$. If $f \geq \lambda$, we require only the first step.

We furthermore verify that the composition of φ_{resp} and $\hat{\varphi}_{\text{chall}}$ is cyclic, by checking that the first 2-isogeny step of φ_2 does not revert the last 2-isogeny step of $\varphi^{(n)}$. This guarantees that $\hat{\varphi}_{\text{chall}} \circ \varphi_{\text{resp}}$ is non-backtracking, hence cyclic.

STEP (C). On E_1, the verifier uses the point $Q' \leftarrow \hat{\varphi}_{\text{chall}}(Q')$, where Q' is some (deterministically generated) point, linearly independent from the generator of $\hat{\varphi}_{\text{chall}}$, and r (given in σ) to compute $[r]Q'$, and checks if $[r]Q'$ matches the hashed point $K_{\text{chall}} = \mathsf{H}(\mathsf{msg}, E_1)$ with hash function H.

4.2 IMPACT OF LARGE f ON VERIFICATION

The techniques of Section 3 extend the possible range of f to any size below $\log(p)$. This gives two benefits to the cost of verification, especially when $f \geq \lambda$.

Number of blocks in φ_{resp}.

The larger f is, the fewer blocks of size 2^f are performed in **Step (a)**. Per block, the dominating part of the cost are FindBasis and FindKernel as we first need to complete the torsion basis $(P^{(j)}, Q^{(j)})$ for $E^{(j)}[2^f]$ (given $Q^{(j)}$ if $j > 1$), followed by computing $K^{(j)} = P^{(j)} + [s^{(j)}]Q^{(j)}$. By minimizing the number of blocks n, we minimize the amount of times we perform FindBasis and FindKernel, and the cost of each individual FindKernel only mildly increases, as $s^{(j)}$ increases in

9 In contrast to earlier versions, SQIsign (NIST) fixes $f_0 = f$. However, our analysis benefits from allowing $f_0 < f$.

size. The overall cost of ComputeIsogeny, that is, performing the n isogenies of degree 2^f given their kernels $K^{(j)}$, only moderately increases with growing f.

We further note that larger f requires fewer T-isogeny computations for the signer, hence signing performance also benefits from smaller n.

Challenge isogeny.

When $f \geq \lambda$, we can simply set $D_{\text{chall}} = 2^\lambda$, which has two main benefits.

- The cost of FindBasis for this step is significantly reduced as finding a basis for $E[2^\lambda]$ is much easier than a basis search for $E[2^f \cdot 3^g]$.
- The cost for ComputeIsogeny for φ_{chall} decreases as we only have to compute a chain of 2-isogenies instead of additional 3-isogenies.

4.3 IMPLEMENTATION AND BENCHMARK OF COST

To measure the influence of the size of f on the performance, we implemented SQIsign verification for the NIST Level I security parameter set in Python, closely following SQIsign (NIST). As is standard in isogeny-based schemes, we use x-only arithmetic and represent points and curve coefficients projectively. The benchmark counts $\mathbb{F}_p$-operations and uses a cost metric that allows us to estimate the runtime of real-world implementations for 256-bit primes $p^{(f)}$, where $p^{(f)}$ denotes a prime such that 2^f divides $p^{(f)} + 1$. We benchmark primes $p^{(f)}$ for all values $50 \leq f \leq 250$. These results serve as a baseline to which we compare the optimisations that we introduce in Sections 5 and 6. We briefly outline how SQIsign (NIST) implements the three main subroutines FindBasis, FindKernel, and ComputeIsogeny.

FindBasis. We search for points of order 2^f by sampling x-coordinates in a specified order,[10] and check if the corresponding point P lies on E (and not on its twist E^t). We then compute $P \leftarrow [\frac{p+1}{2^f}]P$ and verify that $[2^{f-1}]P \neq \infty$. Given two points $P, Q \in E$ of order 2^f, we verify linear independence by checking that $[2^{f-1}]P \neq [2^{f-1}]Q$, and discard and re-sample the second point otherwise.

FindKernel. Given a basis (P, Q), FindKernel computes $K = P + [s]Q$ via the `3ptLadder` algorithm as used in SIKE [221]. In addition to the x-coordinates x_P

10 SQIsign (NIST) fixes the sequence $x_k = 1 + k \cdot i$ with $i \in \mathbb{F}_{p^2}$ such that $i^2 = -1$ and picks the smallest k for which we find a suitable point.

and x_Q of P and Q, it requires the x-coordinate x_{P-Q} of $P - Q$. Hence, after running FindBasis, we further compute x_{P-Q} as described in SQIsign (NIST).

ComputeIsogeny. Given a kernel generator K of order 2^f, ComputeIsogeny follows the approach of SIKE [221], and computes the 2^f-isogeny $\varphi^{(j)}$ as a chain of 4-isogenies for efficiency reasons. If f is odd, we further compute a single 2-isogeny. Following SQIsign (NIST), ComputeIsogeny proceeds as follows:

1. Compute $R = [2^{f-2}]K$ and the corresponding 4-isogeny φ with kernel $\langle R \rangle$. Note that the point $(0, 0)$ might be contained in $\langle R \rangle$ for the first block in φ_{resp}, which requires a special 4-isogeny formula. Thus, we check if this is the case and call the suitable 4-isogeny function. We set $K \leftarrow \varphi(K)$.
2. If f is odd, we compute $R = [2^{f-3}]K$, the 2-isogeny φ with kernel $\langle R \rangle$, and $K \leftarrow \varphi(K)$.
3. Compute the remaining isogeny of degree $2^{f'}$ with even f' as a chain of 4-isogenies, where $(0, 0)$ is guaranteed not to lie in any of the kernels.

In the last step, SQIsign (NIST) uses *optimal strategies* as in SIKE [221] to compute a chain of 4-isogenies. Naive multiplicative strategies would compute $R = [2^{f'-2j}]K$, the 4-isogeny φ with kernel $\langle R \rangle$, and $K \leftarrow \varphi(K)$ for $j = 1, \ldots, f'/2$. However, this strategy is dominated by costly doublings. Instead, we can save intermediate multiples of K during the computation of $R = [2^{f'-2j}]K$, and push them through isogenies to save multiplicative effort in following iterations. Optimal strategies that determine which multiples are pushed through isogenies and minimise the cost can be found efficiently [177, 221].

We note that for $f < \lambda$ the computation of $\hat{\varphi}_{\text{chall}}$ requires small adaptations to these algorithms to allow for finding a basis of $E[D_{\text{chall}}]$ and computing 3-isogenies. Most notably, SQIsign (NIST) does *not* use optimised formulas or optimal strategies for 3-isogenies from SIKE [221], but uses a multiplicative strategy and general odd-degree isogeny formulas [128, 276]. We slightly deviate from SQIsign (NIST) by implementing optimised 3-isogeny formulas, but note that the performance difference is minor and in favor of SQIsign (NIST).

Cost metric.

In implementations, $\mathbb{F}_{p^2}$-operations usually call underlying $\mathbb{F}_p$-operations. We follow this approach and use the total number of $\mathbb{F}_p$-operations in our benchmarks. As cost metric, we express these operations in terms of $\mathbb{F}_p$-multiplications, with $\mathbf{S} = 0.8 \cdot \mathbf{M}$, ignoring $\mathbb{F}_p$-additions and subtractions due

to their small impact on performance. $\mathbb{F}_p$-inversions, $\mathbb{F}_p$-square roots, and Legendre symbols over $\mathbb{F}_p$ require exponentiations by an exponent in the range of p, hence we count their cost as $\log p$ $\mathbb{F}_p$-multiplications. In contrast to measuring clock cycles of an optimised implementation, our cost metric eliminates the dependence on the level of optimisation of finite field arithmetic and the specific device running SQIsign, hence, can be considered more general.

Benchmark results.

Figure 16 shows the verification cost for the NIST Level I-sized primes $p^{(f)}$ for $50 \leq f \leq 250$, fixing $e = 975$, using our cost metric. For more efficient benchmarking, we sample random public key curves and signatures σ of the correct form instead of signatures generated by the SQIsign signing procedure.

The graph shows the improvement for $f \geq 128$. Furthermore, we can detect when the number of blocks n decreases solely from the graph (e.g. $f = 122, 140, 163, 195, 244$). The cost of sampling a 2^f-torsion basis is highly variable between different runs for the same prime, which is visible from the oscillations of the graph. The performance for odd f is worse in general due to the inefficient integration of the 2-isogeny, which explains the zigzag-shaped graph.

From the above observations, we conclude that $f \geq \lambda$ is significantly faster for verification, with local optima found at $f = 195$ and $f = 244$, due to those being (almost) exact divisors of the signing length $e = 975$.

Remark 6. The average cost of FindBasis differs significantly between primes p even if they share the same 2^f-torsion. This happens because SQIsign(NIST) finds basis points from a pre-determined sequence $[x_1, x_2, x_3, \ldots]$ with $x_j \in \mathbb{F}_{p^2}$. As we will see in Section 5, these x_j values can not be considered random: some values x_j are certain to be above a point of order 2^f, while others are certain not to be, for any supersingular curve over p.

§5 Optimisations for Verification

In this section, we show how the improvements from Section 3 that increase f beyond λ together with the analysis in Section 4 allow several other optimisations that improve the verification time of SQIsign in practice. Whereas the techniques in Section 3 allow us to decrease the *number* of blocks, in this section, we focus on the operations occurring *within blocks*. We optimise the cost of FindBasis, FindKernel and ComputeIsogeny.

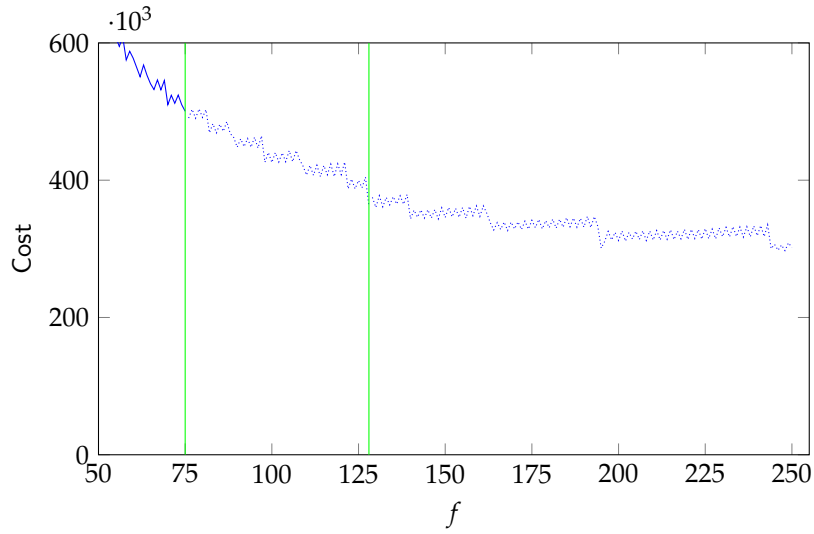


Figure 16: Cost in $\mathbb{F}_p$-multiplications for verification at NIST Level I security, for varying f and $p^{(f)}$, averaged over 1024 runs per prime. The green vertical lines mark $f = 75$ as used in SQIsign (NIST) for signing without extension fields, and $f = \lambda = 128$, beyond which we can set $D_{\text{chall}} = 2^\lambda$. The dotted graph beyond $f = 75$ is only accessible when signing with extension fields.

We first analyse the properties of points that have full 2^f-torsion, and use these properties to improve FindBasis and FindKernel for general f. We then describe several techniques specifically for $f \geq \lambda$. Altogether, these optimisations significantly change the implementation of verification in comparison to SQIsign (NIST). We remark that the implementation of the signing procedure must be altered accordingly, as exhibited by our implementation.

Notation.

As we mostly focus on the subroutines *within* a specific block $E^{(j)} \to E^{(j+1)}$, we will omit the superscripts in $E^{(j)}$, $K^{(j)}$, $P^{(j)}$, ... and write E, K, P, ... to simplify notation.

For reference throughout this section, the pseudocode for a single block in the verification procedure of SQIsign (NIST) and of our optimised variant is in Section B as Algorithm 18 and Algorithm 19, respectively.

5.1 BASIS GENERATION FOR FULL 2-POWER TORSION

We first give a general result on points having full 2^f-torsion that we will use throughout this section. This theorem generalises previous results [130, 258] and will set the scene for easier and more efficient basis generation for $E[2^f]$.

Theorem 4. Let $E : y^2 = (x - \lambda_1)(x - \lambda_2)(x - \lambda_3)$ be an elliptic curve over $\mathbb{F}_{p^2}$ with $E[2^f] \subseteq E(\mathbb{F}_{p^2})$ the full 2-power torsion. Let $L_i = (\lambda_i, 0)$ denote the points of order 2 and $[2]E$ denote the image of E under $[2] : P \mapsto P + P$ so that $E \setminus [2]E$ are the points with full 2^f-torsion. Then

$$Q \in [2]E \text{ if and only if } x_Q - \lambda_i \text{ is square for } i = 1, 2, 3.$$

More specifically, for $Q \in E \setminus [2]E$, Q is above L_i if and only if $x_Q - \lambda_i$ is square and $x_Q - \lambda_j$ is non-square for $j \neq i$.

Proof. It is well-known that $Q = (x, y) \in [2]E$ if and only if $x - \lambda_1$, $x - \lambda_2$ and $x - \lambda_3$ are all three squares [212, Ch. 1, Thm. 4.1]. Thus, for $Q \in E \setminus [2]E$, one of these three values must be a square, and the others non-squares, as their product must be y^2, hence square. We proceed similarly as the proof of [258, Thm. 3]. Namely, let P_1, P_2 and P_3 denote points of order 2^f above $L_1 = (\lambda_1, 0)$, $L_2 = (\lambda_2, 0)$ and $L_3 = (\lambda_3, 0)$, respectively. A point $Q \in E \setminus [2]E$ must lie above one of the L_i. Therefore, the reduced Tate pairing of degree 2^f of P_i and Q gives a primitive 2^f-th root of unity if and only if Q is not above L_i. Let $\zeta_i = e_{2^f}(P_i, Q)$, then by [187, Thm. IX.9] we have

$$\zeta_i^{2^{f-1}} = e_2(L_i, Q).$$

We can compute $e_2(L_i, Q)$ by evaluating a Miller function f_{2,L_i} in Q, where $\operatorname{div} f_{2,L_i} = 2(L_i) - 2(\mathcal{O})$. The simplest option is the line that doubles L_i, that is, $f_{2,L_i}(x, y) = x - \lambda_i$, hence

$$e_2(L_i, Q) = (x_Q - \lambda_i)^{\frac{p^2-1}{2}}.$$

Applying Euler's criterion to this last term, we get that if $x_Q - \lambda_i$ is square, then ζ_i is not a primitive 2^f-th root and hence Q must be above L_i, whereas if $x_Q - \lambda_i$ is non-square, then ζ_i is a primitive 2^f-th root and hence Q is not above L_i. □

Note that for supersingular Montgomery curves $y^2 = x^3 + Ax^2 + x = x(x - \alpha)(x - 1/\alpha)$, the theorem above tells us that non-squareness of x_Q for $Q \in$

$E(\mathbb{F}_{p^2})$ is enough to imply Q has full 2^f-torsion and is not above $(0,0)$ [258, Thm. 3].

Finding points with 2^f-torsion above $(0,0)$.

We describe two methods to efficiently sample Q above $(0,0)$, based on Theorem 4.

1. **Direct x sampling.** By deterministically sampling $x_Q \in \mathbb{F}_p$, we ensure that x_Q is square in $\mathbb{F}_{p^2}$. Hence, if Q lies on E and $x_Q - \alpha \in \mathbb{F}_{p^2}$ is non-square, where α is a root of $x^2 + Ax + 1$, then Theorem 4 ensures that $Q \in E \setminus [2]E$ and above $(0,0)$.

2. **Smart x sampling.** We can improve this using the fact that α is always square [15, 125]. Hence, if we find $z \in \mathbb{F}_{p^2}$ such that z is square and $z-1$ is non-square, we can choose $x_Q = z\alpha$ square and in turn $x_Q - \alpha = (z-1)\alpha$ non-square. Again, by Theorem 4 if Q is on E, this ensures Q is above $(0,0)$ and contains full 2^f-torsion. Hence, we prepare a list $[z_1, z_2, \ldots]$ of such values z for a given prime, and try $x_j = z_j\alpha$ until x_j is on E.

Both methods require computing α, dominated by one $\mathbb{F}_{p^2}$-square root. Direct sampling computes a Legendre symbol of $x^3 + Ax^2 + x$ per x to check if the corresponding point lies on E. If so, we check if $x - \alpha$ is non-square via the Legendre symbol. On average, this requires four samplings of x and six Legendre symbols to find a suitable x_Q with $Q \in E(\mathbb{F}_{p^2})$, and, given that we can choose x_Q to be small, we can use fast scalar multiplication on x_Q (see Section A).

In addition to computing α, smart sampling requires the Legendre symbol computation of $x^3 + Ax^2 + x$ per x. On average, we require two samplings of an x to find a suitable x_Q, hence saving four Legendre symbols in comparison to direct sampling. However, we can no longer choose x_Q small, which means that improved scalar multiplication for small x_Q is not available.

Finding points with 2^f-torsion not *above $(0,0)$.*

As shown in [258], we find a point P with full 2^f-torsion *not* above $(0,0)$ by selecting a point on the curve with non-square x-coordinate. Non-squareness depends only on p, not on E, so a list of small non-square values can be precomputed. In this way, finding such a point P simply becomes finding the first value x_P in this list such that the point $(x_P, -)$ lies on $E(\mathbb{F}_{p^2})$, that is,

$x_P^3 + Ax_P^2 + x_P$ is square. On average, this requires two samplings of x, hence two Legendre symbol computations.

5.2 GENERAL IMPROVEMENTS TO VERIFICATION

In this section, we describe improvements to SQIsign verification and present new optimisations, decreasing the cost of the three main subroutines of verification.

Known techniques from literature.

There are several state-of-the-art techniques in the literature on efficient implementations of elliptic curve or isogeny-based schemes that allow for general improvements to verification, but are not included in SQIsign (NIST). We implemented such methods, e.g., to improve scalar multiplication $P \mapsto [n]P$ and square roots. The details are described in Section A. In particular, we use that $P \mapsto [n]P$ is faster when x_P is small.

Improving the subroutine FindBasis.

In SQIsign (NIST), to find a complete basis for $E[2^f]$ we are given a point $Q \in E[2^f]$ lying above $(0,0)$ and need to find another point $P \in E(\mathbb{F}_{p^2})$ of order 2^f not lying above $(0,0)$. We sample P directly using x_P non-square, as described above and demonstrated by [258], and in particular can choose x_P small. We then compute $P \leftarrow [\frac{p+1}{2^f}]P$ via fast scalar multiplication to complete the torsion basis (P, Q).

Improved strategies for ComputeIsogeny.

Recall that ComputeIsogeny follows three steps in SQIsign (NIST): it first computes a 4-isogeny that may contain $(0,0)$ in the kernel, and a 2-isogeny if f is odd, before entering an optimal strategy for computing the remaining chain of 4-isogenies. However, the first two steps include many costly doublings. We improve this by adding these first two steps in the optimal strategy. If f is even, this is straightforward, with a simple check for $(0,0)$ in the kernel in the first step. For odd f, we add the additional 2-isogeny in this first step.[11]

11 In particular, we compute $R' = [2^{f-3}]K$ and $R = [2]R'$, a 4-isogeny with kernel $\langle R \rangle$, push R' through, and compute a 2-isogeny with kernel $\langle R' \rangle$.

For simplicity of the implementation, we determine optimal strategies as in SIKE [221], thus we assume that only 4-isogenies are used.

Note that techniques for strategies with variable isogeny degrees are available from the literature on CSIDH implementations [107]. However, the performance difference is very small, hence our simplified approach appears to be preferable.

In addition to optimising 4-isogeny chains, we implemented optimised 3-isogeny chains from SIKE [221] for the computation of $\hat{\varphi}_{\text{chall}}$ when $f < 128$.

5.3 TO PUSH, OR NOT TO PUSH – THAT IS, THE Q

In SQIsign (NIST), the point Q is pushed through φ so that we easily get the basis point above $(0,0)$ on the image curve, and we can then use Theorem 4 to sample the second basis point P. Instead of pushing Q, one can also use Theorem 4 to efficiently sample this basis point Q above $(0,0)$. Although pushing Q seems very efficient, the more we increase f, the longer we are pushing Q through isogeny chains. Instead, sampling becomes increasingly more efficient when f grows as the cost of scalar multiplication by $\frac{p+1}{2^f}$ decreases. Furthermore, sampling *both* P and Q allows us to use those points as an *implicit basis* for $E[2^f]$, even if their orders are multiples of 2^f, as described in more detail below. We observe experimentally that this makes sampling Q, instead of pushing Q, more efficient for $f > 128$.

Using implicit bases.

Using Theorem 4, it is possible to find points P and Q efficiently so that both have full 2^f-torsion. The pair (P, Q) is not an *explicit basis* for $E[2^f]$, as the orders of these points are likely to be multiples of 2^f. However, instead of multiplying both points by the cofactor to find an explicit basis, we can use these points implicitly, as if they were a basis for $E[2^f]$. This allows us to compute $K = P + [s]Q$ first, and only then multiply K by the cofactor. This saves a full scalar multiplication by the cofactor $\frac{p+1}{2^f}$. We refer to such a pair (P, Q) as an *implicit basis* of $E[2^f]$. Algorithmically, implicit bases combine FindBasis and FindKernel into a single routine FindBasisAndKernel.

5.4 IMPROVED CHALLENGE FOR $f \geq \lambda$

Recall from Section 4.2 that when $f \geq \lambda$, we can simply set $D_{\text{chall}} = 2^\lambda$. This decreases the cost of FindBasis for the challenge computation considerably, as we can now use Theorem 4 to find a basis for $E[2^\lambda]$.

Improving FindBasis *for the challenge isogeny when* $f \geq \lambda$.

We use Theorem 4 twice, first to find P not above $(0,0)$ having full 2^f-torsion and then to find Q above $(0,0)$ having full 2^f-torsion. We choose x_P and x_Q small such that faster scalar multiplication is available. We find the basis for $E[2^\lambda]$ by $P \leftarrow [\frac{p+1}{2^f}]P$ followed by $f - \lambda$ doublings, and $Q \leftarrow [\frac{p+1}{2^f}]Q$ followed by $f - \lambda$ doublings.[12] Alternatively, if Q is pushed through isogenies, we can reuse $Q \leftarrow \varphi^{(n)}(Q^{(n)}) \in E[2^f]$ from the computation of the last step of φ_{resp}, so that we get a basis point for $E[2^\lambda]$ by $f - \lambda$ doublings of Q. Reusing this point Q also guarantees cyclicity of $\hat{\varphi}_{\text{chall}} \circ \varphi_{\text{resp}}$.

Remark 7. For SQIsign without extension fields, obtaining $f \geq \lambda$ seems infeasible, hence the degree D of φ_{chall} is $2^f \cdot 3^g$. Nevertheless, some optimizations are possible in the computation of φ_{chall} in this case. FindBasis for $E[2^f \cdot 3^g]$ benefits from similar techniques as previously used in SIDH/SIKE, as we can apply known methods to improve generating a torsion basis for $E[3^g]$ coming from 3-descent [130, § 3.3]. Such methods are an analogue to generating a basis for $E[2^f]$ as described in Theorem 4 and [258, Thm. 3].

§6 Size-Speed Trade-Offs in SQIsign Signatures

The increase in f also enables several size-speed trade-offs by adding further information in the signature or by using uncompressed signatures. Some trade-offs were already present in earlier versions of SQIsign [148], however, by using large f and the improvements from Section 5, they become especially worthwhile.

We take a slightly different stance from previous work on SQIsign as for many use cases the main road block to using SQIsign is the efficiency of verification in cycles. In contrast, in several applications the precise size of a signature is less important as long as it is below a certain threshold.[13] For example,

12 Algorithmically, this is faster than a single scalar multiplication by $2^{f-\lambda} \cdot \frac{p+1}{2^f}$.

13 See https://blog.cloudflare.com/sizing-up-post-quantum-signatures/.

many applications can handle the combined public key and signature size of RSA-2048 of 528 bytes, while SQIsign (NIST) features a combined size of only 241 bytes. In this section, we take the 528 bytes of RSA-2048 as a baseline, and explore size-speed trade-offs for SQIsign verification with data sizes up to this range.

We note that the larger signatures in this section encode the same information as standard SQIsign signatures, hence have no impact on the security.

6.1 ADDING SEEDS FOR THE TORSION BASIS IN THE SIGNATURE

We revisit an idea that was previously present in SQIsign verification [148] (but no longer in [100] or [149]), and highlight its particular merits whenever $f \geq \lambda$, as enabled by signing with extension fields. So far, we have assumed that completing or sampling a basis for $E[2^f]$ is done by deterministically sampling points. Recall from Section 5.1 that sampling x_P resp. x_Q (when not pushing Q) on average requires the computation of several Legendre symbols resp. square roots. We instead suggest using a seed to find x_P (when pushing Q) or x_P and x_Q (otherwise), which we include in the signature, so that the verifier saves all of the above cost for finding x_P, resp. x_Q. Finding these seeds adds negligible overhead for the signer, while verification performance improves. Signer and verifier are assumed to agree upon all precomputed values.

Seeding a point not *above* $(0,0)$.

For x_P *not* above $(0,0)$, we fix a large enough $k > 0$ and precompute the 2^k smallest values $u_j \in \mathbb{F}_p$ such that $u_j + i \in \mathbb{F}_{p^2}$ is non-square (where i is the same as in Section 5). During signing, we pick the smallest u_j such that $x_P = u_j + i$ is the x-coordinate of a point $P \in E(\mathbb{F}_{p^2})$, and add the index j to the signature as a seed for x_P. Theorem 4 ensures that any $P \in E(\mathbb{F}_{p^2})$ for non-square x_P is a point with full 2^f-torsion not above $(0,0)$. This furthermore has the advantage of fast scalar multiplication for x_P as the x-coordinate is very small.

Seeding a point above $(0,0)$.

As noted above, when f is large, it is faster to deterministically compute a point of order 2^f above $(0,0)$ than to push Q through φ. We propose a similar seed here for fixed large enough $k > 0$, using Theorem 4 and the "direct sampling" approach from Section 5.1. During signing, we pick the smallest $j \leq 2^k$ such that $x_Q = j$ is the x-coordinate of a point $Q \in E(\mathbb{F}_{p^2})$ and $x_Q - \alpha$ is non-square.

We add $x_Q = j$ to the signature as seed. Note that when using both seeding techniques, we do not explicitly compute $[\frac{p+1}{2^f}]P$ or $[\frac{p+1}{2^f}]Q$, but rather use the seeded points P and Q as an implicit basis, as described in Section 5.3.

Size of seeds.

Per seeded point, we add k bits to the signature size. Thus, we must balance k such that signatures are not becoming too large, while failure probabilities for not finding a suitable seed are small enough. In particular, seeding x_P resp. x_Q via direct sampling has a failure probability of $\frac{1}{2}$ resp. $\frac{3}{4}$ per precomputed value. For the sake of simplicity, we set $k = 8$ for both seeds, such that every seed can be encoded as a separate byte.[14] This means that the failure rate for seeding Q is $(\frac{3}{4})^{256} \approx 2^{-106.25}$ for our choice, while for P it is 2^{-256}. Theoretically it is still possible that seeding failures occur. In such a case, we simply recompute KLPT. We furthermore include similar seeds for the torsion basis on E_A and E_2, giving a size increase of $(n+1) \cdot 2$ bytes.

The synergy with large f now becomes apparent. The larger f gets, the fewer blocks n are required, hence adding fewer seeds overall. For $f = 75$, the seeds require an additional 28 bytes when seeding both P and Q. For $f = 122, 140, 163, 195, 244$ this drops to 18, 16, 14, 12, and 10 additional bytes, respectively, to the overall signature size of 177 bytes for NIST Level I security.

Remark 8. Instead of using direct sampling for Q with failure probability $\frac{3}{4}$, we can reduce it to $\frac{1}{2}$ via "smart sampling" (see Section 5.1). However, this requires the verifier to compute α via a square root to set $x_Q = z\alpha$ with seeded z. We thus prefer direct sampling for seeded Q, which incurs no such extra cost.

6.2 UNCOMPRESSED SIGNATURES

In cases where f is very large, and hence the number of blocks is small, in certain cases it is worthwhile to replace the value s in the signature by the full x-coordinate of $K = P + [s]Q$. In essence, this is the uncompressed version of the SQIsign signature σ, and we thus refer to this variant as *uncompressed* SQIsign.

14 Note that for equal failing rates the number of possible seeds for P can be chosen smaller than for Q, hence slightly decreasing the additional data sizes.

Speed of uncompressed signatures.

Adding the precise kernel point K removes the need for both FindBasis and FindKernel, leaving ComputeIsogeny as the sole remaining cost. This speed-up is significant, and leaves little room for improvement beyond optimizing the cost of computing isogenies. The cost of verification in this case is relatively constant, as computing an 2^e-isogeny given the kernels is only slightly affected by the size of f, as is visible in the black dashed line in Figure 17. This makes uncompressed SQIsign an attractive alternative in cases where the signature size, up to a certain bound, is less relevant.

Size of uncompressed signatures.

Per step, this increases the size from $\log(s) \approx f$ to $2 \cdot \log(p)$ bits, which is still relatively size efficient when f is close to $\log(p)$. For recomputing φ_{chall}, we take a slightly different approach than before. We add the Montgomery coefficient of E_1 to the signature, and seeds for a basis of $E_1[2^f]$. From this, the verifier can compute the kernel generator of φ_{chall}, and verify that the j-invariant of its codomain matches E_2. Hence this adds $2 \cdot \log(p)$ bits for E_1 and two bytes for seeds to the signature, for a total of $(n+1) \cdot (\log p/4) + 2$ bytes.

For $f = 244$, this approach less than doubles the signature size from 177 bytes to 322 bytes for NIST Level I security. For $f = 145$, the signature becomes approximately 514 bytes, while for the current NIST Level I prime with $f = 75$, the size would become 898 bytes. When adding the public key size of 64 bytes, especially the first two cases still appear to be reasonable alternatives to RSA-2048's combined data size of 528 bytes.

Remark 9. Uncompressed signatures significantly simplify verification. Many functionalities required for compressed signatures are no longer necessary. which allows for a much more compact code base. This becomes relevant for use cases featuring embedded devices with strict memory requirements.

§7 Primes and Performance

In this section we show the performance of verification for varying f, using the optimisations from the previous sections. Further, we find specific primes with suitable f for $n = 4$ and $n = 7$, and report their signing performance using our SageMath implementation, comparing it with the current SQIsign (NIST) prime.

7.1 PERFORMANCE OF OPTIMISED VERIFICATION

To compare the verification performance of our optimised variants with compressed signatures to SQIsign (NIST) and SQIsign (LWXZ),[15] we run benchmarks in the same setting as in Section 4.3. In particular, Figure 17 shows the cost of verification for the NIST Level I primes $p^{(f)}$ for $50 \leq f \leq 250$. As before, we sample random public key curves and signatures σ of the correct form instead of using signatures generated by the SQIsign signing procedure.

For the sake of simplicity, Figure 17 displays only the fastest compressed This work. variant, namely the version that does not push Q through isogenies and uses seeds to sample P and Q. This variant significantly outperforms both SQIsign (NIST) and SQIsign (LWXZ) already at $f = 75$, at the cost of slightly larger signatures. A detailed description and comparison of all four compressed variants is in Section C, which shows that our unseeded variants achieve similar large speed-ups with no increase in signature size. Lastly, the uncompressed variant achieves the fastest speed, although at a significant increase in signature size.

7.2 FINDING SPECIFIC PRIMES

We now give two example primes, one prime optimal for 4-block verification, as well as the best we found for 7-block verification. The "quality" of a prime p is measured using the cost metric $\textsc{SigningCost}_p$ defined in Section 3.3.

Optimal 4-block primes.

For 4-block primes, taking $e = 975$ as a baseline, we need f bigger than 244. In other words, we are searching for primes of the form

$$p = 2^{244}N - 1$$

where $N \in [2^4, 2^{12}]$ (accepting primes between 250 and 256 bits). This search space is quickly exhausted. For each prime of this form, we find the optimal torsion T to use, minimising $\textsc{SigningCost}_p(T)$. The prime with the lowest total cost in this metric, which we denote p_4, is

$$p_4 = 2^{246} \cdot 3 \cdot 67 - 1$$

15 Our implementation of SQIsign (LWXZ) [258] is identical to SQIsign (NIST) except for the improved sampling of P described in Section 5.1.

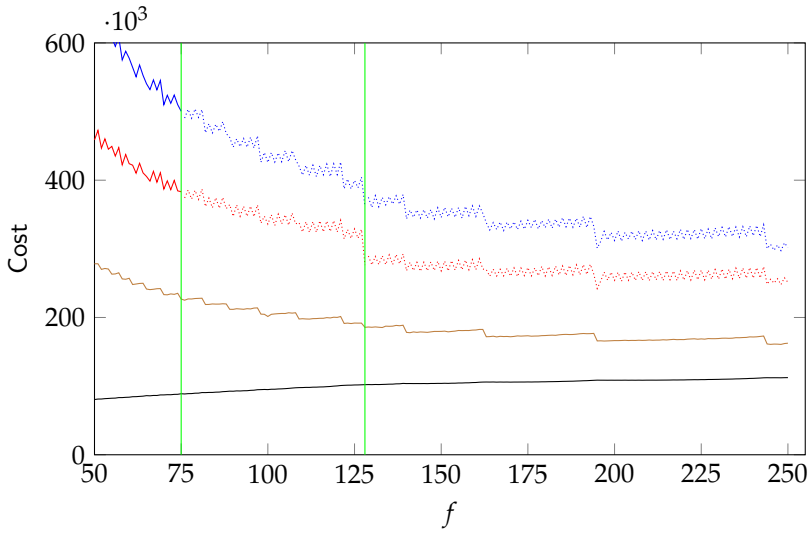


Figure 17: Extended version of Figure 16 showing the cost in $\mathbb{F}_p$-multiplications for verification at NIST Level I security, for varying f and $p^{(f)}$, averaged over 1024 runs per prime. In addition to SQIsign (NIST) in blue, it shows the performance of SQIsign (LWXZ) in red, our fastest compressed AprèsSQI variant in brown, and uncompressed AprèsSQI in black.

Balanced primes.

Additionally, we look for primes that get above the significant $f > 128$ line, while minimizing $\textsc{SigningCost}_p(T)$. To do this, we adopt the "sieve-and-boost" technique used to find the current SQIsign primes [100, §5.2.1]. However, instead of looking for divisors of $p^2 - 1$, we follow Theorem 2 and look for divisors of

$$\prod_{n=1}^{k} \Phi_n(p^2)/2$$

to find a list of good candidate primes. This list is then sorted with respect to their signing cost according to $\textsc{SigningCost}_p$. The prime with the lowest signing cost we could find, which we call p_7, is

$$p_7 = 2^{145} \cdot 3^9 \cdot 59^3 \cdot 311^3 \cdot 317^3 \cdot 503^3 - 1.$$

Remark 10. This method of searching for primes is optimised for looking for divisors of $p^2 - 1$, hence it might be suboptimal in the case of allowing torsion in higher extension fields. We leave it as future work to find methods which further take advantage of added flexibility in the prime search.

7.3 PERFORMANCE FOR SPECIFIC PRIMES

We now compare the performance of the specific primes p_4, p_7, as well as the current NIST Level I prime p_{1973} used in SQIsign (NIST).

Signing performance.

We give a summary of the estimated signing costs in Table 12. For p_{1973}, we adjust the signing metric as SigningCost with the isogeny computations scaling as $\sqrt{\ell}\log\ell$ to (rather optimistically) account for the benefit of $\sqrt{\text{élu}}$. Further, we ran our proof-of-concept SageMath implementation on the three primes, using SageMath 9.8, on a laptop with an Intel-Core i5-1038NG7 processor, averaged over five runs. An optimised C implementation will be orders of magnitude faster; we use these timings simply for comparison.

Table 12: Comparison between estimated cost of signing for three different primes. The third column indicates the maximum field extension required to sign. Signing cost for p_{1973} is adjusted to account for the use of $\sqrt{\text{élu}}$.

p	Smoothness	Max. ext.	$\text{SigningCost}_p(T)$	Timing
p_{1973}	1973	$k = 1$	1956.5*	11m, 32s
p_7	997	$k = 23$	4137.9	9m, 20s
p_4	2293	$k = 53$	9632.7	15m, 52s

We note that the SigningCost-metric somewhat predicts the signing performance of the primes, though the performance difference is smaller than predicted. A possible explanation for this is that the SigningCost-metric ignores all overhead, such as quaternion operations, which roughly adds similar amounts of cost per prime.

Our implementation uses $\sqrt{\text{élu}}$ whenever the kernel generator is defined over $\mathbb{F}_{p^2}$ and ℓ is bigger than a certain crossover point. This mainly benefits p_{1973}, as this prime only uses kernel generators defined over $\mathbb{F}_{p^2}$. The crossover point is experimentally found to be around $\ell > 300$ in our implementation, which

is not optimal, compared to an optimised C implementation.[16] Nevertheless, we believe that these timings, together with the cost metrics, provide sufficient evidence that extension-field signing in an optimised implementation stays in the same order of magnitude for signing time as staying over $\mathbb{F}_{p^2}$.

Verification performance.

In Table 13, we summarise the performance of verification for p_{1973}, p_7, and p_4, both in terms of speed, and signature sizes.

Table 13: Comparison between verification cost for different variants and primes, with cost in terms of 10^3 $\mathbb{F}_p$-multiplications, using $\mathbf{S} = 0.8 \cdot \mathbf{M}$.

p	f	Impl.	Variant	Verif. cost	Sig. size
p_{1973}	75	[100]	-	500.4	177 B
		[258]	-	383.1	177 B
		This work.	unseeded	276.1	177 B
		This work.	seeded	226.8	195 B
p_7	145	This work.	unseeded	211.0	177 B
		This work.	seeded	178.6	193 B
		This work.	uncompr.	103.7	514 B
p_4	246	This work.	unseeded	185.2	177 B
		This work.	seeded	160.8	187 B
		This work.	uncompr.	112.2	322 B

Two highlights of this work lie in using p_7, both with and without seeds, having (almost) the same signature sizes as the current SQIsign signatures, but achieving a speed-up of factor 2.37 resp. 2.80 in comparison to SQIsign (NIST) and 1.82 resp. 2.15 in comparison to SQIsign (LWXZ), using p_{1973}. Another interesting alternative is using uncompressed p_4, at the cost of roughly double signature sizes, giving a speed-up of factor 4.46 in comparison to SQIsign (NIST) and 3.41 in comparison to SQIsign (LWXZ).

Remark 11. We analyse and optimise the cost of verification with respect to $\mathbb{F}_p$-operations. However, primes of the form $p = 2^f \cdot c - 1$ are considered to be particularly suitable for fast optimised finite field arithmetic, especially when f is large [18]. Hence, we expect primes like p_4 to improve significantly more in

16 For instance, work by Adj, Chi-Domínguez, and Rodríguez-Henríquez [3] gives the crossover point at $\ell > 89$, although for isogenies defined over $\mathbb{F}_p$.

comparison to p_{1973} in low-level field arithmetic, leading to a larger speed-up than predicted in Table 13. Furthermore, other low-level improvements, such as fast non-constant time GCD for inversions or Legendre symbols, will improve the performance of primes in terms of cycles, which is unaccounted for by our cost metric.

§A Curve Arithmetic

In this section we describe in detail the known techniques from literature that allow for general improvements to verification, but that are not included in SQIsign (NIST).

We use $\texttt{xDBL}(x_P)$ to denote x-only point doubling of a point P and similarly $\texttt{xADD}(x_P, x_Q, x_{P-Q})$ to denote x-only differential addition of points P and Q. We use $\texttt{xMUL}(x_P, m)$ to denote x-only scalar multiplication of a point P by the scalar m.

1.1 FASTER SCALAR MULTIPLICATIONS

We describe three improvements to the performance of xMUL, that can be applied in different situations during verification.

1. **Affine A.** Throughout verification and specifically in FindBasis and FindKernel, we work with the Montgomery coordinate A in projective form. However, some operations, such as computing the point difference x_{P-Q} given x_P and x_Q require A in affine form. Having an affine A allows an additional speed-up, as xDBL requires one $\mathbb{F}_{p^2}$-multiplication less in this case. Thus, xMUL with affine A is cheaper by 3 **M** per bit of the scalar.

2. **Affine points.** Using batched inversion, whenever we require A in affine form we can get x_P and x_Q in affine form for almost no extra cost. An xMUL with affine x_P or x_Q saves another $\mathbb{F}_{p^2}$-multiplication, hence again 3 **M** per bit of the scalar.

3. **Small x-coordinate.** For a point P with $x_P = a + bi$ with small a and b, we can replace an $\mathbb{F}_{p^2}$-multiplication by x_P with $a + b$ additions. This, in turn, saves almost 3 **M** per bit of the scalar in any xMUL of x_P.

As we can force P and Q to have $b \in \{0, 1\}$ and small a when sampling them in FindBasis, these points are affine and have small x-coordinates. Together with the affine A, this saves almost 9 **M** per bit for such scalar multiplications, saving roughly 27% per xMUL. We call such an xMUL a *fast* xMUL. Whenever xMUL uses 2 of these optimisations, we call it *semi-fast*.

Whenever possible, we use differential addition chains [41] to improve scalar multiplications by certain system parameters, such as $\frac{p+1}{2^f}$. In particular, we will only need to multiply by a few, predetermined scalars, and therefore we follow the method described by Cervantes-Vázquez, Chenu, Chi-Domínguez,

De Feo, Rodríguez-Henríquez, and Smith [98, §4.2]. Our optimal differential addition chains were precomputed using the CTIDH software [20].

1.2 FASTER SQUARE ROOTS

We apply several techniques from the literature to further optimise low-level arithmetic in all of verification. The most significant of these is implementing faster square roots in $\mathbb{F}_{p^2}$ [343, §5.3], which decreases the cost of finding square roots to two $\mathbb{F}_p$-exponentiations and a few multiplications.

1.3 PROJECTIVE POINT DIFFERENCE

The implementation of SQIsign (NIST) switches between affine and projective representations for x_P, x_Q and A within each block. It does so to be able to derive the point difference x_{P-Q} from x_P and x_Q in order to complete the basis P, Q in terms of x-coordinates. However, it is possible to compute the point difference entirely projectively using Proposition 3 from [326]. This allows us to stay projective during the SQIsign (NIST) verification until we reach E_2, where we do normalization of the curve. This saves costly inversions during verification and has the additional benefit of improved elegance for SQIsign (NIST).

However, in our variant of verification, we make no use of projective point difference, as the improvements of Section 5 seem to outperform this already.

§B ALGORITHMS

The bottleneck of SQIsign verification is the computation of an isogeny of fixed degree 2^e, which is computed as $\lceil e/f \rceil$ isogenies of degree 2^f, where $f \leq e$. Each such 2^f-isogeny is called a *block*. In this section, we present algorithms for the computation of a single block in verification of SQIsign (NIST) (see Algorithm 18) and the computation using the improvements described in Sections 5 and 6 (see Algorithm 19).

§C PERFORMANCE OF OPTIMISED VERIFICATION

The optimisations for compressed variants from Section 5 and Section 6 allow for several variants of verification, depending on using seeds and pushing Q

Algorithm 18 Single block in verification of SQIsign (NIST)

Input: Affine coeff. $A \in \mathbb{F}_p$, a basis x_P, x_Q, x_{P-Q} for $E_A[2^f]$ with Q above $(0,0)$ and $s \in \mathbb{Z}/2^f\mathbb{Z}$ defining a kernel
Output: Affine coeff. $A' \in \mathbb{F}_p$ as the codomain of $E_A \to E_{A'}$ of degree 2^f, with a basis x_P, x_Q, x_{P-Q} for $E'_A[2^f]$ with Q above $(0,0)$

1: $K \leftarrow \texttt{3ptLadder}(x_P, x_Q, x_{P-Q}, s, A)$
2: $A_{\text{proj.}}, x_Q \leftarrow \texttt{FourIsogenyChain}(K, x_Q, A)$
3: $A, x_Q \leftarrow \texttt{ProjectiveToAffine}(A_{\text{proj.}}, x_Q)$
4: $x_P, x_{P-Q} \leftarrow \texttt{CompleteBasis}_{2^f}(x_Q, A)$
5: **return** A, x_P, x_Q, x_{P-Q}

Algorithm 19 Single block in verification using improvements of Section 5

Input: Projective coeff. $A \in \mathbb{F}_p$, a seed (n, m) and $s \in \mathbb{Z}/2^f\mathbb{Z}$ defining a kernel
Output: Affine coeff. $A' \in \mathbb{F}_p$ as the codomain of $E_A \to E_{A'}$ of degree 2^f

1: $x_P \leftarrow \texttt{SmallNonSquare}(m)$, $x_Q \leftarrow n$
2: $x_{P-Q} \leftarrow \texttt{PointDifference}(x_P, x_Q, A)$ ▷ implicit basis x_P, x_Q, x_{P-Q}
3: $K \leftarrow \texttt{3ptLadder}(x_P, x_Q, x_{P-Q}, s, A)$
4: $K \leftarrow \texttt{xMUL}(x_K, \frac{p+1}{2^f}, A)$ ▷ semi-fast xMUL
5: $A_{\text{proj.}} \leftarrow \texttt{FourIsogenyChain}(K, A)$
6: $A \leftarrow \texttt{ProjectiveToAffine}(A_{\text{proj.}})$
7: **return** A

through isogenies. We summarise the four resulting approaches and measure their performance.

3.1 FOUR APPROACHES FOR VERIFICATION

To obtain our measurements, we combine our optimisations to give four different approaches to perform SQIsign verification, specifically optimised for $f \geq \lambda$. Firstly, we either push Q through φ in every block, or sample Q. Secondly, we either sample the basis or seed it.

Pushing Q, sampling P without seed.

This variant is closest to the original SQIsign (NIST) and SQIsign (LWXZ) implementations. It is the optimal version for non-seeded verification for $f \leq 128$, using the general optimisations from Section 5.2 and the challenge optimisations from Section 5.4.

Not pushing Q, sampling both P and Q without seed.

This variant competes with the previous version in terms of signature size. Due to Section 5.3, sampling a new Q is more efficient than pushing Q for large f.[17] This is the optimal version for non-seeded verification for $f > 128$, and additionally uses the optimisations from Section 5.3.

Pushing Q, sampling P with seed.

This variant only adds seeds to the signature to describe x_P. As such, it lies between the other three variants in terms of both signature size and speed. The signature is 1 byte per block larger than the unseeded variants, and 1 byte per block smaller than the variant where x_Q is seeded too. In terms of speed, it is faster than the variants where P is unseeded, but slower than the variant where Q is seeded too. It uses the optimisations from Sections 5.2 and 5.4, but cannot benefit from the kernel computation via implicit bases from Section 5.3.

Not pushing Q and sampling both P and Q with seed.

This is the fastest compressed version that we present in this work. Although it adds 2 bytes per block, the small number of blocks n for large f makes the

17 Based on benchmarking results, we sample Q with $x = n\alpha$ for $f < 200$ and directly for $f \geq 200$.

total increase in signature size small. All the optimisations from Section 5 now apply: we additionally have fast `xMUL` for Q, as well as the optimised implicit basis method to compute the kernel and optimised challenge. An algorithmic description of a single block in this version is given in Algorithm 19.

3.2 PERFORMANCE BENCHMARK

We benchmarked these four approaches according to our cost metric by taking the average over 1024 random signatures. The results are given in Figure 18 showing the significant increase in performance compared to SQIsign (NIST) and SQIsign (LWXZ), as well as the additional performance gained from seeding. For comparison, we also show the performance when using uncompressed signatures as a lower bound.

§D Detailed Information on Primes

We give more details on the specific primes used in Section 7.
The prime p_7 is used for a verification with $n = 7$ blocks. It achieves $f = 145$, with T given as below.

$$
\begin{aligned}
p_7 &= \texttt{0x309c04bcaedbb0134cca8373e439ffffffffffffffffffffffffffff} \\
T &= 3^7 \cdot 5^4 \cdot 7^2 \cdot 11 \cdot 13 \cdot 17 \cdot 19^2 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 41 \cdot 43 \cdot 47 \cdot 53^2 \cdot 59^3 \cdot 61 \cdot 67 \\
&\quad \cdot 71 \cdot 73 \cdot 79 \cdot 109 \cdot 113 \cdot 131 \cdot 157 \cdot 181 \cdot 193 \cdot 223 \cdot 239 \cdot 241 \cdot 271 \cdot 283 \cdot 311^3 \\
&\quad \cdot 317^3 \cdot 331 \cdot 349 \cdot 503^2 \cdot 859 \cdot 997 \\
\textsc{SigningCost}_{p_7}(T) &= 4137.91235
\end{aligned}
$$

The prime p_4 is used for a verification with $n = 4$ blocks. It achieves $f = 246$, with T given as below.

$$
\begin{aligned}
p_4 &= \texttt{0x323fffffffffffffffffffffffffffffffffffffffffffffffffffff} \\
T &= 3^3 \cdot 5^2 \cdot 7^2 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 41 \cdot 43 \cdot 47 \cdot 53 \cdot 59 \cdot 61 \cdot 67 \cdot 71 \\
&\quad \cdot 73 \cdot 79 \cdot 83 \cdot 89 \cdot 97 \cdot 101 \cdot 103 \cdot 107 \cdot 109 \cdot 113 \cdot 127 \cdot 149 \cdot 151 \cdot 157 \cdot 163 \cdot 181 \\
&\quad \cdot 197 \cdot 211 \cdot 229 \cdot 241 \cdot 271 \cdot 317 \cdot 397 \cdot 577 \cdot 593 \cdot 641 \cdot 661 \cdot 757 \cdot 1069 \cdot 2293 \\
\textsc{SigningCost}_{p_4}(T) &= 9632.7307
\end{aligned}
$$

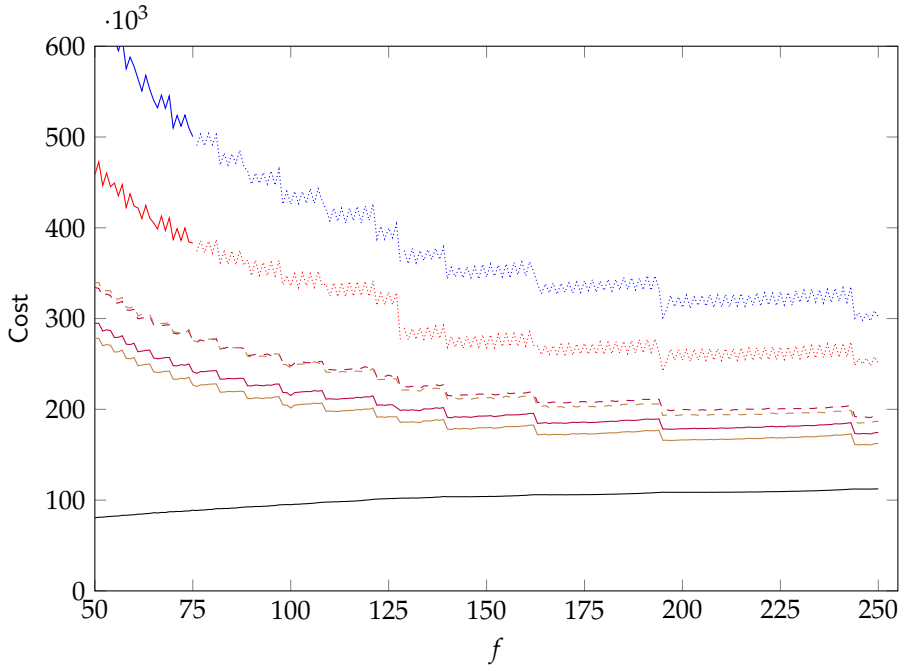


Figure 18: Extended version of Figure 17 showing the cost in $\mathbb{F}_p$-multiplications for verification at NIST-I security level, for varying f and $p^{(f)}$, averaged over 1024 runs per prime. In addition to SQIsign (NIST) in blue, and SQIsign (LWXZ) in red, it shows all AprèsSQI variants: In purple is the performance of AprèsSQI when pushing Q, with dashed blue when not seeding P. In brown is the performance of AprèsSQI when not pushing Q, with dashed brown when not seeding P, Q. The performance of uncompressed AprèsSQI is shown in black.

X

OPTIMIZED ONE-DIMENSIONAL SQISIGN VERIFICATION

We report record-breaking one-dimensional SQIsign verification times: below two milliseconds (5.6 Mcycles) down to a few hundred microseconds (1.3 Mcycles) using 5-core parallelization. SQIsign is a well-known post-quantum signature scheme due to its small combined signature and public-key size. However, SQIsign suffers from notably long signing times, and verification times are not short either. To improve, recent research explores both one-dimensional and two-dimensional SQIsign, both with very different characteristics. In particular with SQIsign2D, with good signing times and fast verification, the research focus has shifted to two-dimensional SQIsign as the main future direction. However, the absence of an optimized 1D verification implementation hampers a thorough comparison between these different variants.

This chapter bridges this gap in the literature: we provide a state-of-the-art implementation of 1D SQIsign verification, including novel optimizations. We achieve fast verification for several variants, matching and even surpassing two-dimensional variants. Moreover, we present the first implementation supporting both 32-bit and 64-bit processors and take advantage of the inherent parallelism available in isogeny computations. Furthermore, we present optimized assembly code for the Cortex-M4, integrated with the pqm4 project.

This chapter is based on the paper

> Marius A. Aardal, Gora Adj, Arwa Alblooshi, Diego F. Aranha, Isaac Canales-Martínez, Jorge Chávez-Saab, Décio Luiz Gazzoni Filho, Krijn Reijnders, and Francisco Rodríguez-Henríquez. "Optimized One-Dimensional SQIsign Verification on Intel and Cortex-M4". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025.1 (2025), pp. 1–25.

The paper was finalized slightly after I finalized the writing for this thesis and contents may therefore differ in details. I have abridged sections on cost comparisons between one-dimensional and two-dimensional SQIsign, and on the optimizations of finite-field-arithmetic.

Our results motivate further research into one-dimensional SQIsign, as it boasts unique features, even among isogeny-based schemes.

§1 Introduction

In June 2023, NIST initiated Round 1 of the Call for Additional Digital Signature Schemes [357], accepting 40 proposals based on diverse cryptographic problems. SQIsign is the sole isogeny-based scheme among these candidates and offers the smallest combined signature and public-key size of all post-quantum signature candidates. However, SQIsign's signing time is not only considerably slower than lattice-based schemes, but all current implementations are not constant-time [100], while verification remains moderately efficient. For this reason, it has been argued that SQIsign's potential lies in applications requiring long-term signatures, such as certificates, software verification, and resource-constrained devices performing only verification tasks.

Introduced at Asiacrypt 2020, SQIsign [148] has seen intense development since then. While an improved version preserving the original framework was presented in [149, 258], recent research has explored more substantial modifications to the underlying scheme. In Chapter IX, we proposed AprèsSQI, a variant specifically designed to prioritize faster verification at the cost of having the signer perform extension-field arithmetic. Our estimates show that AprèsSQI's verification can be up to 4× faster than SQIsign's, with only a factor two slowdown to signing. However, our results were measured in finite-field operations, as we did not have an optimized low-level implementation available. This chapter will fill preciesly this gap, with even more improvements.

Higher-dimensional techniques, introduced in the fallout of SIDH/SIKE [86, 266, 329], led to the development of SQIsignHD [141], which greatly improves signing speed compared to SQIsign, reporting an astonishing 40× acceleration factor. Key generation also benefits from similar speedups. However, SQIsignHD verification requires the evaluation of a 4-dimensional isogeny, which makes it significantly more expensive, raising serious concerns about its practicality. Recently, the algorithmic developments introduced in [287, 304] were exploited to create a two-dimensional (2D) variant of SQIsign, independently proposed in [34, 166, 288]. These variants leverage 2D-isogeny computations for both signing and verification. Compared to the original SQIsign, the performance improvements reported in [34] are incredible: signing achieves a 15× acceleration, and verification enjoys a 4× speedup. Moreover,

the security proofs of these 2D variants demonstrate increased rigor, essentially eliminating the heuristic assumptions required in 1D SQIsign.

Another interesting SQIsign variation was proposed by Onuki and Nakagawa in [303], which employs 2D techniques in the core ideal-to-isogeny algorithm to accelerate the costly isogeny computations in the signature procedure of the original SQIsign. This approach is notably different from the previous ones, as it uses 2D techniques for signature generation, while retaining a 1D verification. Like the other 2D variants, it also offers greater flexibility in prime selection, enabling the use of primes of the form $p = c \cdot 2^f - 1$, with c a small odd integer. This enables more efficient verification due to their higher two-torsion and fast arithmetic, as shown in AprèsSQI. In principle, their signing algorithm can replace the slow extenstion-field arithemtic required for AprèsSQI and therefore enables more practical 1D verification for SQIsign-like schemes. As we show in Chapter IX, the resulting signatures become amenable for parallelization, which we explore in this chapter.

The primary computational bottleneck in 2D verification is the computation of a 2D isogeny of degree approximately $(2^\lambda, 2^\lambda)$, with λ the security parameter, as described in [34, Algorithm 8]. Conversely, 1D verification involves the computation of a simpler but much longer 2^e-isogeny with $e \approx 7.5\lambda$, decomposed into $n = \lceil \frac{e}{f} \rceil$ blocks of 2^f-isogenies, with f as defined above. In this work, we carefully study how to make 1D verification as efficient as possible, which includes exploring and exploiting its inherent parallelizability.

The rapidly evolving landscape of SQIsign has hindered progress on critical research areas, such as the development of efficient implementations that keep up with the most recent developments, and exploring SQIsign's feasibility on resource-constrained platforms like the ARM Cortex-M4. Given SQIsign's primary focus on efficient verification for potentially resource-constrained devices, evaluating its practical deployment readiness has a paramount relevance.

Contributions

This chapter evaluates the performance of 1D SQIsign verification against 2D variants [34], by providing a state-of-the-art library for 1D verification with extended compatibility. Our main contributions are:

1. Several new optimizations for 1D SQIsign, including more efficient public keys and a streamlined hash to a random isogeny. Verification for compressed 1D SQIsign is on par with the 2D verification [34], while

uncompressed 1D SQIsign is roughly two times faster at the cost of an additional 163 bytes in signature size.

2. Variants of compressed and uncompressed SQIsign, allowing for a 5-core parallelization. We achieve the first sub-millisecond verification latency (1.3 Mcycles) at the cost of adding 128 bytes to the signature size for the uncompressed variant.

3. The first optimized 1D SQIsign verification library compatible with both 32-bit and 64-bit architectures, including state-of-the-art arithmetic from third parties, a new portable C option, optimized assembly code for the Cortex-M4, and integration with the pqm4 project [232], yielding the first SQIsign verification benchmarks on Cortex-M4.[1]

	Compressed	Parallel	Sig. (bytes)	Verif. (Mcc)	Accel.
NIST [101]	✓		177	34.28	1.00
2D-West [34]	✓		148	8.43	4.06
This work	✓		157	8.56	4.00
This work	✗		320	5.62	6.10
This work	✓	✓	285	2.05	16.72
This work	✗	✓	448	1.33	25.77

Table 14: Comparison of SQIsign variants at NIST Security Level I, in millions of Raptor Lake clock cycles, measured on an Intel i7-13700K server with TurboBoost and hyperthreading disabled.

Table 14 captures the main results of this chapter: the performance of our SQIsign verification versus previous works, including performance numbers for parallelized SQIsign. The last column measures the acceleration factor in comparison to the first row. All reported benchmarks use our custom C arithmetic, with Ice Lake intrinsics in the case of 2D-West, and Broadwell assembly code in all other cases. The 2D-West implementation does not incorporate Broadwell assembly arithmetic, but, as discussed in Section 5, its optimized arithmetic is found to be roughly comparable.

The library is available at:

https://github.com/Crypto-TII/the-sqisign-1d

1 Although our speed records are achieved on new parameters, our library also implements the current NIST parameters to be compatible with its test vectors.

ORGANIZATION. Section 2 provides a concise overview on the family of SQIsign variants. Section 3 introduces our improvements to 1D SQIsign verification. Section 4 discusses the features of our software library, and Section 5 presents a comprehensive performance evaluation together with concluding remarks.

§2 Preliminaries

Chapter II gives an introduction to curves and their isogenies, and a thorough introduction to SQIsign, including an introduction to the higher-dimensional variants. Here, we recall the most important details on verification and the primes used in SQIsign, before sketching the cost of one-dimensional versus two-dimensional verification.

Recall that we work with curves in Montgomery form, given by the affine equation

$$E : By^2 = x^3 + Ax^2 + x.$$

We mostly deal with Montgomery curve over $\mathbb{F}_{p^k}$, that is, we have constants $A, B \in \mathbb{F}_{p^k}$ such that $B(A^2 - 4) \neq 0$. For Montgomery curves, the j-invariant is $j(E) = 256(A^2 - 3)^3/(A^2 - 4)$, which is independent of B. Because of this, we describe a Montgomery curve using only its A coefficient, as E_A. A quadratic twist of E is a curve E^t that is isomorphic to E over $\mathbb{F}_{p^{2k}}$ but not over $\mathbb{F}_{p^k}$.

NOTATION. For an isogeny $\phi : E \to E'$, we denote the procedure to compute E' given only E and a description of $\ker(\phi)$ by xISOG. We denote the procedure to compute the image $\varphi(Q)$ of a point Q by xEVAL.

FIELD EXTENSIONS. From this point on, we only work with supersingular elliptic curves, and usually only up to isomorphism. Over characteristic p, the isomorphism class of any supersingular curve has a representative E that is defined over $\mathbb{F}_{p^2}$, whose p^2-power Frobenius map equals $[-p]$. The group structure of E and its quadratic twist over $\mathbb{F}_{p^{2k}}$ is fully characterized by

$$E\,(\mathbb{F}_{p^{2k}}) \cong \mathbb{Z}/(p^k - (-1)^k) \times \mathbb{Z}/(p^k - (-1)^k),$$
$$E^t(\mathbb{F}_{p^{2k}}) \cong \mathbb{Z}/(p^k + (-1)^k) \times \mathbb{Z}/(p^k + (-1)^k).$$

Observe that if n divides $p^k + 1$ or $p^k - 1$, then the n-torsion is fully defined over $\mathbb{F}_{p^{2k}}$, either on E or its twist.

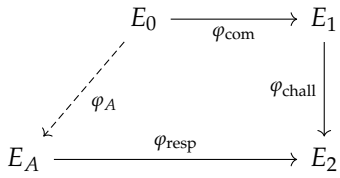


Figure 19: The SQIsign protocol as a diagram.

SQISIGN. We recall furthermore the description of SQIsign as an identification scheme, i.e., before the Fiat-Shamir transform is applied. We assume a starting elliptic curve E_0, whose endomorphism ring $\mathrm{End}(E_0)$ is public. Alice's secret key is an isogeny $\varphi_A : E_0 \to E_A$ that maps E_0 to her public key curve E_A. Knowledge of φ_A allows Alice to efficiently compute $\mathrm{End}(E_A)$. In the following Sigma-protocol, portrayed in Figure 19, she proves knowledge of this endomorphism ring.

COMMITMENT. Alice generates a random isogeny $\varphi_{\text{com}} : E_0 \to E_1$ and commits to E_1. The isogeny φ_{com} must remain secret, to prevent Bob from recovering $\mathrm{End}(E_A)$.

CHALLENGE. Bob samples a random cyclic isogeny $\varphi_{\text{chall}} : E_1 \to E_2$ and sends a description of φ_{chall} to Alice. As only Alice knows $\mathrm{End}(E_1)$, only she can compute $\mathrm{End}(E_2)$.

RESPONSE. Alice then has the right ingredients, namely $\mathrm{End}(E_A)$ and $\mathrm{End}(E_2)$, to compute a cyclic isogeny $\varphi_{\text{resp}} : E_A \to E_2$ using KLPT. She computes it such that $\hat{\varphi}_{\text{chall}} \circ \varphi_{\text{resp}}$ is cyclic and of a specific degree ℓ^k, and sends φ_{resp} to Bob.

VERIFICATION. Bob verifies that φ_{resp} is indeed an isogeny $E_A \to E_2$ of degree ℓ^k, and that $\hat{\varphi}_{\text{chall}} \circ \varphi_{\text{resp}}$ is cyclic.

It is shown in [148] that the Sigma-protocol is special sound with respect to the one-endomorphism relation

$$R = \{(E_A, w) \mid w \text{ is a non-scalar smooth-degree endomorphism of } E_A\}.$$

This proof requires that $\hat{\varphi}_{\text{chall}} \circ \varphi_{\text{resp}}$ is cyclic, which intuitively means that the 2-isogeny walk should not backtrack. Special soundness implies that the

protocol is a proof of knowledge for the hard relation R. To achieve honest-verifier zero-knowledge, we rely on the heuristic assumption that the KLPT variant proposed in [149] does not leak information about φ_A. Note that Alice cannot simply output the isogeny $\varphi_{\text{resp}} = \varphi_{\text{chall}} \circ \varphi_{\text{com}} \circ \hat{\varphi}_A$, since Bob could factor the response, and retrieve φ_A.

We then apply the Fiat-Shamir transform [178] to the aforementioned identification protocol: the challenge φ_{chall} is now computed from the result of a hash function H on inputs msg and E_1, and the signature σ is a description of the transcript $(E_1, \varphi_{\text{chall}}, \varphi_{\text{resp}})$.

One-dimensional SQIsign Verification.

There are several variants of one-dimensional verification, but on a high level they all use a common framework performing steps equivalent to the following:

1. Compute the codomain of $\varphi_{\text{resp}} : E_A \to E_2$.
2. Recompute the hash and compute the codomain of $\varphi_{\text{chall}} : E_1 \to E_2'$.
3. Check that E_2 and E_2' are isomorphic and that $\hat{\varphi}_{\text{chall}} \circ \varphi_{\text{resp}}$ is cyclic.

Step 1 is computationally the most expensive, since it involves the computation of the isogeny φ_{resp} of degree 2^e where $e \approx \lceil (15/4) \log_2 p \rceil$, whereas the degree 2^λ of the challenge isogeny in step 2 is solely determined by the security parameter λ.

Verification performance is primarily influenced by two factors: the choice of isogeny encoding, which offers a size-performance trade-off, and the form of the prime p. We begin by examining three isogeny encoding options.

UNCOMPRESSED SIGNATURES. A rational cyclic isogeny can be described simply by a point that generates its kernel. Hence, our hash function only needs to output a point of order 2^λ. However, the 2^e-isogeny φ_{resp} cannot be efficiently represented by a kernel generator, as such points would live in a large extension field. Instead, we describe φ_{resp} as a sequence of blocks $\varphi_i : E^{(i)} \to E^{(i+1)}$, for which each kernel $\ker \varphi_i$ can be described by a single point $K_i \in E^{(i)}(\mathbb{F}_{p^2})$ of some order which is a power of 2. The largest isogeny whose degree is a power of 2 we can describe by such a rational point K_i is given by the largest f such that $E[2^f] \subseteq E(\mathbb{F}_{p^2})$. For supersingular Montgomery curves, this is given by the largest f such that $2^f \mid p + 1$.

Using x-only arithmetic, a kernel point K can be described by a single $\mathbb{F}_{p^2}$ element. Likewise, a Montgomery curve can be represented only by its A

coefficient which is also an $\mathbb{F}_{p^2}$ element. Thus, the total size of the signature (accounting for a total of 4 kernel points and a curve) is 320 bytes. We refer to this as the *uncompressed* signature.

COMPRESSED SIGNATURES. As the previous naming suggests, we can decrease the signature size of σ using *compression*. Instead of a full description of $x(K)$ to define $\varphi : E \to E/\langle K \rangle$, we can exploit the fact that $E[d] \cong \mathbb{Z}_d \times \mathbb{Z}_d$ to write $K = [k_1]P + [k_2]Q$ for a deterministic basis P, Q and describe the isogeny only by the scalars k_1 and k_2. Furthermore, since multiples of K co-prime to d generate the same isogeny, we can always find an equivalent point of the form $K = P + [k]Q$ or $K = [k]P + Q$. Hence, an isogeny can be described by a single scalar $k \in \mathbb{Z}_d$ plus a single bit to specify one of the two forms, which brings the total size of the signature down to just 157 bytes. A detailed analysis of these steps is given in Section 4.

The size of the signature can be further decreased by including a compressed description for $\hat{\varphi}_{\text{chall}}$, which is enough to recover E_1, instead of including E_1 itself. This places an additional burden on the verifier, who now recovers E_1 from the image of $\hat{\varphi}_{\text{chall}}$ but also needs to compute the dual of $\hat{\varphi}_{\text{chall}}$ in order to compare it against the result of the hash. In practice, this is done by picking any point of order 2^λ not in the kernel of $\hat{\varphi}_{\text{chall}}$ and computing its image under $\hat{\varphi}_{\text{chall}}$, which is then a kernel generator for the dual of $\hat{\varphi}_{\text{chall}}$. The verifier must then make sure that this kernel generator is equivalent to the output of the hash, meaning that the two points must be scalar multiples of each other. In order to make this check faster in the verification, the aforementioned scalar multiple is computed by the signer and included as part of the signature, adding 128 bits, for a total of 173 bytes.

Despite the tempting improvements in signature size, compression has a high impact on performance since the basis (P, Q) must be computed at each curve $E^{(i)}$ in order to recover the kernel points K_i. This needs to be done through a deterministic method that the signer and verifier have previously agreed upon, and this dominates the cost of the verification in the original NIST proposal. Moreover, as basis sampling is not invariant under isomorphisms, signer and verifier must work on the exact same curve rather than only isomorphic curves. This leads to the concept of *curve normalization*, which the NIST proposal defines as a deterministic method for selecting one curve from an isomorphism class. This procedure is costly, requiring one inversion, one square root and several field multiplications, and was used on both E_1 and E_2.

SEEDED SIGNATURES. To improve the performance of the basis sampling, we revisited the idea of adding seeds to the signature in Chapter IX, as the P, Q bases are specified in the signature to avoid the cost of sampling them, but they are chosen from within a small pool of candidates so that they can be encoded in just one byte each with negligible failure probability. While the NIST proposal focused only on compressed signatures, this chapter analyzes and improves both compressed and uncompressed SQIsign signatures, as we are interested in the fastest verification time possible, possibly at the expense of larger signatures. We do not examine seeded signatures, as improvements in basis sampling have turned it into a negligible part of the cost.

Selecting Primes for Fast One-dimensional Verification.

The choice of prime has a significant impact on the verification, since having f as large as possible, with $p = c \cdot 2^f - 1$ allows us to compute the response isogeny in fewer blocks of rational 2^f-isogenies. The original SQIsign [100, 148, 149] could not simply maximize f: For the performance of their KLPT-based signing, the scheme required $p^2 - 1$ to also contain an odd smooth factor $T \mid (p^2 - 1)$ of size approximately $p^{5/4}$, in order for the signer to only need to work with rational isogenies. This condition restricts the search space and led to the prime p_{1973} with $f = 75$, which computes the response isogeny in 13 blocks.

As analyzed in Chapter IX, computing the signer's isogenies over larger extension fields relaxes the constraint $T \mid (p^2 - 1)$ and thus enables primes with larger f values. Taking advantage of this observation, we proposed to use the prime,

$$p_4 \;=\; 2^{246} \cdot 3 \cdot 67 - 1,$$

so that $f = 246$, where the response isogeny is computed in just 4 blocks of 2^{246}-isogenies. While the total degree e remains unchanged, fewer blocks benefits both compressed and uncompressed variants greatly: in uncompressed variants, it improves signature size by reducing the number of kernel points that need to be embedded in the signature, and in compressed variants, it improves on performance by requiring fewer basis samplings. On the downside, having to work over large extension fields would likely have a significant impact on signing performance, which was never measured in an optimized implementation.

A significant improvement was recently proposed by Onuki and Nakagawa [303], where instead of working over large extension fields the signer can use two-dimensional isogenies to produce a one-dimensional response isogeny as

output. Initial analysis suggests that this work enables practical signing times for primes with large 2^f-torsion, perhaps twice as fast as in classical SQIsign for primes with $f = 75$. This method no longer requires the prime to be related to T in any way, and even lifts the smoothness restriction from it, leaving a large f as the sole criterion for efficiency. In view of this, we no longer require the smoothness of p^k from p_4, and can instead use the prime from [34], which has the largest f possible for level-1-sized prime, $f = 248$, given by

$$p_{248} = 2^{248} \cdot 5 - 1.$$

§3 Improving One-Dimensional SQIsign Verification

In this section, we describe a series of algorithmic improvements and modifications to the signature format that allow us to speed up one-dimensional verification beyond the improvements from Chapter IX. We aim to make 1D verification as fast as possible and make no assumptions regarding the algorithmic strategy of the signer, other than the fact that it is able to produce a one-dimensional response isogeny which fits in 4 blocks of rational isogenies for p_{248}. For concrete resuts, we focus on the prime p_{248} which achieves NIST Security Level I, although the techniques generalize straightforwardly to other primes and security levels.

We explore both compressed and uncompressed variants, as the trade-off between signature size and speed may be worthwhile in several applications. In addition, we present versions of both variants that allow for a near-perfect parallelization of the verification with 5 cores, pushing performance to its limits at the cost of a reasonable increase in signature size.

3.1 Improved compressed signature

We first describe an enhanced version of a compressed SQIsign signature which significantly reduces the performance cost of using point compression by exploiting the *smart sampling* technique introduced in Section IX.5.1, as well as other newly developed techniques that build up on it. We begin by refining Theorem IX.4.

Lemma 1. Let E_α be a supersingular curve given by $y^2 = x(x-\alpha)(x-1/\alpha)$ for some $\alpha \in \mathbb{F}_{p^2}$ and let $P = (x, y) \in E_\alpha(\mathbb{F}_{p^2})$. Let 2^f denote the maximal rational

two-power torsion of E_α. Let b_0 and b_1 be the quadratic characters of x and $x - \alpha$, respectively. Then

$$P \in [2]E \quad \Leftrightarrow \quad b_0 = b_1 = 1.$$

In particular, the order of P is a multiple of 2^f if and only if b_0 and b_1 are not both 1. Moreover, if this is the case, b_0 and b_1 precisely determine above with order-2 point P lays:

$$\begin{aligned} P \text{ is above } & (0,0) \quad \Leftrightarrow \quad b_0 = \ \ 1, \quad b_1 = -1, \\ P \text{ is above } & (\alpha,0) \quad \Leftrightarrow \quad b_0 = -1, \quad b_1 = \ \ 1, \\ P \text{ is above } & (1/\alpha,0) \quad \Leftrightarrow \quad b_0 = -1, \quad b_1 = -1. \end{aligned}$$

Corollary 2. In the setting of the above lemma, if $P = (x, y) \in E_\alpha(\mathbb{F}_{p^2})$ with x non-square, then the order of P is divisble by 2^f and P does not lie above $(0,0)$. On the other hand, if $P = (x, y) \in E_\alpha(\mathbb{F}_{p^2})$ where $x = z\alpha$ with z a square but $z - 1$ a non-square, then the order of P is divisible by 2^f and P lies above $(0,0)$.

Proof. The first case is a direct consequence of Lemma 1. For the second case, note that α is always a square [15, 125] and so $x = z\alpha$ is a square but $x - \alpha = (z - 1)\alpha$ is not. □

By precomputing a list of non-squares x_i as well as a list of field elements z_i where z_i is a square but $z_i - 1$ is not, Corollary 2 allows us to reduce the cost of sampling a basis in any curve given its α coefficient, as shown in Algorithm 20. Note that Algorithm 20 returns an implicit basis for $E_\alpha[2^f]$ which can be made explicit by multiplying both P and Q by $[\frac{p+1}{2^f}]$ if necessary.

Algorithm 20 SmartSampling (adapted from Chapter IX)

Input: $\alpha \in \mathbb{F}_{p^2}$ and lists $X, Z \subset \mathbb{F}_{p^2}$ with $Z[i]$ a square and $X[i]$, $Z[i] - 1$ non-squares.
Output: An implicit x-only basis P, Q of $E_\alpha[2^f]$ where Q is above $(0,0)$.

```
for z ∈ Z do
    x ← zα
    if (x, _) ∈ E_α(F_{p^2}) then
        Q ← (x, _)
        break
for x ∈ X do
    if (x, _) ∈ E_α(F_{p^2}) then
        P ← (x, _)
        return (P, Q)
```

As described in more detail in Chapter IX, the cost of Algorithm 20 is dominated by an average of just four quadratic character computations to check if a point is on the curve. In contrast, the traditional basis sampling method used in the NIST proposal requires the same quadratic character computations but also an expensive chain of f point doublings at every execution of each loop to check the order of the point. The cost of basis sampling thus changes from being the dominating cost of verification to just a negligible overhead. The main drawback encountered in Chapter IX to use Algorithm 20 is the requirement of the α coefficient E_A. Obtaining α from the Montgomery coefficient A requires a costly square root computation, which has be performed at every block. We describe a novel technique that bypass this issue, effectively enabling Algorithm 20 at much higher efficiency than before.

SMART PUBLIC KEYS. In order to avoid computing α for the public-key curve, we use α itself as the public key rather than the Montgomery A coefficient. This has no effect on the public key size, since both are $\mathbb{F}_{p^2}$ elements, and recovering $A = -\alpha - 1/\alpha$ can be done projectively at a minimal cost.

SLIGHTLY SHORTER ISOGENIES. When computing an isogeny of degree 2^f and pushing the kernel point through each 2-isogeny, we are left with a point of order 2 different from $(0, 0)$ in the very last step. Thus, this point is either $(\alpha, 0)$ or $(1/alpha, 0)$ for this curve, and so we obtain α for free for this curve. Therefore, it is significantly cheaper to use this point to sample a new basis rather than completing the last step of the isogeny. This means that both signer and verifier must agree to use blocks of 2^{f-1}-isogenies instead of using the entire 2^f torsion. Losing a bit of the 2^f torsion does not present a drawback, as we can still compute the response isogeny using four 2^{f-1}-isogeny blocks whenever f is close enoguh to $\log p$, as is the case for p_{248}.

SMART HASHING. The NIST proposal samples a basis (P, Q) for $E[2^f]$ on the commitment curve, to compute the challenge isogeny with kernel generated by $K = P + [k]Q$ where k is the output of a pre-defined hash H. We instead use a novel method based on Corollary 2 to hash directly to a kernel point K, thus removing the need to sample for a basis (P, Q) and the need to compute $P + [k]Q$ using a three-point ladder. By using rejection sampling to hash to a non-square x-values, we are guaranteed to obtain a point whose order is divible by 2^f, which does not lie above $(0, 0)$. After multiplying out the cofactor, we obtain a point of order 2^λ that can be used as the kernel point. Even though the output is never above $(0, 0)$, we still have a large enough challenge space, since

the number of 2^λ-isogenies whose kernel doesn't contain $(0,0)$ is precisely 2^λ. Using rejection sampling means that the hash is variable time, yet still constant time, as this has no bearing since its inputs msg and cmt are public. The approach is summarized in Algorithm 21. By using smart public keys, shorter isogenies and smart hashing, the α coefficients are always readily available and we essentially remove the cost of basis sampling completely.

Algorithm 21 SmartHashing

Input: A curve E, a message msgand a fixed non-square $\delta \in \mathbb{F}_{p^2}$.
Output: An x-only point $K \in E[2^\lambda]$ not over $(0,0)$

```
ctr ← 0
while 1 do
    x ← H(E|| msg ||ctr)
    x ← δ · x^2
    if (x, _) ∈ E(F_{p^2}) then
        K ← (x, _)
        break
    ctr ← ctr + 1
return [(p + 1)/2^λ]K
```

UNNORMALIZED CHALLENGE CURVE. Curve normalization is another costly procedure that is used in verification, which requires a square root and an inversion. While normalizing the commitment curve is essential in order to recover the same challenge isogeny from the hash, we find, contrary to the NIST proposal, that there is no need to normalize the commitment curve. This was presumed necessary in order to compress the kernel of $\hat{\varphi}_{\text{chall}}$, but instead of using the normalized version of E_2 both parties can agree to use any version of E_2 that results from evaluating $\varphi_{\text{resp}} : E_A \to E_2$ with the common formulas.

Using this unnormalized version of E_2 means that the kernel of $\hat{\varphi}_{\text{resp}}$ is above $(0,0)$, so the cyclicity check is reduced to ensuring that the kernel generator for $\hat{\varphi}_{\text{chall}}$ is of the form $P + kQ$ and not $kP + Q$[2]. Thus, apart from removing the cost of a curve normalization, this trick removes the cost of the cyclicity check and the need for an extra byte in the signature.

Remark 1. We could easily obtain a seeded variant of this signature by including the indices in the lists X, Z from Algorithm 20 as seeds. However, the smart

2 Recall that our SmartSampling always returns the basis point Q above $(0,0)$.

sampling is already so efficient that this did not improve the verification performance significantly. Thus, we omit this variant here.

3.2 COMPUTATIONAL COST OF UNCOMPRESSED SIGNATURES

The uncompressed variant proposed in AprèsSQI removed most of the cost of the verification that is not directly due to isogeny computations: **a)** kernel points for $\varphi_{\text{resp}} : E_A \to E_2$ are included in the signature explicitly so that basis samplings, point differences and 3-point ladders are not needed, and **b)** the commitment curve E_1 is included in the signature, which avoids the need to normalize it and allows the verifier to compute $\varphi_{\text{chall}} : E_1 \to E_2'$ in the natural way, without having to compute a generator for the dual. The verifier then confirms that E_2 and E_2' have the same j-invariant, without needing to normalize either. Moreover, the cyclicity check can be performed by recording the second-to-last j-invariant along the way of both isogenies, at a minimal cost, and confirming that they are different.

Previously, the most significant non-isogeny overhead was recovering φ_{chall} from the hash, which still required a basis sampling, point-difference computation, and a three-point ladder. However, by using SmartHashing, we eliminate this overhead entirely, replacing it with an average of just two efficiently optimized quadratic residue computations (see Section 4.3). Hence, the total cost of verification for uncompressed signatures has only negligible overhead over the core cost of one 2^λ-isogeny and four 2^f-isogenies.

3.3 PARALLELIZATION-FRIENDLY SIGNATURES

The 1D verification of SQIsign, which evaluates multiple blocks of 2^f-isogenies using identical instructions, presents a natural opportunity for parallelization. By including the intermediate curves $E^{(i)}$ of the response isogeny $\varphi_{\text{resp}} : E_A \to E^{(1)} \to E^{(2)} \to E^{(3)} \to E_2$, each block $E^{(i)} \to E^{(i+1)'}$ can be computed simultaneously and the verifier only needs to check that the j-invariants of $E^{(i+1)'}$ and $E^{(i+1)}$ match at each step. This can be taken further by using an additional core to also compute the challenge isogeny (or its dual) in parallel. We present simple implementations of this idea using OpenMP, for both the smart-compressed and uncompressed variants.

A naïve implementation of the idea using p_{248} would require including 4 more curves in the signature, adding a hefty 256 B. However, by inverting the directions in which some isogenies are computed and providing kernels for

the duals instead, we only need to include every other curve in the signature. Using the arrangement shown in Figure 20, the overhead in signature size is reduced by a factor two.

Figure 20: Direction in which isogenies in the parallel verification are evaluated for the compressed version (top) and uncompressed version (bottom).

Compressed:

$$E_A \xrightarrow[\varphi_1]{} E^{(1)} \xleftarrow[\hat{\varphi}_2]{} E^{(2)} \xrightarrow[\varphi_3]{} E^{(3)} \xleftarrow[\hat{\varphi}_4]{} E_2 \xrightarrow[\hat{\varphi}_{\text{chall}}]{} E_1$$

Uncompressed:

$$E_A \xleftarrow[\hat{\varphi}_1]{} E^{(1)} \xrightarrow[\varphi_2]{} E^{(2)} \xleftarrow[\hat{\varphi}_3]{} E^{(3)} \xrightarrow[\varphi_4]{} E_2 \xleftarrow[\varphi_{\text{chall}}]{} E_1$$

The two variants compute their isogenies in opposite directions as a consequence of staying true to their original objectives: in the compressed version, we only need to include two curves in the signature ($E^{(2)}$ and E_2) which minimizes its size, but verifying the hash still requires computing a dual for $\hat{\varphi}_{\text{chall}}$ and normalizing E_1. On the other hand, the uncompressed version computes $\varphi_{\text{chall}} : E_1 \to E_2$ in the natural way and makes essentially only isogeny computations, but has slighly bigger overhead in signature size since it needs to include three curves: $E^{(1)}$, $E^{(3)}$ and E_1.

Although not strictly necessary, our implementation requires the signer to present the isogenies in a way that none of the kernels include the point $(0, 0)$. This is easy to do by applying an inexpensive isomorphism to the intermediate curves and kernel points when needed, and avoids having to add an additional bit per step in the compressed version.

Remark 2. Future work may also consider parallelization for 2D SQIsign. This seems to encounter some roadblocks that are more easily resolved in a 1D variant: the 2D isogeny is performed in a single block, which, if split into smaller blocks requires including information on theta structures efficiently. On the other hand, the core underlying arithmetic building blocks could potentially be efficiently vectorized [39]. Remarkably, we will show in Chapter XI that such an improvement may again improve 1D verification too.

Remark 3. The computation for the 2^λ-isogeny can be embedded into that of a larger 2^f-isogeny that performs dummy calculations for the additional steps. Combined with the fact that we do not need to worry about special

cases for isogenies with $(0,0)$ in the kernel, this means that the entirety of the parallel uncompressed variant verification is compliant with a SIMD (single instruction, multiple data) paradigm and would fit a vectorized implementation nicely. The compressed version could also be largely vectorized, but faces three technicalities. First, we must push a point through the challenge isogeny to get the dual, while the others do not. Second, we need to perform a variable-time basis sampling for each step. And third, the isogeny φ_1 starts from E_A, which we are not free to map through an isomorphism as it is the public key, so it still needs a branch in case $(0,0)$ is in the kernel.

§4 A New Library for One-Dimensional Verification

In this section, we describe our new SQIsign library. It is a modification of the NIST submission's reference library[3] to include our improvements. Namely, we ported this implementation to 32-bit architectures and optimized the verification of one-dimensional SQIsign variants using the improvements from Chapter IX and Section 3.

The reference library implements SQIsign for NIST Security Levels I, III and V. This implementation is comprised of several modules and has GMP as a dependency. The finite-field arithmetic module offers two options: a reference C implementation for all security levels and an optimized assembly implementation for p_{248}. The reference implementation is obtained with the Fiat Cryptography [167] code generator.

To add support for Cortex-M4, we create a 32-bit version of the reference library, replacing the dependency on GMP with mini-GMP, and adding an option to enable verification only. We optimize verification by improving finite-field arithmetic and implementing the techniques discussed in Section 3. The improved finite-field arithmetic is discussed in Section 4.2 and incorporates a more efficient portable implementation and an optimized ARM assembly implementation. In Section 4.3, we improve computations of quadratic characters, inverses and square roots.

In addition to implementing the optimized verification variants for p_{248}, our library also implements the entire protocol for the original parameter sets, which benefit from arithmetic and algorithmic improvements where possible, although limited to retain compatibility with the original test vectors. Although the library does not implement signing for p_{248}, it includes valid test vectors

3 Available at https://github.com/SQISign/the-sqisign.

that have been generating using a fork of the AprèsSQI Sage implementation that we modified to fit the new signature formats.

4.1 CORTEX-M4 COMPATIBILITY

We modify the SQIsign implementation submitted to NIST to make it compatible with embedded devices, in particular the ARM Cortex-M4 core, based on the 32-bit ARMv7-M architecture. Such devices stand to benefit from the short signature sizes of SQIsign, but their limited computing power, due to slower clock speeds, unavailability of 64-bit arithmetic instructions and lack of superscalar execution capabilities, make it challenging to run SQIsign operations at an acceptable speed. The improvements described in this subsection bridge much of this gap, although we additionally invest some effort in speeding up the $\mathbb{F}_p$ arithmetic implementation to get closer to practical runtimes on this platform.

A first roadblock is that, while the original SQIsign implementation did consider 32-bit architectures, it does not actually compile and run in 32-bit mode. We started by rerunning the Fiat-Crypto code generator [167] to output radix-2^{32} arithmetic, as the original radix-2^{64} code requires 128-bit integer types which are not available on 32-bit architectures. We also fix several bugs in routines that access individual bits of a multiprecision integer but implicitly assume 64-bit words.

Remark 4. Our library contains a *full* 32-bit port of SQIsign, including key generation and signing, as we changed the intbig module, which was hardcoded to 64 bits, and including an option to replace the heavyweight GMP library dependency with the much lighter mini-GMP library. However, further work remains to be done for key generation and signing to work on embedded devices; other than the obvious performance issues, the main roadblocks are a very large memory footprint and widespread use of dynamic memory allocation.

Given Remark 4, we implement a verification-only build option, removing unnecessary dependencies related to the key generation- and signing-specific modules of SQIsign, such as intbig, klpt and quaternion. This considerably reduces the number of source files that need to be included in a verification-only build. Additionally, some arrays used to store kernel-point data, whose size grows with the square root of the largest prime degree used for isogenies in signing, are reduced for the verification which uses degree at most 3, significantly reducing the memory footprint. As a result, verification for all SQIsign

security levels now fits in devices with as little as 32 KB of RAM, or even 8 KB in some cases.

A characteristic of embedded systems is that they lack virtual memory features found on larger systems. This, combined with the small amount of RAM available (in the range of tens to hundreds KB), means that using dynamic memory allocation, such as the C standard library's $\mathrm{malloc}()$ function, is not advised, and indeed strictly forbidden by certain coding standards targeting embedded systems. Thus, we remove a few dynamic memory allocations from the verification-only build. As per Remark 4, much more work remains to be done to remove such allocations from the key generation and signing operations, particularly in the intbig module, due to its use of the GMP library.

In order to benchmark SQIsign verification performance on the Cortex-M4, we have integrated our library with pqm4 [232]. This presented yet another roadblock, as pqm4 assumes that all operations (key generation, signing and verification) are available. To work around this, we generated fixed key pairs and signatures, and replaced the NIST API functions $\mathrm{crypto\_keypair}$ and $\mathrm{crypto\_sign}$ with mock implementations that just return these fixed key pairs and signature. Verification, of course, is computed normally.

4.2 OPTIMIZING FINITE-FIELD ARITHMETIC

As isogeny computations comprise most of the execution time of verification, and ultimately perform arithmetic in $\mathbb{F}_p$, it is clear that faster $\mathbb{F}_p$ arithmetic directly translates into improved verification performance. Thus, we have investigated techniques to speed up portable $\mathbb{F}_p$ arithmetic, as well as arithmetic on the ARMv7-M architecture, more specifically the Cortex-M4 core.

Remark 5. The SQIsign implementation submitted to NIST also includes an x86-64 assembly implementation of $\mathbb{F}_p$ arithmetic for the level 1 prime p_{1973}, targeting the Intel Broadwell (5th generation Core) microarchitecture. We have adapted this implementation to p_{248}, but note that it could be further improved by considering the larger power-of-two cofactor 2^{248} of $p_{248} + 1$.

Alternative Portable C Arithmetic.

Recently, Scott [345] introduced a new code generator using quite different design choices compared to Fiat-Crypto. In particular, Fiat-Crypto employs a *saturated* representation, in which a multiprecision integer is split into *limbs* with size matching the CPU word length, i.e. radix 2^{32} or 2^{64} for a 32- or 64-bit CPU, respectively, whereas Scott's generator opts for an *unsaturated*

representation, using radix 2^r with $r < 32$ (for 32-bit CPUs) or $r < 64$ (for 64-bit CPUs). We implement the required changes in our library to support Scott's code generator and performed benchmarks on both Intel and the Cortex-M4. For Intel, we notice that Scott's generator performs consistently at the protocol level whether gcc or clang compilers are used, while Fiat-Crypto performs much better with clang compared to gcc, and indeed generally outperforms Scott's code generator on clang. For the Cortex-M4, to the best of our knowledge, pqm4 is only compatible with gcc and, in our experience, gcc generally produces faster code than clang for this platform. As such, we saw significant speedups from the use of Scott's generator in the Cortex-M4, as we show in Section 5.

Optimized ARM Assembly Arithmetic.

While Intel high-performance CPUs feature superscalar and out-of-order execution, cores based on the ARMv7-M architecture are targeted at microcontroller units in embedded applications, with tight constraints on cost (i.e., silicon area) and power consumption. Meeting these constraints requires much simpler single-issue in-order cores, such as the widely-used Cortex-M4.

We were graciously given access to a multiprecision arithmetic code generator targeting the Cortex-M4.[4] The generator outputs C files with an API compatible with Scott's code generator [345]. We use this generator to produce code for all primes found in the original SQIsign submission, as well as p_{248}. We tweak the implementation of modular reduction for p_{248} from the originally generated code: the high degree of Montgomery-friendliness of this prime allows us to write an especially compact and performant implementation.

4.3 OTHER OPTIMIZED FIELD OPERATIONS

For the remaining operations in $\mathbb{F}_p$, SQIsign originally implements exponentiation to $(p-3)$ in constant time through an addition chain for primes of form $p = 3 \bmod 4$. The result is then reused to compute inversions, Legendre symbols and square roots in $\mathbb{F}_p$ by adjusting the result to the correct powers (respectively $p-2$, $\frac{p-1}{2}$, $\frac{p+1}{4}$). We extended the library to implement faster alternatives. For inversion, we adapted the Montgomery version [214] of the constant-time Bernstein-Yang (BY) algorithm [50], such that it can be reused for signing. For Legendre symbols, we implemented a variable-time version of Pornin's algorithm [315] ported from the Rust version available in [370].

4 Manuscript under preparation by its authors.

Pornin's algorithm is actually simpler and faster in variable-time than a BY-based variant [207], without introducing the concerns with intellectual property of the latter.

In SQIsign's NIST submission source code, inverting in $\mathbb{F}_{p^2}$ amounts to computing an inversion, two multiplications, two squarings, and a few additions in $\mathbb{F}_p$. Quadratic character computations in $\mathbb{F}_{p^2}$ require two squarings, one addition, and a Legendre symbol in $\mathbb{F}_p$. The computation of square roots in $\mathbb{F}_{p^2}$ requires two square roots and two inversions in $\mathbb{F}_p$ (although one of them consists of just inverting 2, which can be precomputed), for a total of four exponentiations. Similar to Chapter IX, we implement Scott's fast square root method [342], reducing this to just two exponentiation computations in $\mathbb{F}_p$, without any precomputation, through a novel reformulation reported in Algorithm 22, applicable to primes of the form $p = 3 \bmod 4$.

Algorithm 22 Deterministic square roots in $\mathbb{F}_{p^2}$, for $p \equiv 3 \bmod 4$

Input: A quadratic residue $a = a_0 + ia_1 \in \mathbb{F}_{p^2}$, with $i^2 = -1$
Output: A deterministic $x \in \mathbb{F}_{p^2}$ such that $x^2 = a$

1: $\delta \leftarrow (a_0^2 + a_1^2)^{(p+1)/4}$
2: $x_0 \leftarrow a_0 + \delta$
3: $t_0 \leftarrow 2x_0$
4: $x_1 \leftarrow t_0^{(p-3)/4}$
5: $x_0 \leftarrow x_0 x_1$
6: $x_1 \leftarrow a_1 x_1$
7: $t_1 \leftarrow (2x_0)^2$
8: **if** $t_1 = t_0$ **then**
9: **return** $x_0 + ix_1$
10: **else**
11: **return** $x_1 - ix_0$

§5 Results and Conclusions

PQM4 RESULTS. We display Cortex-M4 results in Table 15, benchmarked using the pqm4 framework [232] on ST Microelectronics' NUCLEO-L4R5ZI board. The compiler is gcc 13.2.1. As usual, the core runs at a reduced clock (24 MHz) to avoid the use of wait states for Flash memory.

To isolate the effect of our algorithmic improvements, we prepared a version of the library which makes minimal changes with respect to the NIST implementation, only to make it work in 32 bits but without any further improvements. With respect to this baseline for verification, our algorithmic improvements achieve speedups of 3.88×, 5.84× and 8.52× for NIST Security Levels I, III and

Table 15: Cortex-M4 performance, code size and RAM usage results. Sizes reported in kilobytes. Speedup (Acc.) relative to the performance of the NIST submission [101], denoted Ref., at the same security level for verification.

Sec. level	Prime	Impl.	$\mathbb{F}_p$ arith.	Verif. (Mcyc.)	Acc.	Code size	RAM usage
I	p_{1973}	Ref.	Fiat	1849.0	1.0×	229.0	12.8
		Ours	Fiat	1176.0	1.6×	111.0	8.7
			C	757.7	2.4×	85.8	9.1
			ARM	477.1	3.9×	89.7	8.6
I	p_{248}	Ours, smart	Fiat	391.3	4.7×	86.1	4.5
			C	170.7	10.8×	60.7	5.0
			ARM	123.4	15.0×	62.4	4.4
		Ours, uncomp.	Fiat	260.3	7.1×	86.4	5.0
			C	114.2	16.2×	61.0	5.4
			ARM	82.3	22.5×	62.8	4.9
III	p_{47441}	Ref.	Fiat	9948.0	1.0×	270.0	18.3
		Ours	Fiat	4600.0	2.2×	150.0	12.8
			C	2670.0	3.7×	106.0	14.0
			ARM	1704.0	5.8×	104.0	12.9
V	p_{318233}	Ref.	Fiat	31204.0	1.0×	326.0	24.0
		Ours	Fiat	11230.0	2.8×	202.0	16.8
			C	5387.0	5.8×	129.0	17.9
			ARM	3660.0	8.5×	127.0	16.7

V, respectively, compared to the NIST submission ported to 32-bit architectures. For the new parameter set, SmartSampling further improves this to 15× for level 1, combined with a reduction in signature size. Trading off signature size for improved performance, a speedup of 22.5× is achievable for uncompressed SQIsign. Scaling to the CPU's nominal 120 MHz clock speed[5], we reach execution times of ≈ 1 second and ≈ 0.7 seconds for smart and uncompressed variants, respectively.

5 Some performance losses are expected, as Flash memory wait states are required for clock speeds above 30 MHz, although most of it should be mitigated by ST's ART accelerator Flash caching subsystem.

Our implementation also reduces code size significantly, mostly by not compiling unnecessary code as a result of the verification-only build option. The improved $\mathbb{F}_p$ arithmetic implementations further reduce code size significantly, in addition to large performance benefits. The RAM usage figures for the baseline implementation incorporate the memory optimizations discussed in Section 4.1.[6] Our library achieves a welcome reduction in RAM usage when using the same Fiat Cryptography code generator as the NIST submission, and the M4-optimized $\mathbb{F}_p$ arithmetic has a negligible effect in this metric. We highlight the figures for the smart sampling variant (62.4 KB of code, 4.45 KB of RAM), which are eminently practical for many applications.

ONE-DIMENSIONAL VERIFICATION IN HIGH-END PROCESSORS. We have furthermore conducted benchmarks on high-performance processors which are directly comparable to previous results, as shown in **??** in the introduction of this chapter. We compare our library with the SQIsign NIST submission implementation, as well as the 2-dimensional implementation of SQIsign 2D-West[7] [34]. All libraries use the same reference implementation for finite-field arithmetic (Fiat Cryptography). The optimized finite-field arithmetic for SQIsign 2D-West incorporates Ice Lake intrinsics, while for NIST SQIsign and our library, the optimized version is assembly code targeting the Broadwell architecture. While different in nature, both are architecture-specific and our own micro-benchmarks show that their performances at the field-arithmetic level are roughly equal, hence Table 14 offers a roughly fair comparison. All code was compiled on the same Linux-based system using clang 18.3.1 with flags `-march=native -O3` and linking dynamically with GMP (ver. 6.3). The benchmarks were executed on a 13th Gen Intel Core i7-13700K (Raptor Lake) processor with TurboBoost and hyperthreading disabled.

With almost the entirety of the running time dedicated to the 4-isogeny chains in our uncompressed variant, we expect improvements mostly in finite-field arithmetic or in an algorithmic breakthrough that could reduce the degree of the response isogeny. As things stand, we believe our results are a solid approximation of the fastest possible sequential 1D verification for response isogenies of the current length.

6 Prior to these optimizations, some of the security levels did not even fit in our target development board; level V in particular required close to a megabyte of RAM.

7 At time of writing, the source code is not publicly available, but the authors provided us with access to it and permission to benchmark it.

CONCLUDING REMARKS. Overall, our results suggest that 1D SQIsign verification can be fully competitive with 2D verification, with even our most size-efficient variant performing on par with 2D-West [34] and the uncompressed variant providing close to a factor two speed-up. On top of that, the inherent parallelizability of 1D verification allows it to be up to more than five times faster at a modest increase in signature size. We have exploited the flexibility in trade-offs that 1D enjoys to report record-breaking performance for SQIsign verification and, for the first time, to bring SQIsign to the realm of embedded devices in practice.

We stress that our analysis focuses solely on verification and we must concede the current superiority of 2D variants in terms of signing performance. Nevertheless, results such as [303], exploiting 2D techniques in signing while keeping the verification 1D, have the potential to put the spotlight back on 1D verification, and we hope our encouraging results will motivate further research in this direction.

XI

RETURN OF THE KUMMER: A TOOLBOX FOR GENUS-2 CRYPTOGRAPHY

This work expands the machinery we have for isogeny-based cryptography in genus 2 by developing a toolbox of several essential algorithms for *Kummer surfaces*, the dimension-2 analogue of x-only arithmetic on elliptic curves. Kummer surfaces have been suggested in hyper-elliptic curve cryptography since at least the 1980s and recently these surfaces have reappeared to efficiently compute $(2,2)$-isogenies. We construct several essential analogues of techniques used in one-dimensional isogeny-based cryptography, such as pairings, deterministic point sampling and point compression and give an overview of $(2,2)$-isogenies on Kummer surfaces. We furthermore show how Scholten's construction can be used to transform isogeny-based cryptography over elliptic curves over $\mathbb{F}_{p^2}$ into protocols over Kummer surfaces over $\mathbb{F}_p$.

As an example of this approach, we demonstrate that SQIsign verification can be performed completely on Kummer surfaces, and, therefore, that one-dimensional SQIsign verification can be viewed as a two-dimensional isogeny between products of elliptic curves. Curiously, the isogeny is then defined over $\mathbb{F}_p$ rather than $\mathbb{F}_{p^2}$. Contrary to expectation, the cost of SQIsign verification using Kummer surfaces does not explode: verification costs only $1.5\times$ more in terms of finite-field operations. Furthermore, it is plausible that arithmetic on Kummer surfaces can be efficiently vectorised, giving Kummer-based proto-

This chapter is based on the paper

> Maria Corte-Real Santos and Krijn Reijnders. *Return of the Kummer: a Toolbox for Genus-2 Cryptography*. Cryptology ePrint Archive, Paper 2024/948. 2024. URL: https://eprint.iacr.org/2024/948.

I have only made small editorial changes to ensure consistency and improve clarity. This work was developed simultaneously with Chapter X and finished slightly earlier. This chapter therefore does not incorporate those results.

cols over $\mathbb{F}_p$ a potential performance boost on modern architectures, possibly surpassing the performance of elliptic-curve analogues over $\mathbb{F}_{p^2}$.

§1 Introduction

Post-quantum cryptography aims to develop cryptographic primitives that are secure when the adversary has access to a classical and quantum computer. Due to the growing investment into quantum computing, this field has garnered a significant amount of attention in the last decade, culminating in the NIST standardisation of the key encapsulation mechanism Kyber [62] and the digital signature schemes Dilithium [164], Falcon [180], and SPHICNS+ [47]. Nevertheless, post-quantum signatures still deserve more attention: we rely mostly on lattice-based security assumptions, and the signature sizes are significantly larger than pre-quanum signatures. Due to this, NIST is still actively seeking new post-quantum secure signature schemes [357]. Isogeny-based cryptography offers an answer to these problems. SQIsign [100, 148, 149] relies on the hardness of the general isogeny problem and boasts the smallest combined signature and public-key size of any signature scheme submitted to NIST's Call for Additional Digital Signature Schemes.

The main disadvantage of isogeny-based primitives is their speed. The signing operation in SQIsign and variants is a few orders of magnitude slower than the lattice-based alternatives, and verification requires at least a few milliseconds. Therefore, there has been a surge of recent research that aims to improve the efficiency of SQIsign. We highlight two schemes in particular. First, SQIsignHD [141], a new scheme that offers much more competitive signing times and improved security reductions, at the cost of verification speed. Second, AprèsSQI, as presented in Chapter IX, a variant of SQIsign optimised for verification speed, with additional trade-offs between verification time and signature size.

Very recent works [34, 166, 288] have shown incredible advancements in the performance of higher-dimensional SQIsign, achieving relatively fast signing and verification in a few milliseconds. These approaches verify using a 2-dimensional $(2^n, 2^n)$-isogeny over $\mathbb{F}_{p^2}$ between products of elliptic curves, using the machinery developed after the SIDH attacks [86, 266, 329] and in particular using fast isogeny formulas derived from theta structures [142].

Contributions

In this work, we make a step towards having a full-fledged toolbox for isogeny-based cryptography in genus 2. We give an overview of Kummer surfaces, including improvements to crucial maps between different Kummer surface *models*, and give a concrete exposition of degree-2 pairings and isogenies of Kummer surfaces. We then show how to apply these in the context of isogeny-based cryptography by developing several algorithms including CheckOrigin, PointDifference and PointCompression, whose analogues on elliptic curves have existed for years.

We further describe how to exploit Scholten's construction [338] to associate a dimension-2 Kummer surface over $\mathbb{F}_p$ to any elliptic curve over $\mathbb{F}_{p^2}$ that has rational 2-torsion. We also detail the extension of this construction due to Costello [125], which depicts how 2-isogenies $\phi : E_1 \to E_2$ between elliptic curves can be associated to $(2,2)$-isogenies $\varphi : \mathcal{K}_1 \to \mathcal{K}_2$ between the corresponding Kummer surfaces. In this way, isogeny-based primitives using supersingular elliptic curves defined over $\mathbb{F}_{p^2}$ can be modified to work with *superspecial* Kummer surfaces instead, with all computations now over $\mathbb{F}_p$. Additionally, by restricting to Kummer surfaces that arise from level-2 theta structures, we can potentially exploit vectorisation for the core Kummer arithmetic [39].

We show that these special $(2,2)$-isogenies between Scholten Kummer surfaces, first described by Costello [125], are a natural restriction of the general $(2,2)$-isogenies defined for these objects. Compared to the $(2,2)$-isogenies derived from theta structures [142], in this work, we use the more geometric interpretation from Costello [125], which views such $(2,2)$-isogenies as morphisms between Jacobians of hyperelliptic curves, and in particular their Kummer surfaces. Fundamentally, our approach relies on theta structures too, but the geometric interpretation using hyperelliptic curves allows us to more naturally develop similar techniques as those used for elliptic curves.

As a showcase of these tools and techniques, we show that the original SQIsign verification [100, 148, 149], or the AprèsSQI variant from Chapter IX, can also be performed completely over Kummer surfaces defined over $\mathbb{F}_p$. Using Scholten's construction [338], this turns the one-dimensional response isogeny into a two-dimensional isogeny between products of elliptic curves. To achieve SQIsign verification on Kummer surfaces[1], we require the 2-pairings,

1 We choose to do SQIsign on Kummer surfaces "not because it is easy, but because it is hard; because that goal will serve to organise and measure the best of our energies and skills".

new algorithms, and optimised isogeny formulas, we develop in this chapter. In more detail, we do the following.

- Section 2 describes general Kummer surfaces and squared Kummer surfaces, including those that arise from Scholten's construction and their twists. We explicitly construct maps between these, if they exist, and give improved maps between the squared Kummer and the Jacobian. We also categorise elliptic Kummer surfaces.
- Section 3 describes the use of the generalised Tate pairing to describe the image of isogenies, which allows us to generalise Theorem IX.4 to the dimension-2 (and higher) case in Theorem 11.
- Section 4 develops essential tools for cryptography on Kummer surfaces, namely CheckOrigin and PointDifference, and applies those to efficiently sample and compress points in PointCompression.
- Sections 5 and 6 contain an overview and improvement to the computation of the fifteen $(2,2)$-isogenies between squared Kummer surfaces, including a new derivation of the elliptic $(2,2)$-isogenies between elliptic Kummer surfaces given by Costello [125].
- Section 7 combines all of the above to enable SQIsign verification on Kummer surfaces for both compressed and uncompressed signatures, including benchmarks in terms of finite-field operations.

SOFTWARE. Alongside the theory developed in this article, we provide accompanying software, written in MAGMA [66] and Python. Our code is available under the MIT license in the following repository:

https://github.com/Krijn-math/return-of-the-kummer

The source code contains the following:

- An optimised Python implementation of compressed and uncompressed SQIsign verification on Kummer surfaces, used for benchmarking. We use SageMath for the Jacobian arithmetic required in PointCompression, an algorithm performed only in signing.
- All the algorithms and maps in this article are implemented in MAGMA, with the aim to allow a reader to verify many of the claims made throughout and gain an understanding on how the various objects behave. To this end, we have documented the MAGMA code to expose various useful tricks and insights.

RELATED WORK. Kummer surfaces of genus-2 Jacobians were first introduced to cryptography by the Chudnovsky brothers [113], who gave a variant of Lenstra's ECM factoring algorithm. Gaudry [192] then proposed these Kummer surfaces as a setting for efficient discrete-logarithm-based cryptosystems. Many later works built on this to demonstrate high-speed, high-security Kummer-based implementations of Diffie–Hellman key exchange [39, 63, 325] and signature schemes [325, 326]. In particular, efficient vectorisation of Kummer arithmetic [39] lead to new speed records for high-security constant-time (hyper)elliptic curve Diffie–Hellman. In parallel to this, Lubicz and Robert [261] developed algorithms for efficient arithmetic on Kummer surfaces using the theory of theta functions of level 2, and gave efficient algorithms for pairing computation [262], later improved by Robert [331]

More recently, in a work by Costello [125], Kummer surfaces were introduced to isogeny-based cryptography in the context of SIDH. In particular, Costello extended Scholten's construction to transport any chain of 2-isogenies between elliptic curves over $\mathbb{F}_{p^2}$ to a chain of $(2,2)$-isogenies between Kummer surfaces, now defined over $\mathbb{F}_p$. Kummer surfaces are also implicitely used by Dartois, Maino, Pope, and Robert [142] to compute $(2,2)$-isogenies between theta structures of level 2. We note that the three $(2,2)$-isogenies described by Costello, which we rederive in this work, are somewhat special as they arise from 2-isogenies between elliptic curves. We therefore call these *elliptic* isogenies. Such elliptic isogenies can be computed more efficiently than the general $(2,2)$-isogenies.

Very recently, new variants of SQIsign have emerged [34, 166, 288] that use higher-dimensional techniques to achieve fast signing and verification for SQIsign. The breakthroughs are spectacular and shift the focus in SQIsign from one-dimensional to two-dimensional verification. Our work is different in that it transforms the one-dimensional verification of SQIsign into a two-dimensional isogeny. In this way, we demonstrate that one-dimensional SQIsign can itself be viewed as having two-dimensional verification. More generally, the aim of this work is to show that it is possible to transform one-dimensional protocols into two-dimensional protocols. Furthermore, because of the generality of the techniques developed in this work, we believe it can be applied in the context of two-dimensional SQIsign, and our exposition of genus-2 cryptography from the perspective of Kummer surfaces may clarify and complement the description of two-dimensional SQIsign for some readers.

§2 Kummer Surfaces

Chapter II gives an introduction to (hyper-)elliptic curves and their isogenies, and an introduction to SQIsign. Here, we give a detailed description of Kummer surfaces, assuming some knowledge of hyperelliptic curves.

This section discusses different models of Kummer surfaces, associated to the same (or isomorphic) hyperelliptic curve $\mathcal{C}$ defined over a field k. Given a hyperelliptic curve $\mathcal{C}$ of genus 2 with Jacobian $\mathcal{J}_\mathcal{C}$, the corresponding *Kummer surface* is given by the quotient $\mathcal{K} := \mathcal{J}_\mathcal{C}/\langle \pm 1 \rangle$. The Kummer surface has a quartic model in $\mathbb{P}^3$, so that $\mathcal{K}$ can be embedded into projective space with coordinates $(X_1 : X_2 : X_3 : X_4) \in \mathbb{P}^3$. Furthermore, $\mathcal{K}$ has sixteen point singularities, called *nodes*, given by the images of the 2-torsion points of $\mathcal{J}_\mathcal{C}$ under this quotient, as these are precisely the points fixed by $[-1] : P \mapsto -P$. The quotient map destroys the group law on the Jacobian $\mathcal{J}_\mathcal{C}$ and thus $\mathcal{K}$ only inherits scalar multiplication from $\mathcal{J}_\mathcal{C}$. However, we still have a *pseudo-group* law on $\mathcal{K}$, e.g., to add two points $P, Q \in \mathcal{K}$, we require knowledge of $P - Q$.

There are multiple types of Kummer surface models. For any hyperelliptic curve $\mathcal{C}$, we can construct the *general* Kummer surface $\mathcal{K}_\mathcal{C}^{\text{gen}}$, which we discuss in Section 2.1. When $\mathcal{C}$ is isomorphic over a field k to a curve in Rosenhain form then $\mathcal{C}_{\lambda,\mu,\nu}$ also admits a *canonical* and *squared* Kummer surface, which we describe in Sections 2.3 and 2.4, respectively. We also give maps $\mathcal{K}_\mathcal{C}^{\text{gen}} \to \mathcal{K}_{\lambda,\mu,\nu}^{\text{gen}}$ and $\mathcal{K}_{\lambda,\mu,\nu}^{\text{gen}} \to \mathcal{K}_{\lambda,\mu,\nu}^{\text{Sqr}}$. We then introduce and classify squared Kummer surfaces with $\nu = \lambda \cdot \mu$ in Section 2.5. These allow several optimizations beyond regular squared Kummer surfaces and we show their connection to elliptic curves through Scholten's construction in Section 2.9.

In many cases, the literature may call any of these models *the* Kummer surface $\mathcal{K}$. In this work, as we deal with several different isomorphic curves and their associated Kummers, we avoid this and use the following explicit notation. For a general hyperelliptic curve $\mathcal{C}$ of genus 2, with Jacobian $\mathcal{J}_\mathcal{C}$, we denote

- the general Kummer surface by $\mathcal{K}_\mathcal{C}^{\text{gen}}$, and
- the squared Kummer surface by $\mathcal{K}_\mathcal{C}^{\text{Sqr}}$, if it exists.

For a curve $\mathcal{C}_{\lambda,\mu,\nu}$ in Rosenhain form, with Jacobian $\mathcal{J}_{\lambda,\mu,\nu}$, we denote

- the associated general Kummer surface by $\mathcal{K}_{\lambda,\mu,\nu}^{\text{gen}}$,
- the associated squared Kummer surface by $\mathcal{K}_{\lambda,\mu,\nu}^{\text{Sqr}}$, and

- the associated elliptic Kummer surface by $\mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\lambda\mu}$, only if $\nu = \lambda\mu$.

For a curve $\mathcal{C}_\alpha$ associated to an elliptic curve E_α through Scholten's construction, with Jacobian $\mathcal{J}_\alpha$, we denote

- the general Kummer surface by $\mathcal{K}^{\text{gen}}_\alpha$, and
- the squared Kummer surface by $\mathcal{K}_\alpha$, a special form of $\mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\lambda\mu}$.

Any hyperelliptic curve of genus 2 has five or six x-values w_i where the curve intersects with $y = 0$. These are called the Weierstrass points $(w_i, 0)$. In a degree-5 model, we consider the (single) point at infinity as the Weierstrass point ∞. For a curve in Rosenhain form, the six Weierstrass points are $w_1 = \infty$, $w_2 = 0$, $w_3 = 1$, $w_4 = \lambda$, $w_5 = \mu$ and $w_6 = \nu$. This numbering is strictly and often used throughout this work whenever we work with curves in Rosenhain form, in particular to describe two-torsion. When using pairs w_i, w_j in this chapter, we *always* assume $i < j$. In particular, we can only have $i = 1$, and always have $j > 1$.

With our main motivation being Kummer surfaces for use in isogeny-based cryptography, throughout this work we often restrict to *superspecial* Jacobians of genus-2 curves and their associated Kummer surfaces, the natural analogue of supersingular elliptic curves to arbitrary dimension [73, 85, 257]. We refer to a hyperelliptic curve as superspecial when its Jacobian is superspecial. We furthermore require rational 2-torsion on the Jacobians and Kummer surfaces.

2.1 GENERAL KUMMER SURFACES

We begin by discussing the general Kummer surface in more detail. Readers only interested in the cryptographically relevant Kummer surfaces used later in this work can skip directly to Section 2.4.

Construction of the General Kummer Surface.

We follow Cassels and Flynn [84, §3] to describe the general Kummer surface. Consider a genus-2 curve $\mathcal{C}$ defined over a field k,

$$\mathcal{C} : y^2 : c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 + c_5 x^5 + c_6 x^6,\ c_i \in k,$$

where c_6 can equal 0 if $\mathcal{C}$ is in the degree 5 model.[2] The Kummer surface $\mathcal{K}^{\text{gen}}_{\mathcal{C}}$ corresponding to the curve $\mathcal{C}$ is defined over k and is given by elements $(X_1 :$

2 Sometimes called the *odd-degree model*.

$X_2 : X_3 : X_4) \in \mathbb{P}^3$ such that $K_2X_4^2 + K_1X_4 + K_0 = 0$, where the polynomials K_0, K_1 and K_2 are given as

$$K_2 := X_2^2 - 4X_1X_3,\ K_1 := -2\begin{pmatrix} c_1X_1^2X_2 + 2c_2X_1^2X_3 + c_3X_1X_2X_3 \\ +2c_0X_1^3 + 2c_4X_1X_3^2 + c_5X_2X_3^2 + 2c_6X_3^3 \end{pmatrix},$$

$$\begin{aligned} K_0 := {} & (c_1^2 - 4c_0c_2)X_1^4 - 4c_0c_3X_1^3X_2 - 2c_1c_3X_1^3X_3 - 4c_0c_4X_1^2X_2^2 \\ & + 4(c_0c_5 - c_1c_4)X_1^2X_2X_3 + (c_3^2 + 2c_1c_5 - 4c_2c_4 - 4c_0c_6)X_1^2X_3^2 \\ & - 4c_0c_5X_1X_2^3 + 4(2c_0c_6 - c_1c_5)X_1X_2^2X_3 + 4(c_1c_6 - c_2c_5)X_1X_2X_3^2 \\ & - 2c_3c_5X_1X_3^3 - 4c_0c_6X_2^4 - 4c_1c_6X_2^3X_3 - 4c_2c_6X_2^2X_3^2 \\ & - 4c_3c_6X_2X_3^3 + (c_5^2 - 4c_4c_6)X_3^4. \end{aligned}$$

The identity point $\mathbf{o} \in \mathcal{K}_{\mathcal{C}}^{\text{gen}}$ is given by $\mathbf{o} = (0 : 0 : 0 : 1)$.

Maps to the General Kummer Surface.

We can map pairs of points (x_1, y_1), (x_2, y_2) lying on $\mathcal{C}$ to $\mathcal{K}_{\mathcal{C}}^{\text{gen}}$, where $x_1 \neq x_2$, by $\rho : ((x_1, y_1), (x_2, y_2)) \mapsto (X_1 : X_2 : X_3 : X_4)$ where

$$X_1 := 1,\ \ X_2 := x_1 + x_2,\ \ X_3 := x_1x_2,\ \ X_4 := \frac{F(x_1, x_2) - 2y_1y_2}{(x_1 - x_2)^2},$$

with

$$\begin{aligned} F(x_1, x_2) = {} & 2c_0 + c_1(x_1 + x_2) + 2c_2x_1x_2 + c_3(x_1 + x_2)x_1x_2 + 2c_4(x_1x_2)^2 \\ & + c_5(x_1 + x_2)(x_1x_2)^2 + 2c_6(x_1x_2)^3. \end{aligned}$$

We construct the map $\tilde{\rho} : \mathcal{J}_{\mathcal{C}} \to \mathcal{K}_{\mathcal{C}}^{\text{gen}}$, from the Jacobian $\mathcal{J}_{\mathcal{C}}$ to $\mathcal{K}_{\mathcal{C}}^{\text{gen}}$, by constructing all the rational functions in ρ in terms of the coefficients of a divisor[3] $\langle x^2 + a_1x + a_0, b_1x + b_0\rangle \in \mathcal{J}_{\mathcal{C}}$. Thus, we get $\tilde{\rho} : \langle x^2 + a_1x + a_0, b_1x + b_0\rangle \mapsto (X_1 : X_2 : X_3 : X_4)$ where

$$X_1 := 1,\ \ X_2 := -a_1,\ \ X_3 := a_0,\ \ X_4 := \frac{F'(a_0, a_1) - 2(b_1^2a_0 - b_0b_1a_1 + b_0^2)}{a_1^2 - 4a_0},$$

3 In this work, we always use the *Mumford representation* for elements of Jacobians.

with $F'(a_0, a_1) = 2c_0 - c_1a_1 + 2c_2a_0 - c_3a_0a_1 + 2c_4a_0^2 - c_5a_1a_0^2 + 2c_6a_0^3$. We emphasise that this construction of $\mathcal{K}_{\mathcal{C}}^{\text{gen}}$ associated to $\mathcal{C}$ applies to any hyperelliptic curve $\mathcal{C}$ of genus 2.

Points of Order 2 on the General Kummer Surface.

The map $\tilde{\rho} : \mathcal{J}_{\mathcal{C}} \to \mathcal{K}_{\mathcal{C}}$ is of order 2 except at the sixteen points in $\mathcal{J}_{\mathcal{C}}[2]$ which map precisely to the sixteen nodes $\mathcal{K}_{\mathcal{C}}[2]$. The elements of order 2 in $\mathcal{J}_{\mathcal{C}}$ are given by divisors $D_{i,j}$, where the index i and j refer to pairs of Weierstrass points $(w_i, 0) + (w_j, 0)$ in the support of $D_{i,j}$ for $1 \leq i < j \leq 6$. The Mumford representation of $D_{i,j}$ is $\langle x^2 - (w_i + w_j)x + w_i \cdot w_j, 0\rangle$ whenever $w_i, w_j \neq \infty$. Whenever ∞ is a Weierstrass point (i.e., when using the degree 5 model) we consider $w_1 = \infty$ and the Mumford representation of $L_{1,j}$ simply ignores this factor $(x - w_1)$. Using $\tilde{\rho}$, we find the sixteen points $L_{i,j}$ of order 2 on $\mathcal{K}_{\mathcal{C}}$ given by

$$\begin{aligned} L_{i,j} &= (1 : w_i + w_j : w_iw_j : F(w_i, w_j)/(w_i - w_j)^2), && \text{when } w_i, w_j \neq \infty, \\ L_{1,j} &= (0 : 1 : w_j : w_j^2), && \text{when } w_1 = \infty. \end{aligned}$$

Addition by Points of Order 2 on the General Kummer Surface.

Addition by points of order 2 on Kummer surfaces is well-defined, and yields a linear map from $\mathcal{K}_{\mathcal{C}}^{\text{gen}}$ to itself. For $L_{i,j} \in \mathcal{K}[2]$ of order 2, we can represent the translation by $L_{i,j}$, e.g. $P \mapsto P + L_{i,j}$, as a 4×4 matrix $W_{i,j}$ over k. As these maps are involutions on $\mathcal{K}_{\mathcal{C}}^{\text{gen}}$, we get $W_{i,j}^2 = c \cdot I_4$ for some $c \in k$. These matrices are computed and described by Cassels and Flynn [84], we provide a compact presentation in Section A.

2.2 ROSENHAIN FORM OF A HYPERELLIPTIC CURVE

To construct the canonical and squared Kummer surfaces, we require a hyperelliptic curves in Rosenhain form $\mathcal{C}_{\lambda,\mu,\nu}$ defined over k. These Kummer surfaces models are theta structures of level 2, which implies that $\mathcal{K}[2]$ plays an important role in their arithmetic. We will see this in later sections when we give efficient algorithms to compute 2-pairings on and $(2,2)$-isogenies between these Kummer surfaces using their points of order 2.

Recall from Chapter II that a hyperelliptic curve $\mathcal{C}_{\lambda,\mu,\nu}$ is in *Rosenhain form* when

$$\mathcal{C}_{\lambda,\mu,\nu} : y^2 = x(x-1)(x-\lambda)(x-\mu)(x-\nu) \quad \text{with } \lambda, \mu, \nu \in k\,.$$

and that the values λ, μ and ν are called the *Rosenhain invariants* of $\mathcal{C}_{\lambda,\mu,\nu}$.

The Rosenhain form of a hyperelliptic curve of genus 2 can be viewed as somewhat of an analogue to the Montgomery form of elliptic curves[4]. Whereas a general elliptic curve in (short) Weierstrass form admits x-only arithmetic, this x-only arithmetic is much more efficient for curves in Montgomery form. We find a similar situation in genus 2: the general Kummer surface can be constructed for any hyperelliptic curve $\mathcal{C}$, yet high-speed cryptography requires the use of the more efficient Kummer surfaces, which arise from the theory of theta functions.

The original idea to use such Kummer surfaces in cryptography is due to the Chudnovsky brothers [113] in 1986. These more efficient Kummer surfaces come in two forms: the *canonical* Kummer surface, as described by Gaudry [192]; and the closely related *squared* Kummer surface, described by Bernstein [42].

2.3 CANONICAL KUMMER SURFACE

Following Gaudry [192], the canonical Kummer surface associated to a hyperelliptic curve $\mathcal{C}_{\lambda,\mu,\nu}$ over k in Rosenhain form is defined by four *fundamental theta constants* which can be computed from the Rosenhain invariants of $\mathcal{C}_{\lambda,\mu,\nu}$. Given a hyperelliptic curve $\mathcal{C}_{\lambda,\mu,\nu}$ with Rosenhain invariants $\lambda, \mu, \nu \in k$, we define the fundamental theta constants $a, b, c, d \in \overline{k}$ following [63] as

$$d^2 = 1, \quad c^2 = \sqrt{\frac{\lambda\mu}{\nu}}, \quad b^2 = \sqrt{\frac{\mu(\mu-1)(\lambda-\nu)}{\nu(\nu-1)(\lambda-\mu)}}, \quad a^2 = b^2c^2\frac{\nu}{\mu}. \tag{15}$$

and their associated *dual fundamental theta constants* $A, B, C, D \in \overline{k}$ such that

$$\begin{aligned} A^2 &= a^2 + b^2 + c^2 + d^2, \quad B^2 = a^2 + b^2 - c^2 - d^2, \\ C^2 &= a^2 - b^2 + c^2 - d^2, \quad D^2 = a^2 - b^2 - c^2 + d^2. \end{aligned} \tag{16}$$

4 Although the *Legendre form* $y^2 = x(x-1)(x-\lambda)$ is perhaps a better analogy.

Given theta constants, we can also derive Rosenhain invariants for the associated curve by

$$\lambda = \frac{a^2c^2}{b^2d^2},\ \mu = \frac{c^2e^2}{d^2f^2},\ \nu = \frac{a^2e^2}{b^2f^2},$$

where $e, f \in \overline{k}$ such that $e^2/f^2 = (AB + CD)/(AB - CD)$. Note that the constants a, b, c, d are defined up to sign, but the resulting Kummer surfaces are isomorphic. The canonical Kummer surface $\mathcal{K}^{\mathrm{can}}_{\lambda,\mu,\nu}$ defined over $\overline{k}$ is then given by the following equation.

$$\mathcal{K}^{\mathrm{can}}_{\lambda,\mu,\nu}: \quad \begin{array}{c} T_1^4 + T_2^4 + T_3^4 + T_4^4 + 2E \cdot T_1T_2T_3T_4 \\ = \\ F \cdot (T_1^2T_4^2 + T_2^2T_3^2) + G \cdot (T_1^2T_3^2 + T_2^2T_4^2) + H \cdot (T_1^2T_2^2 + T_3^2T_4^2). \end{array}$$

where

$$E := \frac{(16ABCD)^2 \cdot abcd}{(a^2d^2 - b^2c^2)(a^2c^2 - b^2d^2)(a^2b^2 - c^2d^2)}, \text{ and}$$
$$F := \frac{a^4 - b^4 - c^4 + d^4}{a^2d^2 - b^2c^2},\ G := \frac{a^4 - b^4 + c^4 - d^4}{a^2c^2 - b^2d^2},\ H := \frac{a^4 + b^4 - c^4 - d^4}{a^2b^2 - c^2d^2}.$$

Note that as A^2, B^2, C^2, D^2 are linear combinations of a^2, b^2, c^2, d^2, the equation for $\mathcal{K}^{\mathrm{can}}_{\lambda,\mu,\nu}$ is determined entirely by a, b, c, d. The identity point $\mathbf{o} \in \mathcal{K}^{\mathrm{can}}_{\lambda,\mu,\nu}$ is given by $\mathbf{o} = (a : b : c : d) \in \mathcal{K}^{\mathrm{can}}_{\lambda,\mu,\nu}$. We will not use this model of Kummer surface, more details of the arithemtic on these surfaces are given by Gaudry [192].

Points of Order 2 on the Canonical Kummer Surface.

The 16 points of order 2 on $\mathcal{K}^{\mathrm{can}}_{\lambda,\mu,\nu}$ are

$$\begin{array}{llll} \mathbf{o} = (a:b:c:d), & (a:b:-c:-d), & (a:-b:c:-d), & (a:-b:-c:d), \\ (b:a:d:c), & (b:a:-d:-c), & (b:-a:d:-c), & (b:-a:-d:c), \\ (c:d:a:b), & (c:d:-a:-b), & (c:-d:a:-b), & (c:-d:-a:b), \\ (d:c:b:a), & (d:c:-b:-a), & (d:-c:b:-a), & (d:-c:-b:a). \end{array}$$

For the addition matrices for these points, see Section A.

2.4 SQUARED KUMMER SURFACE

The squared Kummer surface has been a subject of interest in hyperelliptic curve cryptography [39, 42, 63, 325] as it boasts very fast arithmetic when one can stay on a single surface.

Construction of the Squared Kummer Surface.

The squared Kummer surface corresponding to $\mathcal{C}_{\lambda,\mu,\nu}$ is closely related to the canonical Kummer surface $\mathcal{K}^{\text{can}}_{\lambda,\mu,\nu}$. An isomorphism is given via the squaring map $\mathcal{K}^{\text{can}}_{\lambda,\mu,\nu} \to \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu} : (T_1 : T_2 : T_3 : T_4) \to (T_1^2 : T_2^2 : T_3^2 : T_4^2)$. The squared Kummer surface $\mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$ is defined by four constants

$$(\mu_1 : \mu_2 : \mu_3 : \mu_4) := (a^2 : b^2 : c^2 : d^2),$$

where now $\mu_i \in k$ as long as $\mathcal{C}_{\lambda,\mu,\nu}$ is defined over k. From this, we obtain a relation between the μ_i and the dual fundamental theta constants as follows

$$\begin{aligned} A^2 &= \mu_1 + \mu_2 + \mu_3 + \mu_4, \quad B^2 = \mu_1 + \mu_2 - \mu_3 - \mu_4 \\ C^2 &= \mu_1 - \mu_2 + \mu_3 - \mu_4, \quad D^2 = \mu_1 - \mu_2 - \mu_3 + \mu_4. \end{aligned} \tag{17}$$

The equation defining the squared Kummer is given by

$$\mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu} : E \cdot X_1X_2X_3X_4 = \begin{pmatrix} X_1^2 + X_2^2 + X_3^2 + X_4^2 - F \cdot (X_1X_4 + X_2X_3) \\ -G \cdot (X_1X_2 + X_2X_4) - H \cdot (X_1X_2 + X_3X_4) \end{pmatrix}^2.$$

with

$$F := \frac{\mu_1^2 - \mu_2^2 - \mu_3^2 + \mu_4^2}{\mu_1\mu_4 - \mu_2\mu_3}, \quad G := \frac{\mu_1^2 - \mu_2^2 + \mu_3^2 - \mu_4^2}{\mu_1\mu_3 - \mu_2\mu_4}, \quad H := \frac{\mu_1^2 + \mu_2^2 - \mu_3^2 - \mu_4^2}{\mu_1\mu_2 - \mu_3\mu_4},$$

$$E := 4\mu_1\mu_2\mu_3\mu_4 \left(\frac{A^2B^2C^2D^2}{(\mu_1\mu_1 - \mu_3\mu_4)(\mu_1\mu_3 - \mu_2\mu_4)(\mu_1\mu_4 - \mu_2\mu_3)} \right)^2.$$

The identity point $\mathbf{o} \in \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ is $\mathbf{o} = (\mu_1 : \mu_2 : \mu_3 : \mu_4)$. As given by Bos, Costello, Hisil, and Lauter [63], we can derive the constants $\mu_1, \mu_2, \mu_3, \mu_4$ from the Rosenhain invariants $\lambda, \mu, \nu \in k$ as

$$\mu_4 = 1, \quad \mu_3 = \sqrt{\frac{\lambda\mu}{\nu}}, \quad \mu_2 = \sqrt{\frac{\mu(\mu-1)(\lambda-\nu)}{\nu(\nu-1)(\lambda-\mu)}}, \quad \mu_1 = \mu_2\mu_3\frac{\nu}{\mu}. \tag{18}$$

Map from the Jacobian to the Squared Kummer Surface.

The map $\rho^{\mathrm{Sqr}} : \mathcal{J}_{\lambda,\mu,\nu} \to \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ is given by [114, 123], where any divisor $D = \langle x^2 + u_1 x + u_0, v_1 x + v_0 \rangle$ is mapped to a point $(X_1 : X_2 : X_3 : X_4) \in \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ by

$$\begin{aligned}
X_1 &= \mu_1 \cdot \left(u_0(w_3 w_5 - u_0)(w_4 + w_6 + u_1) - v_0^2 \right), \\
X_2 &= \mu_2 \cdot \left(u_0(w_4 w_6 - u_0)(w_3 + w_5 + u_1) - v_0^2 \right), \\
X_3 &= \mu_3 \cdot \left(u_0(w_3 w_6 - u_0)(w_4 + w_5 + u_1) - v_0^2 \right), \\
X_4 &= \mu_4 \cdot \left(u_0(w_4 w_5 - u_0)(w_3 + w_6 + u_1) - v_0^2 \right).
\end{aligned} \tag{19}$$

Here, $w_3 = 1$, $w_4 = \lambda$, $w_5 = \mu$, and $w_6 = \nu$ are the Weierstrass points with our fixed numbering.

Points of order 2 on the Squared Kummer Surface.

The 16 points in $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}[2]$, which we denote $\mathcal{K}[2]$, correspond precisely to the 16 elements $D_{i,j}$ of order 2 on $\mathcal{J}_{\lambda,\mu,\nu}[2]$. To fill a gap in the literature on this topic, we give a full description of $\mathcal{K}[2]$. Suppose $\mu_1, \mu_2, \mu_3, \mu_4$ are the theta constants of $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ with Rosenhain invariants $\lambda, \mu, \nu \in k$.[5] Let τ and $\tilde{\tau}$ denote the roots of $x^2 - Gx + 1$ (with G as in the defining equation), so that $\tilde{\tau} = 1/\tau$. Let $L_{i,j}$ be the element in $\mathcal{K}[2]$ corresponding to $D_{i,j} = \langle (x - w_i)(x - w_j), 0 \rangle \in \mathcal{J}_{\lambda,\mu,\nu}[2]$, and $D_{1,j} = \langle (x - w_j), 0 \rangle$. Then

5 Do not confuse the Rosenhain invariant μ with the theta constants μ_i.

$$\begin{aligned}
\mathbf{0} &= (\mu_1 : \mu_2 : \mu_3 : \mu_4), & L_{1,2} &= (\mu_2 : \mu_1 : \mu_4 : \mu_3),\\
L_{5.6} &= (\mu_3 : \mu_4 : \mu_1 : \mu_2), & L_{3,4} &= (\mu_4 : \mu_3 : \mu_2 : \mu_1),\\
L_{4,5} &= (\mu \cdot \mu_4 : \mu_3 : 0 : 0), & L_{3,6} &= (\mu_3 : \mu \cdot \mu_4 : 0 : 0),\\
L_{4,6} &= (0 : 0 : \mu \cdot \mu_4 : \mu_3), & L_{3,5} &= (0 : 0 : \mu_3 : \mu \cdot \mu_4),\\
L_{2,3} &= ((\nu-1)\mu_2 : 0 : (\mu-1)\mu_4 : 0), & L_{1,4} &= ((\mu-1)\mu_4 : 0 : (\nu-1)\mu_2 : 0),\\
L_{1,3} &= (0 : (\nu-1)\mu_2 : 0 : (\mu-1)\mu_4), & L_{2,4} &= (0 : (\mu-1)\mu_4 : 0 : (\nu-1)\mu_2),\\
L_{2,5} &= (\tau : 0 : 0 : 1), & L_{1,6} &= (1 : 0 : 0 : \tau),\\
L_{2,6} &= (0 : 1 : \tau : 0), & L_{1,5} &= (0 : \tau : 1 : 0).
\end{aligned}$$

Remark 1. We can also write these 2-torsion points in terms of dual constants A, B, C, D rather than Rosenhain invariants. Indeed, following Gaudry in the proof of Lemma 4.2 in [192] and [192, §7.5], we have that

$$\frac{AB+CD}{AB-CD} = \mu \cdot \frac{\mu_4}{\mu_3}, \quad \frac{AC+BD}{AC-BD} = \frac{(\nu-1)\mu_2}{(\mu-1)\mu_4}, \quad \frac{AD+BC}{AD-BC} = \frac{(\mu-\lambda)\mu_2}{(\mu-1)\mu_3}.$$

Addition by Points of Order 2 on the Squared Kummer Surface.

As far as we are aware, the linear maps $W_{i,j}$ that represent addition by a point $L_{i,j}$ of order 2 on the squared Kummer surface do not appear in the literature. We present an approach to derive such matrices, with more details given in Section A. The resulting matrices $W_{i,j}$ are available in our code. This approach differs from the usual algebraic approach and a similar approach works for any Kummer surface or, generally, any projective linear map.

Let $D_{i,j} \in \mathcal{J}[2]$ denote the pre-image of $L_{i,j} \in \mathcal{K}[2]$. Our goal is to find a description of $W_{i,j}$. On the Jacobian, it is easy to add $D_{i,j}$ to any random element $D_Q \in \mathcal{J}$. Thus, we can take a large enough sequence of points $\mathbf{A}_\mathcal{J} = [D_1, \ldots, D_m]$ and apply this translation to get $\mathbf{B}_\mathcal{J} = [D_1 + D_{i,j}, \ldots, D_m + D_{i,j}]$.

We map both sequences $\mathbf{A}_\mathcal{J}$ and $\mathbf{B}_\mathcal{J}$ down to $\mathcal{K}^{\text{Sqr}}$ to get $\mathbf{A}_\mathcal{K}$ and $\mathbf{B}_\mathcal{K}$. Let A_n denote the n-th element of $\mathbf{A}_\mathcal{K}$, and similarly for B_n. Then $W_{i,j}$ must map A_n to $\lambda_n B_n$ for some $\lambda_n \in k$, as points on the Kummer are defined up to a scalar. Hence, we get

$$W_{i,j}\mathbf{A}_\mathcal{K} = \mathbf{B}_\mathcal{K}\Lambda,$$

where Λ denotes the diagonal matrix of size $m \times m$ with λ_n on the diagonal. Assuming $W_{i,j}$ and Λ are unknown, with m large enough, a Gröbner basis computation readily yields a solution for $W_{i,j}$, up to some unknown scaling

factor. Given such a solution for $W_{i,j}$, we normalise the top-left corner to 1. By performing this for a few different concrete instantiations of curves $\mathcal{C}$ and primes p, we are able to determine the coefficients of each $W_{i,j}$ in terms of the theta constants μ_i and the Rosenhain invariants λ, μ, ν. We then verify the correctness of the resulting matrices $W_{i,j}$.

2.5 ELLIPTIC KUMMER SURFACES

In this article, we will work with a special squared Kummer surface which arises from a Rosenhain curve $\mathcal{C}_{\lambda,\mu,\nu}$ with $\nu = \lambda\mu$. Such Kummer surfaces have strong ties to elliptic curves, as we show in Lemma 1 and becomes even more concrete when we introduce Scholten's construction in Section 2.9. We therefore call such Kummer surfaces *elliptic*. In this section, we discuss special properties of the elliptic Kummer surface, which are needed to construct our toolbox for Kummer surfaces in Sections 3 and 4.

Lemma 1. The Jacobian of a genus-2 curve $\mathcal{C}/\overline{k}$ is $(2,2)$-isogenous to a product of elliptic curves $E \times E'$ if and only if $\mathcal{C}$ has Rosenhain invariants satisfying $\nu = \lambda\mu$.

Proof. Let $\mathcal{J}_{\lambda,\mu,\lambda\mu}$ be the Jacobian of a Rosenhain curve $\mathcal{C}_{\lambda,\mu,\lambda\mu}$ of genus 2. In particular, here $\nu = \lambda\mu$. The quadratic splitting (see [356, § 8.2]) of $\mathcal{C}_{\lambda,\mu,\lambda\mu}$ by $G_1 = x$, $G_2 = (x-1)(x-\nu)$, $G_3 = (x-\lambda)(x-\mu)$ has determinant 0, hence $\mathcal{J}_{\lambda,\mu,\lambda\mu}$ is $(2,2)$-isogenous to a product of elliptic curves. For the other direction, let $\mathcal{J}_{\lambda,\mu,\nu}$ be $(2,2)$-isogenous to a product of elliptic curves. Then, up to trivial reordering, this Richelot isogeny is given by the splitting $G_1 = x$, $G_2 = (x-1)\cdot(x-w_i)$, $G_3 = (x-w_j)(x-w_k)$ for some assignment of i, j, k to $\{4,5,6\}$. By $\det(G_1, G_2, G_3) = 0$ we get $w_i = w_j \cdot w_k$. Permuting the Rosenhain invariants, we get $\nu = \lambda\mu$. □

Remark 2. Lemma 1 suggests that the moduli space of elliptic Kummers given by the Igusa-Clebsch invariants [215, p. 620] is a 'nice' subspace of the general moduli space.

Construction of the Elliptic Kummer Surface.

We construct an elliptic Kummer surface $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\lambda\mu}$ in the same way as any squared Kummer surface, as described in Section 2.4, with certain simplifications given $\nu = \lambda \cdot \mu$. Thus, Equation (18) tells us for elliptic Kummer surfaces $\mu_3 = \mu_4 = 1$,

greatly simplifying the equation defining $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\lambda\mu}$. For example, in this case $G = \mu_1 + \mu_2$.

Points of Order 2 on the Elliptic Kummer Surface.

In the case where $\nu = \lambda\mu$, we have that $\mu_3 = \mu_4 = 1$ and $\tau = \frac{(\mu-1)\mu_4}{(\nu-1)\mu_2}$. In this way, the two-torsion points simplify as follows:

$$\begin{aligned}
\mathbf{0} &= (\mu_1 : \mu_2 : 1 : 1), & L_{1,2} &= (\mu_2 : \mu_1 : 1 : 1),\\
L_{5,6} &= (1 : 1 : \mu_1 : \mu_2), & L_{3,4} &= (1 : 1 : \mu_2 : \mu_1),\\
L_{4,5} &= (\mu : 1 : 0 : 0), & L_{3,6} &= (1 : \mu : 0 : 0),\\
L_{4,6} &= (0 : 0 : \mu : 1), & L_{3,5} &= (0 : 0 : 1 : \mu),\\
L_{2,3} &= (1 : 0 : \tau : 0), & L_{1,4} &= (\tau : 0 : 1 : 0),\\
L_{1,3} &= (0 : 1 : 0 : \tau), & L_{2,4} &= (0 : \tau : 0 : 1),\\
L_{2,5} &= (\tau : 0 : 0 : 1), & L_{1,6} &= (1 : 0 : 0 : \tau),\\
L_{2,6} &= (0 : 1 : \tau : 0), & L_{1,5} &= (0 : \tau : 1 : 0).
\end{aligned}$$

Note the symmetry between the left and right column, which positively impacts the efficiency of arithmetic on elliptic Kummer surfaces.

Addition by Points of Order 2 on the Elliptic Kummer Surface.

The matrices that describe addition by points of order 2 on $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\lambda\mu}$ are easily derived from the addition matrices $W_{i,j}$ of $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ by specialising to $\mu_3 = \mu_4 = 1$. We give these matrices concretely in Section A.

2.6 ARITHMETIC OF SQUARED KUMMER SURFACES

As with elliptic curves, the construction of the Kummer surface $\mathcal{K}$ as the quotient of $\mathcal{J}$ by ± 1 implies that we only have a *pseudo*-group structure on $\mathcal{K}$. Nevertheless, this is enough to compute scalar multiplications $P \mapsto [n]P$ and differential addition, and in general is rich enough for cryptographic applications. The pseudo-group structure on squared Kummer surfaces looks particularly nice due to surprisingly elegant symmetries.

CORE MORPHISMS. The arithmetic on Kummer surfaces is constructed from four core morphisms fron $\mathbb{P}_3$ to $\mathbb{P}_3$, namely the squaring map $\mathtt{S}$, the Hadamard involution $\mathtt{H}$, the scaling map $\mathtt{C}_P$, and the inversion map $\mathtt{Inv}$, defined as follows:

$$\begin{aligned}
\mathtt{S}\colon (X\colon Y\colon Z\colon T) &\mapsto (X^2\colon Y^2\colon Z^2\colon T^2),\\
\mathtt{H}\colon (X\colon Y\colon Z\colon T) &\mapsto \begin{array}{l}(X+Y+Z+T\colon X+Y-Z-T\colon\\ \quad X-Y+Z-T\colon X-Y-Z+T),\end{array}\\
\mathtt{C}_{(P_1\colon P_2\colon P_3\colon P_4)}\colon (X\colon Y\colon Z\colon T) &\mapsto (P_1\cdot X\colon P_2\cdot Y\colon P_3\cdot Z\colon P_4\cdot T),\\
\mathtt{Inv}\colon (X\colon Y\colon Z\colon T) &\mapsto (1/X\colon 1/Y\colon 1/Z\colon 1/T)\\
&\qquad = (YZT\colon XZT\colon XYT\colon XYZ).
\end{aligned}$$

The map $\mathtt{S}$ costs $4\mathbf{S}$, $\mathtt{H}$ costs $8\mathbf{a}$, $\mathtt{C}_P$ costs $4\mathbf{M}$, and $\mathtt{Inv}$ costs $6\mathbf{M}$.

CURVE ARITHMETIC. The four basic morphisms $\mathtt{S}, \mathtt{H}, \mathtt{C}$ and $\mathtt{Inv}$ are enough to define the curve operations doubling $\mathtt{xDBL} : P \mapsto 2P$, differential addition $\mathtt{xADD} : P, Q, P-Q \mapsto P+Q$, scalar multiplication $\mathtt{xMUL} : P \mapsto [n]P$ as described in [114, Appendix A]. We also derive a three-point ladder $P, Q, P-Q, s \mapsto P+sQ$.

2.7 MAPS BETWEEN KUMMER SURFACES

We can derive a map between $\mathcal{K}_{\mathcal{C}}^{\text{gen}} \to \mathcal{K}_{\lambda,\mu,\nu}^{\text{gen}}$ from an isomorphism between the curve $\mathcal{C}$ and its Rosenhain form $\mathcal{C}_{\lambda,\mu,\nu}$. When the Weierstrass points of $\mathcal{C}$ are k-rational, there is a k-rational isomorphism $\kappa : \mathcal{C} \to \mathcal{C}_{\lambda,\mu,\nu}$ between $\mathcal{C}$ and $\mathcal{C}_{\lambda,\mu,\nu}$, given by five values $(a, b, c, d, e) \in k$, namely

$$\kappa : \mathcal{C} \to \mathcal{C}_{\lambda,\mu,\nu}, \quad (x, y) \mapsto \left(\frac{ax+b}{cx+d}, \frac{ey}{(cx+d)^2}\right)$$

as described by Costello [125, §2]. This map κ induces a map $\kappa_* : \mathcal{J}_{\mathcal{C}} \to \mathcal{J}_{\lambda,\mu,\nu}$ between their Jacobians [125, §3] and similarly a map $\kappa_{**} : \mathcal{K}_{\mathcal{C}}^{\text{gen}} \to \mathcal{K}_{\lambda,\mu,\nu}^{\text{gen}}$ given in terms of $a, b, c, d, e^2 \in k$. For simplicity, we normalise the inputs and outputs to the first coordinate, so that $\kappa_{**} : (1 : X_2 : X_3 : X_4) \mapsto (1 : X_2' : X_3' : X_4')$ with

$$X_2' = \frac{2acX_3 + (ad+bc)X_2 + 2bd}{c^2X_3 + cdX_2 + d^2}, \quad X_3' = \frac{a^2X_3 + abX_2 + b^2}{c^2X_3 + cdX_2 + d^2}$$

and X_4' computed in terms of the defining polynomials K_1, K_2, K_3 for the domain and K_1', K_2', K_3' for the codomain as

$$X_4' = -\frac{K_1'(1, X_2', X_3') + 4v'}{2(X_2'^2 - 4X_3')}$$

where

$$v' = -\frac{e^2 \cdot (K_1(1, X_2, X_3) + 2X_4 K_2(1, X_2, X_3))}{4(c^2 X_3 + cdX_2 + d^2)^3}.$$

The isomorphism between $\mathcal{K}^{\text{gen}}_{\lambda,\mu,\nu}$ and $\mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$, associated to the same curve $\mathcal{C}_{\lambda,\mu,\nu}$, is also known, and given in [114] as a matrix $\mathbf{M}$. To derive $\mathbf{M}$, one can interpolate image points under both $\rho^*_{\lambda,\mu,\nu}$ and ρ^{Sqr} of divisors of $D \in \mathcal{J}_{\lambda,\mu,\nu}$. The maps κ, κ_*, κ_{**} and $\mathbf{M}$ are visualised in Figure 21.

2.8 MAP FROM KUMMER TO JACOBIAN

For several applications later in this work (such as Sections 3 and 4), we require a (partial) inverse of the map $\rho^{\text{Sqr}} : \mathcal{J}_{\lambda,\mu,\nu} \to \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$, i.e., a map that computes $D, -D \in \mathcal{J}_{\lambda,\mu,\nu}$ given a point $P = (X_1 : X_2 : X_3 : X_4) \in \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$ such that $\rho^{\text{Sqr}}(D) = \rho^{\text{Sqr}}(-D) = P$. Even simply recovering the values u_0, u_1 or the value v_0^2 of the Mumford representation is enough for such applications. Such maps were originally given by Gaudry [192], making use of additional theta constants and functions. When working on a single Kummer surface, these constants and functions are fixed, and we can easily precompute these. In isogeny-based cryptography, however, we no longer have this luxury, and the computation of these additional constants and functions is rather expensive. Using algebraic tools, detailed in Section C, we find more elegant and efficient maps

$$(\rho^{\text{Sqr}})^{-1} : \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu} \to \mathcal{J}_{\lambda,\mu,\nu}, \qquad P = (X_1 : X_2 : X_3 : X_4) \to \{D, -D\}$$

using three polynomials $F_0, F_1, F_2 \in k[X]$ and $\tilde{X}_i := X_i/\mu_i$ to recover u_0 and u_1, given by

$$\begin{aligned} F_0(X) =& (w_4 - w_6)\tilde{X}_1 + (w_3 - w_5)\tilde{X}_2 - (w_4 - w_5)\tilde{X}_3 - (w_3 - w_6)\tilde{X}_4, \\ F_1(X) =& (w_3 - w_5)w_4 w_6 \tilde{X}_1 + (w_4 - w_6)w_3 w_5 \tilde{X}_2 \\ & - (w_3 - w_6)w_4 w_5 \tilde{X}_3 - (w_4 - w_5)w_3 w_6 \tilde{X}_4, \\ F_2(X) =& -(\tilde{X}_1 + \tilde{X}_2 - \tilde{X}_3 - \tilde{X}_4)(w_3 w_4 - w_5 w_6). \end{aligned} \tag{20}$$

We then recover u_0 and u_1 as

$$u_0 = F_1(\tilde{X})/F_0(\tilde{X}), \quad u_1 = F_2(\tilde{X})/F_0(\tilde{X}). \tag{21}$$

This recovery works regardless of the projective representation of P, i.e. we recover the same u_0 and u_1 if we consider $P = (\omega X_1 : \omega X_2 : \omega X_3 : \omega X_4)$ for any $\omega \in k$ as the image of D under ρ^{Sqr}. In particular, we can apply this to any randomly sampled point on $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$. To recover v_0^2, we require an additional polynomial $G \in k[X]$, which allow us to recover ω.

$$\begin{aligned} G(X) &= -(w_3 - w_4)(w_5 - w_6)(w_3 w_4 - w_5 w_6)F_1(\tilde{X}), \\ \omega &= G(\tilde{X})/F_0^2(\tilde{X}). \end{aligned} \tag{22}$$

Together with u_0 and u_1, we recover v_0^2 by computing

$$v_0^2 = u_0(w_3 w_5 - u_0)(w_4 + w_6 + u_1) - \omega \tilde{X}_1. \tag{23}$$

For most applications, having u_0, u_1, v_0^2 is sufficient. However, to fully recover the points $\{D, -D\}$, we need to compute the corresponding v_0, v_1. To do so, we compute the two roots x_1, x_2 of $x^2 + u_1 x + u_0$ and get the corresponding y-values y_1, y_2 such that $(x_1, \pm y_1)$ and $(x_2, \pm y_2)$ lie on the curve $\mathcal{C}_{\lambda,\mu,\nu}$. As there are two possible y_i-values for each x_i, we set

$$\widetilde{v_0} = \frac{(y_1 - x_2)(y_2 - x_2)}{x_2 - x_1}$$

and choose the y-values such that $(\widetilde{v_0})^2$ matches the v_0^2 computed above, and derive v_1 as $v_1 = (y_1 - y_2)/(x_1 - x_2)$.

Remark 3. We stress that the above map is non-trivial: recovering u_0, u_1 and v_0^2 *efficiently* is a major improvement to working with Kummer surfaces effectively, as it allows us to use the additional structure of the Jacobian when needed.

2.9 SCHOLTEN'S CONSTRUCTION

In 2003, Scholten [338] introduced a specific Kummer surface $\mathcal{K}_\alpha$ defined over $\mathbb{F}_p$ associated to a given elliptic curve E_α over $\mathbb{F}_{p^2}$ with rational 2-torsion. This construction provides a tool to translate cryptographic protocols defined between elliptic curves to one between Kummer surfaces, which we will exploit in Section 7. The Kummer surfaces $\mathcal{K}_\alpha$ derived in this construction are *elliptic*,

that is, they can be described as the squared Kummer surface of a curve $\mathcal{C}$ with Rosenhain invariants $\lambda, \mu, \lambda\mu$ (see Section 2.5). This subsection describes Scholten's derivation.

As this class of Kummer surfaces will be of most interest to us, from this point forward in the article, we will restrict to $k = \mathbb{F}_p$ or $k = \mathbb{F}_{p^2}$, with $p = 3 \bmod 4$ and write $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$ with $i^2 = -1$. We remark that our results hold more generally than this.

THE WEIL RESTRICTION. Let $E_\alpha/\mathbb{F}_{p^2} : y^2 = x(x-\alpha)(x-\frac{1}{\alpha})$ be an elliptic curve where $x = x_0 + ix_1$, $y = y_0 + iy_1$, and $\alpha = \alpha_0 + i\alpha_1$. The *Weil restriction* W_α of E_α is an abelian surface defined over $\mathbb{F}_p$ given by

$$W_\alpha := \mathrm{Res}_{\mathbb{F}_p}^{\mathbb{F}_{p^2}}(E_\alpha) = V(W_0, W_1),$$

with W_0 and W_1 given by the *real* and *imaginary* part (respectively) of

$$(y_0 + iy_1)^2 - (x_0 + ix_1)\left((x_0 + ix_1) - (\alpha_0 + i\alpha_1)\right)\left((x_0 + ix_1) - \frac{1}{(\alpha_0 + i\alpha_1)}\right).$$

Scholten [338] shows that the Weil restriction of $E_\alpha/\mathbb{F}_{p^2}$ is $(2,2)$-isogenous to the Jacobian of a hyperelliptic curve $\mathcal{C}_\alpha/\mathbb{F}_p$ with defining equation given by Costello [125, Prop. 1]. To the Jacobian $\mathcal{J}_\alpha$ of $\mathcal{C}_\alpha$, we can associate a general Kummer surface $\mathcal{K}_\alpha^{\mathrm{gen}}$. As we prefer to work with the squared Kummer model, the next proposition follows Costello [125, §5] to present the isomorphism that maps $\mathcal{C}_\alpha$ to a curve in Rosenhain form. We exploit the fact that, by construction, $\mathcal{C}_\alpha$ has 6 $\mathbb{F}_p$-rational Weierstrass points. We emphasise that we require that the Jacobian $\mathcal{J}_\alpha$ is superspecial.

Proposition 2 (§5, [125])**.** Consider the hyperelliptic curve $\mathcal{C}_\alpha/\mathbb{F}_p$ of genus 2 with superspecial Jacobian J_α, and let $\beta = \beta_0 + i\beta_1$, $\gamma = \gamma_0 + i\gamma_1 \in \mathbb{F}_{p^2}$ such that $\gamma^2 = \alpha$ and $\beta^2 = (\alpha^2 - 1)/\alpha$. Define four values $\zeta_i \in \mathbb{F}_p$ by

$$\begin{aligned} \zeta_1 &= \beta_0\gamma_1 + \beta_1\gamma_0, \quad \zeta_2 = \beta_0\gamma_0 + \beta_1\gamma_1, \\ \zeta_3 &= \beta_0\gamma_1 - \beta_1\gamma_0, \quad \zeta_4 = \beta_0\gamma_0 - \beta_1\gamma_1. \end{aligned}$$

Then $\mathcal{C}_\alpha$ is isomorphic over $\mathbb{F}_p$ to $\mathcal{C}_{\lambda,\mu,\nu}$ where

$$\lambda = -\frac{\zeta_1 \cdot \zeta_2}{\zeta_3 \cdot \zeta_4}, \quad \mu = \frac{\zeta_2 \cdot \zeta_4}{\zeta_1 \cdot \zeta_3}, \quad \nu = -\left(\frac{\zeta_2}{\zeta_3}\right)^2.$$

One can verify that $\nu = \lambda\mu$ and hence, by Lemma 1, $\mathcal{K}_{\lambda,\mu,\nu}$ is elliptic. By Section 2.7, the isomorphism $\kappa : \mathcal{C}_\alpha \to \mathcal{C}_{\lambda,\mu,\nu}$ will induce an isomorphism $\kappa_{**} : \mathcal{K}_\alpha^{\text{gen}} \to \mathcal{K}_{\lambda,\mu,\nu}^{\text{gen}}$. Composing this with $\mathbf{M} : \mathcal{K}_{\lambda,\mu,\nu}^{\text{gen}} \to \mathcal{K}_{\lambda,\mu,\nu}^{\text{Sqr}}$, we obtain a map $\mathcal{K}_\alpha^{\text{gen}} \to \mathcal{K}_{\lambda,\mu,\nu}^{\text{Sqr}}$. The theta constants can be computed using Equation (18) combined with γ_0 and γ_1 as

$$\mu_1 = \sqrt{\lambda} \cdot \left(\frac{\gamma_0^2 - \gamma_1^2}{\gamma_0^2 + \gamma_1^2} \right), \quad \mu_2 = \mu_1/\lambda, \quad \mu_3 = \mu_4 = 1. \tag{24}$$

We refer to this elliptic Kummer surface $\mathcal{K}_{\lambda,\mu,\lambda\mu}^{\text{Sqr}}$ as the "Kummer surfaces associated to E_α" through Scholten's construction, and denote $\mathcal{K}_\alpha := \mathcal{K}_{\lambda,\mu,\lambda\mu}^{\text{Sqr}}$ to make the association explicit.

Mapping Points from E_α to $\mathcal{K}_\alpha$.

Explicit maps between the Weil restriction of E_α and $\mathcal{J}_\alpha$ are originally given by Bernstein and Lange [40]. We use the formulation by Costello [125, §3], giving a $(2,2)$-isogeny $\eta : W_\alpha(\mathbb{F}_p) \to \mathcal{J}_\alpha(\mathbb{F}_p)$ as a composition of several maps, with the map $E_\alpha(\mathbb{F}_{p^2}) \to W_\alpha(\mathbb{F}_p)$ implicitly assumed. By extending η with the map $\rho^{\text{Sqr}} \circ \kappa_\star$ from Section 2.7 to get $\bar{\eta} := \rho^{\text{Sqr}} \circ \kappa_\star \circ \eta$, we can map points $P \in E_\alpha(\mathbb{F}_{p^2})$ to $\bar{\eta}(P) \in \mathcal{K}_\alpha(\mathbb{F}_p)$. We summarise this in Figure 21.

2.10 ELLIPTIC TWISTS OF ELLIPTIC KUMMER SURFACES

We briefly discuss elliptic Kummer surfaces arising from an elliptic curve E_α and its twist $E_{-\alpha}$ as this influences our choice of map η (i.e., what constant e we choose) from Section 2.7. To the best of our knowledge, the discussion on twists in this section does not appear in previous literature.

Definition 3. The *elliptic twist* of a squared Kummer surface $\mathcal{K}^{\text{Sqr}}$, denoted $\mathcal{K}^T$ is defined as the surface with theta coordinates $(-\mu_1, -\mu_2, \mu_3, \mu_4)$, where μ_i are the theta coordinates of $\mathcal{K}^{\text{Sqr}}$.

We remark that $\mathcal{K}^T$ is always isomorphic to $\mathcal{K}^{\text{Sqr}}$ over $\mathbb{F}_p$ using the isomorphism $\Omega : (X : Y : Z : T) \mapsto (-X : -Y : Z : T)$. The values F, G, H that define $\mathcal{K}^{\text{Sqr}}$ change to $F, -G, H$ for $\mathcal{K}^T$. The name *elliptic twist* is justified by the following lemma.

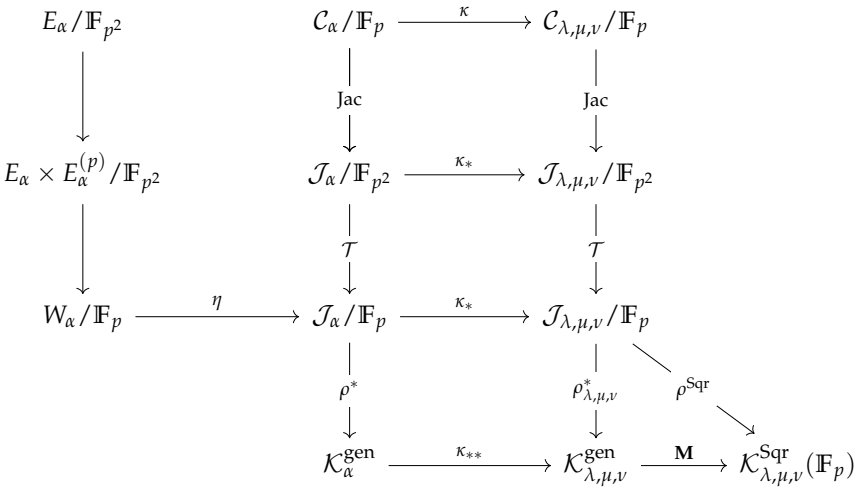
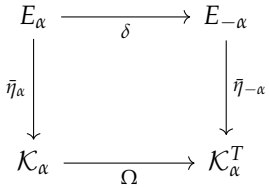


Figure 21: The maps involved when finding the elliptic Kummer surface defined over $\mathbb{F}_p$ associated to an elliptic curve defined over $\mathbb{F}_{p^2}$, including the maps between Kummers described in this section.

Lemma 4. Let $\mathcal{K}_\alpha/\mathbb{F}_p$ be the elliptic Kummer surface associated to $E_\alpha/\mathbb{F}_{p^2}$. Then $\mathcal{K}_\alpha^T/\mathbb{F}_p$ is the elliptic Kummer surface associated to $E_{-\alpha}$, the twist of E_α. In other words, the following square commutes.



Proof. The twist map $\delta : (x, y) \mapsto (-x, iy)$ maps E_α to $E_{-\alpha}$. We want to show that, for $P_{\pm\alpha} \in E_{\pm\alpha}(\mathbb{F}_{p^2})$,

$$(\rho^{\mathrm{Sqr}} \circ \kappa_* \circ \eta_{-\alpha} \circ \delta)(P_\alpha) = (\Omega \circ \rho^{\mathrm{Sqr}} \circ \kappa_* \circ \eta_\alpha)(P_\alpha).$$

By direct calculation $\eta_{-\alpha}(P_{-\alpha}) = (\eta_\alpha \circ \delta)(P_\alpha)$. Therefore, it suffices to show that for divisors $D_{\pm\alpha} \in \mathcal{J}_{\pm\alpha}(\mathbb{F}_p)$, we have $\Omega \circ \rho^{\mathrm{Sqr}}(D_\alpha) = \rho^{\mathrm{Sqr}}(D_{-\alpha})$.

To show this, we observe that, if $\alpha \mapsto -\alpha$, we have $\beta \mapsto i\beta$ and $\gamma \mapsto i\gamma$. Writing $\beta = \beta_0 + i\beta_1$ and $\gamma = \gamma_0 + i\gamma_1$ we have $\beta_0 \mapsto \beta_1, \beta_1 \mapsto -\beta_0$

and similar for γ_0, γ_1. Thus, using Proposition 2, to compute the Rosenhain invariants λ, μ, ν, we find these are invariant under these changes to $\beta_i$and γ_i. Then, by Equation (24), we get that $\mu_1 \mapsto -\mu_1$ and $\mu_2 \mapsto -\mu_2$[6]. Thus, $\rho^{\text{Sqr}} : \mathcal{J}_\alpha(\mathbb{F}_p) \to \mathcal{K}_\alpha$ maps a divisor $D_\alpha \in \mathcal{J}_\alpha(\mathbb{F}_p)$ to $(X_1 : X_2 : X_3 : X_4)$ and $\rho^{\text{Sqr}} : \mathcal{J}_{-\alpha}(\mathbb{F}_p) \to \mathcal{K}_{-\alpha}$ maps $D_{-\alpha} \in \mathcal{J}_{-\alpha}(\mathbb{F}_p)$ to $(-X_1 : -X_2 : X_3 : X_4)$. □

The proof of Lemma 4 shows that both $\mathcal{C}_\alpha$ and $\mathcal{C}_{-\alpha}$ map to the same Rosenhain curve $\mathcal{C}_{\lambda,\mu,\nu}$. However, the choice of constants μ_i defining the Kummer surface $\mathcal{K}_\alpha$ not only depends on λ, μ, ν, but also on the concrete values of β_i and γ_i. This choice impacts the design slightly: the map $\kappa : \mathcal{C}_\alpha \to \mathcal{C}_{\lambda,\mu,\nu}$ is given by five values (a, b, c, d, e). The constants a, b, c, d can be computed over $\mathbb{F}_p$, however, e is defined as the square root of $e^2 \in \mathbb{F}_p$, and therefore lies in $\mathbb{F}_p$ only half the time. Whenever $e \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$, the map κ is only defined over $\mathbb{F}_{p^2}$, even though the curves themselves are isomorphic over $\mathbb{F}_p$. As e only determines the y-coordinate of the image, this does not affect the composition map $\bar{\eta} : E_\alpha \to \mathcal{K}_\alpha$, however it impacts the difficulty of implementation and efficiency of $\bar{\eta}$. Fortunately, the following result shows that we can always avoid this.

Lemma 5. Denote by $\kappa : \mathcal{C}_\alpha \to \mathcal{C}_{\lambda,\mu,\nu}$ the map given by (a, b, c, d, e) and by $\kappa^T : \mathcal{C}_{-\alpha} \to \mathcal{C}_{\lambda,\mu,\nu}$ the analogous map for the elliptic twist given by $(a^T, b^T, c^T, d^T, e^T)$. Then, either κ or κ^T is defined over $\mathbb{F}_p$. In other words, $e \in \mathbb{F}_p$ if and only if $e^T \notin \mathbb{F}_p$, and vice versa.

Proof. As both (a, b, c, d) and (a^T, b^T, c^T, d^T) are always defined over $\mathbb{F}_p$ by definition, we only need to show that e^2 and $(e^T)^2$ differ by a non-square in $\mathbb{F}_p$. Direct computation shows that

$$e^2 = -\left(\frac{\gamma_0}{\gamma_1}\right)^6 \cdot e^{T2},$$

where $-\left(\frac{\gamma_0}{\gamma_1}\right)^6$ is a non-square [7]. This ensures that precisely one of e^2 or e^{T2} has a square root over $\mathbb{F}_p$. □

6 For a Magma version of this proof, see TwistProof.m.
7 See TwistProof.m for a proof of this in MAGMA.

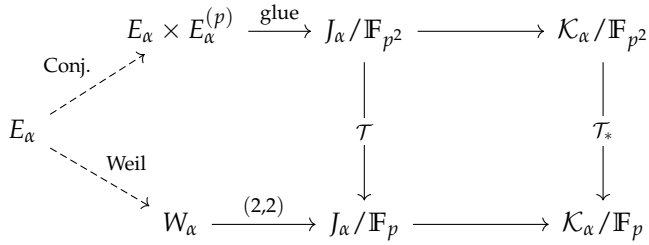


Figure 22: A clearer picture of the Weil restriction maps

2.11 SCHOLTEN'S CONSTRUCTION IS GLUING

Let $\mathcal{J}_\alpha$ be as in Section 2.5. The constructed $\mathcal{J}_\alpha$ is $(2,2)$-isogenous *over* $\mathbb{F}_{p^2}$ to the product of elliptic curves $E_\alpha \times E_\alpha^{(p)}$, where $E_\alpha^{(p)}$ is the Frobenius conjugate, e.g., there exists a $(2,2)$-isogeny $\varphi_\alpha : \mathcal{J}_\alpha \to E_\alpha \times E_\alpha^{(p)}$. Such an isogeny is often called a *splitting* and its dual $\hat{\varphi}_\alpha$ is the *gluing* of E_α and $E_\alpha^{(p)}$ over some two-torsion. Importantly, the projections $E_\alpha \times E_\alpha^{(p)} \to E_\alpha$ and $E_\alpha \times E_\alpha^{(p)} \to E_\alpha^{(p)}$ are not defined over $\mathbb{F}_p$. We can frame Scholtens construction similarly in the language of gluing and splitting, visualised in Figure 22.

Lemma 6. Let $\alpha = \alpha_0 + \alpha_1 i \in \mathbb{F}_{p^2}$ with $\alpha_0, \alpha_1 \neq 0$. Then the Weil restriction W_α is $(2,2)$-isogenous over $\mathbb{F}_p$ to some $\mathcal{J}_\alpha/\mathbb{F}_p$. For the right gluing, $E_\alpha \times E_\alpha^{(p)}$ is $(2,2)$-isogenous to the extension of this $\mathcal{J}_\alpha$ over $\mathbb{F}_{p^2}$.

§3 USING PAIRINGS ON KUMMER SURFACES

Pairings on abelian varieties have proven to be essential in the construction and cryptanalysis of many cryptographic primitives [32, 208, 226]. Most relevant to this article is their use in isogeny-based cryptography, in particular recent work [121, 130, 258] that shows how degree-2 Tate pairings can be used to deterministically sample specific 2^n-torsion points on elliptic curves. The aim of this section is to generalise this result to genus 2, and in particular Kumemr surfaces, in order to enable efficient point sampling and point compression. To this end, we introduce the general theory of generalized Tate pairings in Section 3.1, and apply this to Jacobians in genus 2 in Section 3.2, which

allows us to generalise Theorem IX.4 to genus 2 and more specifically Kummer surfaces. We then show how to concretely compute Tate pairings of degree 2 in Section 3.4.

We remark that the possible applications of generalised Tate pairings to study the image of isogenies spreads much wider than SQIsign, or even isogeny-based cryptography.[8]

3.1 DESCRIBING THE IMAGE OF ISOGENIES USING PAIRINGS

To generalise elliptic curve techniques for deterministic point sampling to Kummer surfaces, we first describe a general method to decompose the cokernel $A/\operatorname{Im}\hat{\varphi}$ of $\hat{\varphi}$ in terms of $\ker\varphi$ for any isogeny $\varphi : A \to B$ between abelian varieties. A similar interpretation of the Tate pairing was independently given by Robert [333]. We then apply this technique to decompose $[2]\mathcal{J}_{\mathcal{C}}$, resp. $[2]\mathcal{K}_{\mathcal{C}}$, which allows us to sample points with improved precision in $\mathcal{J}_{\mathcal{C}} \setminus [2]\mathcal{J}_{\mathcal{C}}$.

IN DIMENSION 1. Before stating the general theorem, let us recall the following classical result in dimension 1 on the stucture of $[2]E$.

Theorem 7 (Thm. 4.1, [212])**.** Let $E/k : y^2 = (x-\lambda_1)(x-\lambda_2)(x-\lambda_3)$ be an elliptic curve. Then $P \in [2]E$ if and only if

$$(x-\lambda_1), (x-\lambda_2) \text{ and } (x-\lambda_3) \text{ are all squares.}$$

In Theorem IX.4, we rephrased and specialised this result in terms of reduced Tate pairings: A point $P \in E$ is in $[2]E$ if and only if the reduced Tate pairing with all three 2-torsion points $(\lambda_i, 0)$ is trivial. Furthermore, points $P \in E \setminus [2]E$ lie above $L_i = (\lambda_i, 0)$ if $t_2(L_i, P) = 1$ and $t_2(L_j, P) = -1$ for $j \neq i$.

Generalised Tate Pairings.

The above result is a particular instantiation of a much more general result on *generalised Tate pairings*, associated to any isogeny $\varphi : A \to B$ between abelian varieties. We first sketch this general framework, and detail how the dimension 1 example is a specific case, before applying it in the dimension 2 setting.

8 Even pre-quantum elliptic-curve subgroup membership testing [243] can be rewritten in this language.

As shown by Bruin [74], to any separable isogeny $\varphi : A \to B$ between abelian varieties over $\mathbb{F}_q$ such that the kernel of $\hat{\varphi}$ is annihilated by $[q-1]$, we can associate a perfect pairing

$$t_\varphi : \ker \varphi \times \operatorname{coker} \hat{\varphi} \to \mathbb{F}_q^*.$$

We refer to t_φ as the *generalised φ-Tate pairing*. This pairing has been studied in the context of cryptography [94] for elliptic curves.

As Robert [333] notes, perfectness implies that we can precisely identify $\operatorname{Im} \hat{\varphi}$ using $\ker \varphi$ and this pairing, in the following sense.

Lemma 8 (Cor. 5.2, [333])**.** Let t_φ denote the (reduced) φ-Tate pairing. Then

$$Q \in \operatorname{Im} \hat{\varphi} \quad \Leftrightarrow \quad t_\varphi(P, Q) = 1 \text{ for all } P \in \ker \varphi.$$

Beyond this, we can decompose the cosets of $\operatorname{coker} \hat{\varphi} = A/\operatorname{Im} \hat{\varphi}$ using the *profile* of a point $Q \in A$.

Definition 9. Let t_φ denote the (reduced) φ-Tate pairing and let $Q \in A$. The *profile* of Q is the array of evaluations of $t_\varphi(P, Q)$ in all points $P \in \ker \varphi$. We denote the profile of Q with respect to t_φ by $t_{\ker \varphi}(Q)$.

By Lemma 8, the generalised φ-Tate pairing is constant on each coset, so the profile of a point Q determines precisely in which coset of $A/\operatorname{Im} \hat{\varphi}$ it lies. By bilinearity of the Tate pairing, the profile of $Q + Q'$ is the pointwise multiplication of their profiles. Furthermore, any basis of $\ker \varphi$ is enough to determine the full profile, and we therefore use the smaller array of evaluations $(t_\varphi(P_i, Q))_i$ for P_i a basis of $\ker \varphi$.

Example 1. When we take $\varphi = [2]$ to be the multiplication-by-2 map on an elliptic curve E, we recover the dimension 1 result of Theorem IX.4 with a much more elegant proof. If, instead, we take $\varphi : E \to E/\langle L_i \rangle$, with $L_i \in E[2] \setminus \mathcal{O}_E$, we have that $\operatorname{coker} \hat{\varphi}$ consists precisely of two cosets: $\mathcal{O} + [2]E$ and $L_i + [2]E$. Then $t_\varphi(L_i, P) = 1$ if and only if P is above L_i, for $P \notin [2]E$.

As $t_{\ker \varphi}$ gives information only with respect to the smaller set $\ker \varphi$ about the coarser cosets $A/\operatorname{Im} \hat{\varphi}$, we see that $t_{\ker \varphi}$ gives a subset of information of $t_{\ker[\deg \varphi]}$. However, this information is given with fewer computations, and may in some settings give enough information. For example, the technique, used in SIDH/SIKE, CSIDH and SQIsign, to sample points on Montgomery curves *not above* $(0,0)$ with order divisible by 2^f by sampling a non-square x-coordinate x_P can be rephrased as sampling points in $E \setminus \operatorname{Im} \hat{\varphi}$, where $\varphi : E \to E/\langle (0,0) \rangle$,

given that the reduced Tate pairing $t_\varphi((0,0), P)$ is exactly the Legendre symbol of x_P.

The cokernel as a group.

The group structure of coker φ for a separable d-isogeny φ with d prime and kernel of dimension n is the same as that of the kernel: both are isomorphic to μ_d^n. Assuming μ_d is rational, both are isomorphic to $(\mathbb{Z}/d\mathbb{Z})^n$ by some choice of primitive root $\zeta_d \in \mathbb{F}_q$. Using $t_{\ker\varphi}$, this result becomes intuitive.

Lemma 10. For principally polarised abelian varieties A, B over $\mathbb{F}_q$, let $\varphi : A \to B$ be a separable d-isogeny with d prime and rational kernel generators, such that the generalised Tate pairing t_φ is perfect. Then

$$\ker\varphi \xrightarrow{\sim} \operatorname{coker}\hat{\varphi}(\mathbb{F}_q) \xrightarrow{\sim} \mu_d^n,$$

where μ_d are the d-th roots of unity in $\mathbb{F}_q$.

Proof. Per definition of φ, its kernel is a $\mathbb{Z}$-module of rank n. Let $K_1, \ldots, K_n \in A(\mathbb{F}_q)$ be a basis of $\ker\varphi$. The map

$$t_{\ker\varphi} : P \mapsto t_\varphi(K_1, P), \ldots, t_\varphi(K_g, P)$$

gives a surjective map $A(\mathbb{F}_q) \to \mu_d^n$, with kernel $\operatorname{Im}\hat{\varphi}(\mathbb{F}_q)$ [333, Cor. 5.2]. Hence, the map induces an isomorphism $A(\mathbb{F}_q)/\operatorname{Im}\hat{\varphi}(\mathbb{F}_q) = \operatorname{coker}\hat{\varphi}(\mathbb{F}_q) \xrightarrow{\sim} \mu_d^n$. □

This result is used in some cases in pairing-based cryptography, where the Tate pairing of level n can sometimes be viewed as a pairing $E[n] \times E[n] \to \mathbb{F}_q^*$, and in general is useful in practical applications of profiles of generalised Tate pairings, as we see in next sections.

3.2 DECOMPOSING THE JACOBIAN USING PROFILES

This section uses the Tate pairing for the multiplication-by-2 map $[2]$ on $\mathcal{J}_\mathcal{C}$ to decompose $\mathcal{J}_\mathcal{C}$ into cosets $\mathcal{J}_\mathcal{C}/[2]\mathcal{J}_\mathcal{C}$, following the general theory developed in Section 3.1. In contrast to elliptic curves, for our dimension-2 setting we want to identify points in a precise coset, rather than simply identifying points in $\mathcal{J}_\mathcal{C} \setminus [2]\mathcal{J}_\mathcal{C}$. For the rest of this work, we assume Jacobians whose complete 2-torsion is rational, so that Lemma 10 applies over the base field.

We demonstrate how Theorem IX.4 can be generalised to dimension 2, or more generally to any dimension. This aligns with a description of $[2]\mathcal{J}_\mathcal{C}$

sketched by Cassels and Flynn [84, Ch. 10]. We explore in more detail how to identify points in the cosets of $\mathcal{J}_\mathcal{C}/[2]\mathcal{J}_\mathcal{C}$ using the profile $t_{\ker[2]}(Q)$ of Q. We then show how to compute these pairing values on $\mathcal{K}_\mathcal{C}$ too, both for the general and squared models, to get an analogous result for Kummer surfaces. As a direct consequence of Lemma 8, we get a first generalistaion of Theorem 7.

Theorem 11. Let $P \in \mathcal{J}_\mathcal{C}$ and let $\{D_{i,j}\}_{1\leq i<j\leq 6}$ denote the fifteen points of order 2. Then $P \in [2]\mathcal{J}_\mathcal{C}$ if and only if $t_{\ker[2]}(P)$ is trivial, e.g.

$$t_2(D_{i,j}, P) = 1 \quad \text{for all } 1 \leq i < j \leq 6.$$

By bilinearity of the Tate pairing, given a basis $B_1, \ldots, B_4$ for $\mathcal{J}_\mathcal{C}[2]$ and writing $D_{i,j} = a \cdot B_1 + b \cdot B_2 + c \cdot B_3 + d \cdot B_4$ for $a, b, c, d \in \{0, 1\}$, we can compute $t_2(D_{i,j}, P)$ in terms of the four Tate pairings $t_2(B_i, P)$ as

$$t_2(D_{i,j}, P) = t_2(B_1, P)^a \cdot t_2(B_2, P)^b \cdot t_2(B_3, P)^c \cdot t_2(B_4, P)^d.$$

and thus we will use a more succing form of Theorem 11 by $P \in [2]\mathcal{J}_\mathcal{C}$ if and only if $t_2(B_i, P) = 1$ for all $1 \leq i \leq 4$. Hence, we can quickly identify points in $\mathcal{J}_\mathcal{C} \setminus [2]\mathcal{J}_\mathcal{C}$, by sampling a random point P until $t_2(B_i, P) = -1$ for some $1 \leq i \leq 4$.

Remark 4. Let f be such that 2^f is the maximal power-of-two torsion on $\mathcal{J}_\mathcal{C}(\mathbb{F}_p)$. It is tempting to think that $P \in \mathcal{J}_\mathcal{C} \setminus [2]\mathcal{J}_\mathcal{C}$ if and only if the order of P is divisible by 2^f. However, this implication only works in one direction: Any P with order divisible by 2^f cannot be in $[2]\mathcal{J}_\mathcal{C}(\mathbb{F}_p)$ as this contradicts the maximality of f. Points in $\mathcal{J}_\mathcal{C} \setminus [2]\mathcal{J}_\mathcal{C}$ do not necessarily have an order divisible by 2^f.

3.3 DECOMPOSING ELLIPTIC JACOBIANS

Consider the superspecial Jacobian $\mathcal{J}_\alpha$ that is $(2,2)$-isogenous to the Weil restriction of a supersingular E_α, and thus, has Rosenhain invariants λ, μ, ν such that $\lambda \cdot \mu = \nu$. Let $m = \frac{p+1}{2}$ and let 2^f be the largest power of 2 dividing m. Then, as a group, we find

$$\mathcal{J}_\alpha(\mathbb{F}_p) \cong \mathbb{Z}_m \times \mathbb{Z}_m \times \mathbb{Z}_2 \times \mathbb{Z}_2.$$

In the following paragraphs, we show how profiles $t_{\ker[2]}(P)$ identify points $P \in \mathbb{Z}_m \times \mathbb{Z}_m$ whose order is divisible by 2^f, e.g. have maximal power-of-two

torsion. This is in general useful for isogeny-based cryptography, and we will rely heavily on this fact in later sections.

Decomposition into cosets.

Let $\mathcal{B} = (P_1, P_2, P_3, P_4)$ be a basis for $\mathcal{J}_\mathcal{C}$ such that P_1 and P_2 generate the subgroup $\mathbb{Z}_m \times \mathbb{Z}_m$, and P_3 and P_4 generates $\mathbb{Z}_2 \times \mathbb{Z}_2$. Theorem 11 gives us that $P \in [2]\mathcal{J}_\mathcal{C}$ if and only if $t_{\ker[2]}(P)$ is trivial. We find that $\mathcal{J}_\mathcal{C}/[2]\mathcal{J}_\mathcal{C}$ decomposes into 16 cosets

$$aP_1 + bP_2 + cP_3 + dP_4 + [2]\mathcal{J}_\mathcal{C}, \quad a, b, c, d \in \{0, 1\}$$

with the trivial coset $[2]\mathcal{J}_\mathcal{C}$ given by $a = b = c = d = 0$, and so the group structure of $\mathcal{J}_\mathcal{C}/[2]\mathcal{J}_\mathcal{C}$ is isomorphic to $(\mathbb{Z}/2\mathbb{Z})^4$. As the profile of points in a coset is constant given the basis $\mathcal{B}$, we can write $t_{\mathcal{B},[2]}(a, b, c, d)$ for the profile associated to the coset $aP_1 + bP_2 + cP_3 + dP_4 + [2]\mathcal{J}_\mathcal{C}$. We compute $t_{\mathcal{B},[2]}(a, b, c, d)$ simply as $t_{\ker[2]}(Q)$ for $Q = aP_1 + bP_2 + cP_3 + dP_4$, given $\mathcal{B} = (P_1, \ldots, P_4)$. By bilinearity of the Tate pairing, addition of these profiles is well-defined.

The three cosets, $P_3 + [2]\mathcal{J}_\mathcal{C}$, $P_4 + [2]\mathcal{J}_\mathcal{C}$ and $P_3 + P_4 + [2]\mathcal{J}_\mathcal{C}$ contain precisely all points of $\mathcal{J}_\mathcal{C}$ whose order is divisible by 2^{f-1}, besides the trivial coset. Hence, points with order divisble by at most 2^{f-1} have profiles $t_{\mathcal{B},[2]}(0, 0, c, d)$ with $c, d \in \{0, 1\}$.

All 12 other cosets therefore *only* contain points whose order is divisible by 2^f. In particular, the points in $\mathbb{Z}_m \times \mathbb{Z}_m$ whose order is divisble by 2^f are given precisely by the cosets $P_1 + [2]\mathcal{J}_\mathcal{C}$, $P_2 + [2]\mathcal{J}_\mathcal{C}$ and $P_1 + P_2 + [2]\mathcal{J}_\mathcal{C}$. That is, they are identified by profiles $t_{\mathcal{B},[2]}(a, b, 0, 0)$ with $a, b \in \{0, 1\}$. Summarizing, we generalize Theorem 11 as follows.

Theorem 12. Let $P \in \mathcal{J}_\mathcal{C}$. Let $\mathcal{B} = (P_1, P_2, P_3, P_4)$ be a basis of $\mathcal{J}_\mathcal{C}$ as given above. Let $t_{\mathcal{B},[2]}(a, b, c, d) := t_{\ker[2]}(aP_1 + bP_2 + cP_3 + dP_4)$. Then we get

$$P \in [2]\mathcal{J}_\mathcal{C} \quad \Leftrightarrow \quad t_{\ker[2]}(P) = t_{\mathcal{B},[2]}(0, 0, 0, 0),$$

and

$$2^f \mid \operatorname{ord}(P) \quad \Leftrightarrow \quad t_{\ker[2]}(P) \neq t_{\mathcal{B},[2]}(0, 0, c, d).$$

Furthermore, $P \in \langle P_1, P_2 \rangle$ with order divisible by 2^f if and only if

$$t_{\ker[2]}(P) = t_{\mathcal{B},[2]}(a, b, 0, 0), \quad \text{with not both } a, b = 0.$$

It is easy to see that similar theorems can be derived for general Tate pairings t_φ using $t_{\ker\varphi}$ to decompose coker φ.

3.4 COMPUTING PAIRINGS OF DEGREE 2 IN DIMENSION 2

The general theory to compute pairings on Jacobians of hyperelliptic curves is well-developed and a good overview is given by Galbraith, Hess, and Vercauteren [184]. In this section, we compute pairings of degree 2 on Jacobians and Kummers with fully rational 2-torsion.

Computing 2-Pairings On Jacobians.

Let $D_{i,j} = (w_i, 0) + (w_j, 0) \in \mathcal{J}_\mathcal{C}[2] \setminus \{\mathcal{O}\}$ of order 2 where w_i, w_j are Weierstrass values, and let $D_P = (x_1, y_1) + (x_2, y_2)$ be any element of $\mathcal{J}_\mathcal{C}(\mathbb{F}_p)$. Whenever $D_{i,j}$ and D_P are coprime (i.e., have disjoint support), the unreduced Tate pairing $T_2(D_{i,j}, D_P)$ is computed as

$$T_2(D_{i,j}, D_P) = (w_i - x_1)(w_i - x_2)(w_j - x_1)(w_j - x_2). \tag{25}$$

Using resultants, we can express this computation in terms of the Mumford representations of $D_{i,j}$ and D_P, ensuring computations can stay over the base field. Let k denote the embedding degree, then we get the *reduced* Tate pairing $t_2(D_{i,j}, D_P)$ by exponentiating $T_2(D_{i,j}, D_P)$ by $\frac{p^k-1}{2}$. In our case, as [2] has embedding degree $k = 1$, this coincides with the Legendre symbol of the Tate pairing. When $D_{i,j}$ and D_P are not coprime, we take any random element $S \in \mathcal{J}_\mathcal{C}$ such that $D_{i,j}$ is coprime with both S and $D_P + S$. which allows us to compute $t_2(D_{i,j}, D_P)$ as

$$t_2(D_{i,j}, D_P) = t_2(D_{i,j}, D_P + S)/t_2(D_{i,j}, S).$$

Computing 2-Pairings On General Kummers.

The 2-torsion point $L_{i,j} \in \mathcal{K}^{\text{gen}}[2] \setminus \{\mathbf{o}\}$ is given as the image of $D_{i,j} = (w_i, 0) + (w_j, 0) \in \mathcal{J}_\mathcal{C}$, with w_i, w_j Weierstrass values and $i < j$. Thus, we can write $L_{i,j}$ as $(1 : l_2^{(i,j)} : l_3^{(i,j)} : l_4^{(i,j)}) \in \mathcal{K}^{\text{gen}}$ with

$$l_2^{(i,j)} = w_i + w_j, \quad l_3^{(i,j)} = w_i \cdot w_j,$$

and where $l_4^{(i,j)}$ can be derived from l_2 and l_3. Given a second, coprime, Kummer point $P = (1 : k_2 : k_3 : k_4)$, we can write the Tate pairing $t_2(L_{i,j}, P)$ for $i, j > 1$ in terms of the l_i and k_i as

$$t_2(L_{i,j}, P) = l_3^2 + k_3^2 - k_2 l_2 l_3 - k_2 k_3 l_2 + k_2^2 l_3 + k_3 l_2^2 - 2 \cdot k_3 l_3. \tag{26}$$

Whenever $i = 1$, the point $L_{1,j}$ has the form $(0 : 1 : w_j : w_j^2)$ and one can compute similar formulas[9]. Whenever $L_{i,j}$ and P are not co-prime, we add some element $L_{i',j'} \in \mathcal{K}^{\text{gen}}[2]$ to P to obtain the required co-primality.[10]

Computing 2-Pairings On Squared Kummers.

On squared Kummers, there are three distinct methods to compute the degree-2 Tate pairing: (a) follow a similar method to that used for general Kummer surfaces; (b) follow the monodromy approach due to Robert [331]; and (c) map the Kummer points to partial Jacobian elements and compute the pairing on the Jacobian.

(A) USING SQUARED KUMMER COORDINATES. The approach used for general Kummers also works on the squared Kummer surface. Due to the more difficult map ρ^{Sqr} from $\mathcal{J}_{\lambda,\mu,\nu}$ to $\mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$, we require the use of additional theta functions, beyond the four coordinate functions X_1, X_2, X_3, X_4, to compute such pairings. Specifically, we require the coordinate functions X_8 and X_{10}, as given by Gaudry [192], to compute all possible Tate pairings of degree 2. However, due to the need for additional theta functions, the above approach is neither efficient nor elegant.

(B) MONODROMY APPROACH BY ROBERT. An alternative approach to compute such a Tate pairing on Kummer surfaces is sketched by Lubicz and Robert [262]. To compute $t_2(L_{i,j}, Q)$, we require the matrix $W_{i,j}$ representing the translation-by-$L_{i,j}$ map. For elliptic Kummers, these are given in Section A. These computations are explained more generally by Robert [331]; we apply the Algorithm 5.2 to Kummer surfaces of genus-2 hyperelliptic curves. See Section B for a detailed explanation.

9 Alternatively, one can find suitable $L_{i',j'}$ to compute the pairing as $t_2(L_{i,j} + L_{i',j'}, P)/t_2(L_{i',j'}, P)$, where the above formula can be applied as long as both $L_{i,j} + L_{i',j'}$ and $L_{i',j'}$ are of the required form.

10 As discussed throughout Section 2, the action of a 2-torsion point $L_{i,j}$ on $\mathcal{K}$ is well-defined and given by a matrix $W_{i,j}$.

(c) USING THE EFFICIENT SEMI-INVERSE MAP. The last method uses the efficient map to recover u_0 and u_1 for a point P as given by Equation (21). The values u_0 and u_1 allow us to compute the left-side of the Mumford representation $a(x) = x^2 + u_1 x + u_0$. All pairings $t_2(L_i, P)$ can then be computed as the resultant of $a(x)$ with the Mumford coordinate $(x - w_i)(x - w_j)$ of $D_{i,j}$. This has the additional advantage that many values can be re-used to compute multiple pairings $t_2(D_{i,j}, P)$ for the same point P, which we heavily rely on in later sections to compute the profile $t_{\ker[2]}(P)$.

§4 ALGORITHMS FOR KUMMER-BASED CRYPTOGRAPHY

The aim of this section is to introduce several tools required to transport isogeny-based cryptography to Kummer surfaces. Throughout this section, the tools we develop are largely motivated by our showcasing example: compressed SQIsign verification between Kummer surfaces. To develop an efficient point compression algorithm in Section 4.4, we require the following tools.

- For many of our algorithms, we often need that two points $P, Q \in \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ originate both from the Jacobian $\mathcal{J}_{\lambda,\mu,\nu}$ or both from its twist. Thus, we require the algorithm CheckOrigin, which allows us to ensure $\mathsf{CheckOrigin}(\mathcal{K}, \mathbf{o}, P) = \mathsf{CheckOrigin}(\mathcal{K}, \mathbf{o}, Q)$.

- To compress the point K defining a $(2^f, 2^f)$-isogeny in Section 5, we need to write K as $P + [s]Q$ for some deterministically sampled $P, Q \in \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}[2^f]$ and some scalar $s \in \{1, \ldots, 2^f\}$. In Section 4.4, we develop an algorithm to deterministically sample P, Q using pairings on Kummer surfaces.

- To decompress P, Q, s and compute the kernel generator K we require a three point ladder – `3ptLadder` – which takes as input $P, Q, P - Q$ and s. The point difference, $P - Q$, will not be known a priori, and so must be computed using PointDifference.

We emphasise that these algorithms are more widely applicable beyond the scope of this work and make a step towards making isogeny-based cryptography using Kummer surfaces practical.

4.1 IDENTIFYING TWIST POINTS

The rational points on the Kummer surface $\mathcal{K}_{\lambda,\mu,\nu}(\mathbb{F}_p)$ consist of the points originally coming from $\mathcal{J}_{\lambda,\mu,\nu}(\mathbb{F}_p)$ as well as from its twist $\mathcal{J}^T_{\lambda,\mu,\nu}(\mathbb{F}_p)$[11]. In other words, the elements

$$D = \langle x^2 + u_1 x + u_0, i \cdot (v_1 x + v_0) \rangle \in \mathcal{J}_{\lambda,\mu,\nu}(\mathbb{F}_{p^2})$$

also map to $\mathbb{F}_p$-rational points $P \in \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}(\mathbb{F}_p)$.

Recognizing if a random point $P \in \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}(\mathbb{F}_p)$ is an image point of the original Jacobian $\mathcal{J}_{\lambda,\mu,\nu}(\mathbb{F}_p)$ or its twist is essential for many steps in later algorithms. For example, the algorithm PointDifference, which computes $P \pm Q$ given $P, Q \in \mathcal{K}$, requires both P and Q to originate from the same Jacobian in order to return rational points $P \pm Q \in \mathcal{K}(\mathbb{F}_p)$.

The map $(\rho^{\text{Sqr}})^{-1} : \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu} \to \mathcal{J}_{\lambda,\mu,\nu}$, described by polynomials F_0, F_1, F_2 over $\mathbb{F}_p$ in Equations (20) and (21), gives us the criteria we need to recognise the origin of a points on $\mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$, as summarised by the following lemma.

Lemma 13. Let $\mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$ be a squared Kummer surface arising from a hyperelliptic curve $\mathcal{C}_{\lambda,\mu,\nu}$ with Weierstrass points $(w_i, 0)$. A point $P = (X_1 : X_2 : X_3 : X_4) \in \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}(\mathbb{F}_p)$ originates from an element $D \in \mathcal{J}_{\lambda,\mu,\nu}(\mathbb{F}_p)$ if and only if the following value z is a square in $\mathbb{F}_p$:

$$z = u_0(w_3 w_4 - u_0)(w_4 + w_6 + u_1) - \omega X_1 / \mu_1 \in \mathbb{F}_p,$$

with u_0, u_1 as given in Equation (21), and ω in Equation (22). Otherwise, P originates from an element $D \in \mathcal{J}^T_{\lambda,\mu,\nu}(\mathbb{F}_p)$.

Proof. Direct computation shows that the values u_0 and u_1 are such that the pre-image $D_P \in \mathcal{J}(\mathbb{F}_{p^2})$ of P has Mumford representation $\langle x^2 + u_1 x, +u_0, - \rangle$. As D_P maps to $P \in \mathcal{K}(\mathbb{F}_p)$, using Equation (19) we find that u_0, u_1 and v_0^2 must be rational. Hence, either $D_P \in \mathcal{J}(\mathbb{F}_p)$, with $v_0 \in \mathbb{F}_p$, or $D_P \in \mathcal{J}^T(\mathbb{F}_p)$, with $v_0 = i \cdot \alpha$ for some $\alpha \in \mathbb{F}_p$.

11 This is a different twist than the *elliptic twist* defined in Section 2.10. The elliptic twist $\mathcal{K}^T$ originates from twisting the elliptic curve E_α, whereas this twist originates from a twist of the hyperelliptic curve $\mathcal{C}_{\lambda,\mu,\nu}$.

As $z = v_0^2$ (see Equation (23)), when $D_P \in \mathcal{J}(\mathbb{F}_p)$, this is therefore a square in $\mathbb{F}_p$. Conversely, if $D_P \in \mathcal{J}^T(\mathbb{F}_p)$, we find $z = (i\alpha)^2 = -\alpha^2$, which is not a square for $p \equiv 3 \mod 4$. □

We use this lemma to construct the algorithm CheckOrigin. Given a point $P \in \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$, it will output true if $P \in \rho^{\mathrm{Sqr}}(\mathcal{J}_{\lambda,\mu,\nu}(\mathbb{F}_p))$ or false if $P \in \rho^{\mathrm{Sqr}}(\mathcal{J}^T_{\lambda,\mu,\nu}(\mathbb{F}_p))$.

Algorithm 23 CheckOrigin.

Input: A Kummer $\mathcal{K} = \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ and $P = (X_1 : X_2 : X_3 : X_4) \in \mathcal{K}(\mathbb{F}_p)$.
Output: true if $P \in \rho^{\mathrm{Sqr}}(\mathcal{J}_{\lambda,\mu,\nu}(\mathbb{F}_p))$ or false if $P \in \rho^{\mathrm{Sqr}}(\mathcal{J}^T_{\lambda,\mu,\nu}(\mathbb{F}_p))$.

1: Set $\widetilde{X}_i \leftarrow X_i/\mu_i$ for $1 \le i \le 4$.
2: Set $u_0 \leftarrow F_1(\widetilde{X})/F_0(\widetilde{X})$, $u_1 \leftarrow F_2(\widetilde{X})/F_0(\widetilde{X})$, $\omega \leftarrow G(\widetilde{X})/F_0^2(\widetilde{X})$.
3: Compute $z \leftarrow u_0(w_3w_5 - u_0)(w_4 + w_6 + u_1) - \omega\widetilde{X}_1$.
4: **return** `IsSquare` (z)

Remark 5. CheckOrigin is much simpler than the general maps given by theta functions, described by Gaudry [192], to compute the origin. Previous works [63] using the general maps would avoid computing the origin as the computations are too involved, and would simply work their way around such problems. The above algorithm may therefore also improve, for example, the twist security of hyperelliptic curve Diffie–Hellman-style protocols.

4.2 DIFFERENCE OF POINTS

Though we want to do arithmetic on the Kummer surface $\mathcal{K}$ for efficiency, we only have a pseudo-group law. In particular, to compute $P + Q \in \mathcal{K}$ or $P + [s]Q \in \mathcal{K}$, we require the knowledge of P, Q *and* $P - Q \in \mathcal{K}$. For many applications, however, we will only have access to $P, Q \in \mathcal{K}$ a priori. To overcome this, we describe an algorithm PointDifference to compute $P \pm Q \in \mathcal{K}$, given $P, Q \in K$, defined only up to sign.

Let $S = P + Q$ and $D = P - Q$, given on the Kummer surface as $S = (S_1 : S_2 : S_3 : S_4)$ and $D = (D_1 : D_2 : D_3 : D_4)$. We follow the approach by Renes and Smith [326, Prop. 4], using the biquadratic forms $B_{i,j}$ associated to the Kummer surface with the property

$$S_i \cdot D_j + D_i \cdot S_j = \lambda_{ij} B_{i,j}(P, Q),$$

i.e., they are equal up to some projective constant $\lambda_{ij} \in \mathbb{F}_p$. Writing $B_{i,j}$ for $B_{i,j}(P, Q)$, we get six degree-2 forms $f_{i,j}$ in variables x_1, x_2, x_3, and x_4 for $1 \leq i < j \leq 4$ given by

$$f_{i,j}(x_i, x_j) = B_{j,j} \cdot x_i^2 - 2B_{i,j}x_i x_j + B_{i,i}x_j^2,$$

such that $f_{i,j}(X_i, X_j) = 0$ if and only if $R = (X_1 : X_2 : X_3 : X_4)$ corresponds to either S or D. Without loss of generality, we set $X_1 = 1$ and solve each subsequent equation to determine a solution $R = (X_1 : X_2 : X_3 : X_4)$ to this system of equations corresponding to either S or D. In this way, we can deterministically find $R = P \pm Q$ given P and Q. In practice, we compute the greatest common divisor of two polynomials of degree-2 to derive the shared root. For example, the root shared between $f_{1,3}(X_1, x_3)$ and $f_{2,3}(X_2, x_3)$ corresponds to X_3. In our implementation, we specialise the inversion-free Euclid algorithm from [117] to polynomials of degree-2 to get a precise cost for such a computation. The resulting algorithm PointDifference is optimised to reduce finite field arithmetic.

Algorithm 24 PointDifference

Input: A Kummer surface $\mathcal{K}$ with two points $P, Q \in \mathcal{K}$.
Output: A deterministic point $R \in \mathcal{K}$, with $R = P \pm Q$.

1: **for** $i, j \in [1..4]$ with $i \leq j$ **do**
2: $\quad B_{i,j} \leftarrow B_{i,j}^{\mathcal{K}}(P, Q)$
3: **for** $1 \leq i, j \leq 4$ with $i < j$ **do**
4: $\quad f_{i,j} \leftarrow B_{j,j}x_i^2 - 2B_{i,j}x_i x_j + B_{i,i}x_j^2$
5: Write $f_{1,2}(1, x_2)$ as $(x_2 - \alpha_1)(x_2 - \alpha_2)$ with $\alpha_1 > \alpha_2$.
6: Write $\gcd(f_{1,3}(1, x_3), f_{2,3}(\alpha_2, x_3))$ as $(x_3 - \alpha_3)$
7: Write $\gcd(f_{1,4}(1, x_4), f_{2,4}(\alpha_2, x_4))$ as $(x_4 - \alpha_4)$
8: **return** $R = (1 : \alpha_2 : \alpha_3 : \alpha_4)$

It is possible to return both S and D (without knowing which one is which) by using the other root α_1 of $f_{1,2}(1, x_2)$. The extra cost for this is only two extra gcd computations. This version of the algorithm is used in later sections too.

On the Squared Kummer Surface.

The above algorithm PointDifference works well on the canonical Kummer surface $\mathcal{K}^{\text{can}}_{\lambda,\mu,\nu}$, as the bilinear forms $B_{i,j}$ are symmetric and efficiently computable for this surface. However, as explained well by Renes and Smith [326], this is not

the case on $\mathcal{K}^{\text{Sqr}}$ which has costly equations for $B_{i,j}$. Nevertheless, there exists another Kummer surface model, called the *Intermediate* Kummer surface $\mathcal{K}^{\text{Int}}$, which has elegant bilinear forms and is related to $\mathcal{K}^{\text{Sqr}}$ via a Hadamard map $\mathcal{K}^{\text{Sqr}} \xrightarrow{\mathtt{H}} \mathcal{K}^{\text{Int}}$. Therefore, rather than working with the inefficient biquadratics of $\mathcal{K}^{\text{Sqr}}$ we simply use the efficient isomorphisms

$$\mathcal{K}^{\text{Sqr}} \xrightarrow{\mathtt{H}} \mathcal{K}^{\text{Int}} \xrightarrow{\mathtt{H}} \mathcal{K}^{\text{Sqr}}.$$

That is, we map P and Q to $\mathtt{H}(P)$ and $\mathtt{H}(Q)$ on $\mathcal{K}^{\text{Int}}$, compute the point difference R of $\mathtt{H}(P)$ and $\mathtt{H}(Q)$ using PointDifference on $\mathcal{K}^{\text{Int}}$, and map back $R \mapsto \mathtt{H}(R) \in \mathcal{K}^{\text{Sqr}}$ to find the required $\mathtt{H}(R) = P \pm Q$.

4.3 POINT COMPRESSION

We now describe how to perform point compression for points on Kummer surfaces. More precisely, we show how to compress a point K of order 2^f to a scalar $s \in \mathbb{Z}/2^f\mathbb{Z}$, where $P + [s]Q$ and $P, Q \in \mathcal{K}[2^f]$ are deterministically sampled points. This is useful in cryptographic settings where K is sent over a public channel, as we can send s instead of K thus reducing the communication cost. As long as the receiver deterministically arrives at the same points P, Q, they can recompute the same $K = P + [s]Q$ given only s. We restrict here to elliptic Kummer surfaces $\mathcal{K}_\alpha$, though the theory is generally applicable.

Using Theorem 12, we may assume two elements $D_P, D_Q \in \mathcal{J}_\mathcal{C}[N]$ that span a subgroup $\mathbb{Z}_N \times \mathbb{Z}_N$ with $N = 2^f$ and $P, Q \in \mathcal{K}_\mathcal{C}[N]$ their images. We want to compress a given point $K \in \mathcal{K}^{\text{Sqr}}[N]$, whose pre-image on $\mathcal{J}^{\text{Sqr}}_{\lambda,\mu,\nu}$ is in the subgroup $\langle D_P, D_Q \rangle$.[12] In general, this is feasible as long as we can solve a discrete logarithm in base N, which is simple in our case $N = 2^f$. Thus, after finding preimages for the points P, Q and K on $\mathcal{J}_\mathcal{C}$, we can solve the discrete logarithm on $\mathcal{J}_\mathcal{C}$ by adapting algorithms from [313] to obtain $s \in \mathbb{Z}_N$. We then compute both $D, S = P \pm Q, P \mp Q$ using PointDifference and recompute both $K_D = \mathtt{3ptLadder}(P, Q, D, s)$ and $K_S = \mathtt{3ptLadder}(P, Q, S, s)$. One of K_D and K_S will then equal K. The compression is thus $s \in \mathbb{Z}_N$ plus an additional bit to indicate the use of D or S.

Decompression requires the value s and a single bit to determine whether to use D or S in `3ptLadder`. We then recompute K by deterministically sampling

12 We stress that these properties can be derived from the profiles of P, Q and K and do not require the pre-images D_P, D_Q and D_K explicitly.

Algorithm 25 PointCompression

Input: A Kummer surface $\mathcal{K}$, deterministically generated points P, Q of order N and a point K of order N such that $D_K \in \langle D_P, D_Q \rangle \subset \mathcal{J}_\mathcal{C}$.
Output: An $s \in \mathbb{Z}_N$ and $b \in \{0,1\}$ such that $K = \texttt{3ptLadder}(P, Q, D_b, s)$.

$D_P \leftarrow \rho^{-\mathrm{Sqr}}(P),\ D_Q \leftarrow \rho^{-\mathrm{Sqr}}(Q),\ D_K \leftarrow \rho^{-\mathrm{Sqr}}(K)$.
$s \leftarrow \texttt{DiscLog}(K, P, Q)$
$D_0, D_1 \leftarrow$ PointDifference(P, Q)
$K_0 \leftarrow \texttt{3ptLadder}(P, Q, D_0, s)$
if $K_0 = K$ **then**
    return $s, 0$
return $s, 1$

P, Q, recomputing $P - Q$ as either D or S, and deriving $K = \texttt{3ptLadder}(P, Q, P - Q, s)$.

4.4 EFFICIENT POINT SAMPLING

To use PointCompression, we require deterministically-sampled points P and Q such that their pre-images D_P and D_Q span the $\mathbb{Z}/2^f \times \mathbb{Z}/2^f$ subgroup containing the pre-image of K, which can be guaranteed by Theorem 12 as long as P and Q have the right profile. We describe two methods to deterministically sample points with the right profile.

Naive Point Sampling.

Sample P as a deterministic random element of $\mathcal{K}^{\mathrm{Sqr}}$, and compute the profile $t_{\ker[2]}(P)$ until $t_{\ker[2]}(P)$ is as predescribed. Then sample Q deterministically random until $t_{\ker[2]}(Q)$ is as predescribed and, additionally, different from $t_{\ker[2]}(P)$. Multiply both by the cofactor $\frac{m}{2^f}$ to ensure both P and Q have order 2^f.

Elegant Point Sampling.

As a generic element D_P of the Jacobian $\mathcal{J}_\mathcal{C}$ can be obtained by two *curve* points $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ on $\mathcal{C}(\mathbb{F}_p)$, we can sample deterministic quasi-random points on $\mathcal{K}$ by sampling random points P_i on $\mathcal{C}$, and mapping them to $\mathcal{K}$. We fix $P_2 = (w_i, 0)$ to be a Weierstrass point of $\mathcal{C}$, so that we get quasi-random elements $D_P = (P_1) + (P_2) \in \mathcal{J}_\mathcal{C}$ with Mumford representation

$D_P = \langle (x-x_1)(x-w_i), - \rangle$. This has the advantage that the profile $t_{\ker[2]}(D_P) = (t_2(D_{k,m}, D_P))_{1 \le k,m \le 6}$ is completely determined by products of the Legendre symbols of $(x_1 - w_j)$ and $(w_i - w_j)$ for $1 \le j \le 6$. By Equation (25),

$$T_2(D_{k,m}, D_P) = (w_i - w_k)(w_i - w_m)(x_1 - w_k)(x_1 - w_m). \tag{27}$$

Thus, if we know the quadratic residues of $w_i - w_j$, then the complete profile of D_P only depends on the quadratic residues of $x_1 - w_i$. Adapting the main theorem of Ohashi [298], we derive the quadratic residues of $w_i - w_j$ for $\mathcal{C}_{\lambda,\mu,\lambda\mu}$ as in Scholten's construction.

Lemma 14. Let $\lambda, \mu, \lambda\mu$ be (superspecial) Rosenhain invariants derived from a supersingular E_α through Scholten's construction. Then,

$$\begin{gathered} \lambda,\ 1-\mu,\ 1-\nu,\ \lambda-\mu,\ \mu-\nu \text{ are squares in } \mathbb{F}_p, \\ \text{and } \mu,\ \nu,\ 1-\lambda,\ \nu-\lambda \text{ are non-squares in } \mathbb{F}_p. \end{gathered}$$

Proof. First, $\lambda = \mu_1/\mu_2$ is a square and, per technical details of Scholten's construction, the quadratic reciprocity of μ is the opposite of λ. This proves both μ and $\nu = \lambda\mu$ are non-squares. The quadratic reciprocities of $1-\mu$, $1-\nu$ and $\lambda - \mu$ are identical through their relation to the theta constants (see Section 2.5). By direct computation, $\lambda - \mu$ has the same quadratic reciprocity as $(\beta_0^2 + \beta_1^2) \cdot (\gamma_0^2 + \gamma_1^2) = n(\beta)n(\gamma)$, and γ, β are squares [125, Lemma 1]. For $1-\lambda$, observe by [192, §7.5] that the quadratic reciprocity of $\lambda - 1$ is that of $(\nu - 1) \cdot (\mu - 1)$ and thus $1 - \lambda$ is non-square. Then, combining these results, we find that $\mu - \nu = \mu \cdot (1 - \lambda)$ is square, and $\nu - \lambda = -\lambda \cdot (1 - \mu)$ non-square. □

Thus, to efficiently sample points with some predescribed profile, we do the following.

1. Sample a random $x_1 \in \mathbb{F}_p$ until $x_1(x_1-1)(x_1-\lambda)(x_1-\mu)(x_1-\nu)$ is square i.e. $P_1 = (x_1, -) \in \mathcal{C}_{\lambda,\mu,\nu}$.
2. Compute the Legendre symbols of x_1, $x_1 - 1$, $x_1 - \mu$ and $x_1 - \nu$.
3. Use these values to determine the profile of D_P using Equation (27). Start over if the profile is not as predescribed.
4. Map D_P to $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ to obtain a point $P \in \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ of pre-described profile.

We can go one step further: instead of sampling a random $x_1 \in \mathbb{F}_p$, for the fixed system parameter p, we precompute a list $\mathtt{L}_{b_0,b_1}$ of small $\mathbb{F}_p$-values x, where x

has a Legendre symbol $b_0 \in \{1, -1\}$ and $x - 1$ a Legendre symbol $b_1 \in \{1, -1\}$. Thus, Step 3 only requires the computation of the Legendre symbol of $x - \mu$ and $x - \nu$ to derive the full profile $t_{\ker[2]}(P)$.

The value $t_2(D_P, D_{2,3})$ is the Legendre symbol of $x_1(x_1 - 1)w_i(w_i - 1)$ and thus the specific element $t_{2,3}(P)$ of the profile $t_{\ker[2]}(P)$ is completely precomputable by precomputing the sets $\mathtt{L}_{b_0,b_1}$. Thus, to find a point with a given profile $T = (t'_{i,j})_{1 \le i < j \le 6}$, we can find the associated list $\mathtt{L}_{b_0,b_1}$ that matches up to $t'_{2,3} = t_{2,3}(P)$. We then go through the $x \in \mathtt{L}_{b_0,b_1}$ until we find $P_1 = (x, y) \in \mathcal{C}_{\lambda,\mu,\nu}$, compute the Legendre symbols of $x - \mu$ and $x - \nu$, and derive $t_{\ker[2]}(D_P)$. If $t_{\ker[2]}(D_P) = T$, we map $D_P \mapsto P \in \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$. This computes a point on $\mathcal{K}$ using one square root (to get y) and two Legendre symbols, plus the multiplications required to map to $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ using Equation (19). With no loss in performance, we fix $P_2 = (w_4, 0)$.

This is summarised in Algorithm 26, where $\mathtt{DeriveProfile}_{\mathtt{L}}$ determines $t_{\ker[2]}(P)$ using Equation (27) given the Legendre symbols of $x - \mu$ and $x - \nu$ for $x \in \mathtt{L}_{b_0,b_1}$.

Algorithm 26 Improved point sampling on squared Kummer surfaces

Input: A squared Kummer surface $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$, a target profile T, and a precomputed list of $\mathbb{F}_p$-values $\mathtt{L}_T = [x_1, x_2, \ldots]$.

Output: A point $P \in \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ with profile $t_{\ker[2]}(P) = T$.

for $x \in \mathtt{L}_T$ **do**
  $\mathtt{RHS} \leftarrow x(x-1)(x-\lambda)(x-\mu)(x-\nu)$
  if $\mathtt{IsSquare}(\mathtt{RHS})$ **then**
    $y \leftarrow \sqrt{\mathtt{RHS}},\ \gamma_1 \leftarrow \mathtt{IsSquare}(x-\mu),\ \gamma_2 \leftarrow \mathtt{IsSquare}(x-\nu)$
    $t_{\ker[2]}(D_P) \leftarrow \mathtt{DeriveProfile}_L(\gamma_1, \gamma_2)$
    if $t_{\ker[2]}(D_P) = T$ **then**
      $u_0 \leftarrow x \cdot w_4,\ u_1 \leftarrow -(x + w_4),\ v_0^2 \leftarrow (w_4/(x - w_4))^2 \cdot \mathtt{RHS}$
      $X_1 \leftarrow \mu_1 \cdot \big(u_0(w_3w_5 - u_0)(w_4 + w_6 + u_1) - v_0^2\big)$
      $X_2 \leftarrow \mu_2 \cdot \big(u_0(w_4w_6 - u_0)(w_3 + w_5 + u_1) - v_0^2\big)$
      $X_3 \leftarrow \mu_3 \cdot \big(u_0(w_3w_6 - u_0)(w_4 + w_5 + u_1) - v_0^2\big)$
      $X_4 \leftarrow \mu_4 \cdot \big(u_0(w_4w_5 - u_0)(w_3 + w_6 + u_1) - v_0^2\big)$
      return $P = (X_1 : X_2 : X_3 : X_4)$

§5 (2, 2)-Isogenies on Kummer Surfaces

This chapter aims to showcase Kummer surfaces as objects that can lead to practical isogeny-based cryptography. Central to this, therefore, is an understanding of isogenies between Kummer surfaces. This section describes the known and new theory of isogenies between squared Kummer surfaces. In particular, let $\mathcal{J}/\mathbb{F}_p$ be the Jacobian of a genus-2 curve in Rosenhain form, with corresponding squared Kummer surface $\mathcal{K}^{\mathrm{Sqr}}$ defined over $\mathbb{F}_p$.

In Section 5.1, we discuss $(2,2)$-isogenies on (squared) Kummer surfaces, described as pairs of 2-torsion points $L_{i,j}$, $L_{k,l}$ on $\mathcal{K}^{\mathrm{Sqr}}$ whose preimages $D_{i,j}$, $D_{k,m}$ generate a $(2,2)$-subgroup of $\mathcal{J}[2]$. For each of these kernels, in Section 5.2, we give efficient formulas for computing the corresponding $(2,2)$-isogeny defined over $\overline{\mathbb{F}}_p$. For the construction of efficient cryptographic protocols, we require the isogenies to be defined over the base field $\mathbb{F}_p$. In Section 5.3, we describe how this can be achieved for certain isogenies, which will be sufficient for our application.

Comparison with other works.

Computing $(2,2)$-isogenies between Kummer surfaces has been studied by various other works. Cassels and Flynn [84, Ch. 9] study $(2,2)$-isogenies between general Kummer surfaces, whilst Dartois, Maino, Pope, and Robert [142] use the theta model to give efficient formulas for computing $(2,2)$-isogenies between canonical Kummer surfaces including a constant-time implementation of their algorithm in Rust, later improved by [332, §8]. In constrast, in Section 5.2 we focus instead on deriving isogenies between *squared* Kummer surfaces.

5.1 From subgroups to $(2,2)$-isogenies

Consider an (N,N)-subgroup $G \subseteq \mathcal{J}_1[N]$, i.e., a group $G = \langle D_R, D_S \rangle$ generated by two N-torsion points $D_R, D_S \in \mathcal{J}_1[N]$ with $e_N(R,S) = 1$, where e_N is the N-Weil pairing. Any (N,N)-*isogeny* between Jacobians of genus 2 curves is a surjective finite morphism $\Phi\colon \mathcal{J}_1 \to \mathcal{J}_2 = \mathcal{J}_1/G$, whose kernel is a (N,N)-subgroup $G \subseteq \mathcal{J}_1[N]$, and where $\Phi(\mathcal{O}_1) = \mathcal{O}_2$. Any (N,N)-isogeny Φ

descends to a morphism φ of squared Kummer surfaces $\varphi\colon \mathcal{K}_1^{\mathrm{Sqr}} \to \mathcal{K}_2^{\mathrm{Sqr}}$, such that the following diagram commutes:

$$\begin{array}{ccc} \mathcal{J}_1 & \xrightarrow{\quad\Phi\quad} & \mathcal{J}_2 \\ \downarrow{\scriptstyle \rho^{\mathrm{Sqr}}} & & \downarrow{\scriptstyle \rho^{\mathrm{Sqr}}} \\ \mathcal{K}_1^{\mathrm{Sqr}} & \xrightarrow{\quad\varphi\quad} & \mathcal{K}_2^{\mathrm{Sqr}} \end{array}$$

With our target application in mind, we focus in particular on the case $N = 2$. By abuse of notation, we then call φ a $(2,2)$-*isogeny of squared Kummer surfaces*, whose kernel is given by the image of G in $\mathcal{K}_1^{\mathrm{Sqr}}$.

To construct such $(2,2)$-isogenies, we decribe how the map φ is derived from two 2-torsion points in $\mathcal{K}^{\mathrm{Sqr}}[2]$, described in Section 2.4. As long as $\{i,j\} \cap \{k,l\} = \emptyset$, two points $L_{i,j}, L_{k,l} \in \mathcal{K}[2]$ will generate a $(2,2)$-subgroup, and thus give us a kernel of a $(2,2)$-isogeny[13]. These fifteen $(2,2)$-subgroups are given by $\langle L_{i,j}, L_{k,l}\rangle = \{\mathbf{o},\ L_{i,j},\ L_{k,l},\ L_{i,j} + L_{k,l}\}$, where

$$(i,j,k,l) \in \left\{ \begin{array}{l} (1,2,3,4),(1,2,4,6),(1,2,3,6),(2,3,5,6),(1,3,5,6), \\ (1,6,3,4),(2,6,3,4),(2,3,4,5),(1,3,4,5),(1,4,3,6), \\ (2,4,3,6),(2,3,4,6),(1,3,4,6),(1,4,3,5),(2,4,3,5) \end{array} \right\}$$

Note that $L_{i,j} + L_{k,l} = L_{m,n}$ where $\{m,n\} = \{1,2,3,4,5,6\}\backslash\{i,j,k,l\}$. The corresponding $(2,2)$-isogenies are $\varphi_{ijkl} : \mathcal{K}_1^{\mathrm{Sqr}} \to \mathcal{K}_2^{\mathrm{Sqr}} = \mathcal{K}_1^{\mathrm{Sqr}}/\langle L_{i,j}, L_{k,l}\rangle$.

5.2 ISOGENIES DEFINED OVER $\overline{\mathbb{F}}_p$

We first analyse the general case, where the $(2,2)$-isogenies are defined over $\overline{\mathbb{F}}_p$. For each $(2,2)$-subgroup G, there is a map $\alpha_G : \mathcal{K}_1^{\mathrm{Sqr}} \to \mathcal{K}_1^{\mathrm{Sqr}}$ defined by a 4×4-matrix $\mathbf{A}_G$ whose entries lie in $\{0, \pm 1, \pm i\}$, with $i^2 = -1$. The corresponding $(2,2)$-isogeny φ_G is given by

$$\varphi_G := \mathtt{S} \circ \alpha_G \circ \mathtt{C}_{\mathtt{Inv}(A:B:C:D)} \circ \mathtt{H},$$

13 This is equivalent to a quadratic splitting [356] of the curve equation as used to compute Richelot isogenies.

where the maps $\mathtt{S}, \mathtt{C}$ and $\mathtt{H}$ are as defined in Section 2.6. The matrix $\mathbf{A}_G$ for each $(2,2)$-subgroup is specified in [119, Appendix A].

Field of Definition of the $(2,2)$-Isogeny.

For $\mathcal{K}^{\mathrm{Sqr}}$ defined over $\mathbb{F}_p$, the maps $\mathtt{H}$ and $\mathtt{S}$ are defined over $\mathbb{F}_p$. Whenever $\mathbf{A}_G$ contains only entries in $\{0, \pm 1\}$, also α_G is defined over $\mathbb{F}_p$. This is the case for *six* kernels $G = \langle L_{i,j}, L_{k,l} \rangle$, where

$$(i,j,k,l) \in \left\{ \begin{array}{l} (1,2,3,4), (1,2,4,6), (2,3,5,6), \\ (1,6,3,4), (2,3,4,5), (1,4,3,6) \end{array} \right\}.$$

The scaling map $\mathtt{C}_{\mathtt{Inv}(A:B:C:D)}$, however, requires the computation of $(A : B : C : D)$ from $\mathbf{o} = (\mu_1 \colon \mu_2 \colon \mu_3 \colon \mu_4)$ which in turn requires at most three square roots (and a handful of additions) in $\overline{\mathbb{F}}_p$, using the fact that

$$(A^2 \colon B^2 \colon C^2 \colon D^2) = \mathtt{H}(\mathbf{o}).$$

In some cases, however, $(A : B : C : D)$ can be computed more efficiently without taking square roots.

For the kernel $G = \langle L_{1,2}, L_{3,4} \rangle$, the map α_G is the identity, and so the corresponding $(2,2)$-isogeny is given by

$$\varphi_G = \mathtt{S} \circ \mathtt{C}_{\mathtt{Inv}(A:B:C:D)} \circ \mathtt{H} = \mathtt{C}_{\mathtt{Inv}(A^2:B^2:C^2:D^2)} \circ \mathtt{S} \circ \mathtt{H},$$

where we swap $\mathtt{S}$ and $\mathtt{C}$ to avoid taking square roots, so φ_G is defined over $\mathbb{F}_p$.

For the kernels $G = \langle L_{i,j}, L_{k,l} \rangle$, where

$$(i,j,k,l) \in \left\{ \begin{array}{l} (2,3,4,5), (1,3,4,5), (1,4,3,6), (2,4,3,6), \\ (2,3,4,6), (1,3,4,6), (1,4,3,5), (2,4,3,5) \end{array} \right\},$$

the required square roots can be extracted from the 4-torsion points lying above the kernel points. More precisely, let R, S be such that $[2]R = L_{i,j}$ and $[2]S = L_{k,l}$, and let $h_{P,i}$ be the i-th coordinate of $\mathtt{H}(P)$. For the sake of clarity, we suppose that $L_{i,j}$ and $L_{k,l}$ are of the form $(1 \colon \star \colon 0 \colon 0)$ and $(1 \colon 0 \colon \star \colon 0)$, respectively. The other cases follow similarly.

We compute the scaling value $(1/A : 1/B : 1/C : 1/D)$ using the points lying above the kernel by noting that

$$\frac{A}{B} = c_1 \frac{h_{R,1}}{h_{R,2}}, \quad \frac{A}{C} = c_2 \frac{h_{S,1}}{h_{S,3}}, \text{ and } \frac{C}{D} = c_3 \frac{h_{R,3}}{h_{R,4}}, \tag{28}$$

for some constants $c_i \in \{1, -i\}$. From this, we get $(1/A : 1/B : 1/C : 1/D)$ by

$$\Big(\; h_{R,2}h_{S,3}h_{S,3}h_{R,4} \; : \; c_1 h_{R,1}h_{S,3}h_{S,3}h_{R,4} \; : \\ c_2 h_{R,2}h_{S,1}h_{S,3}h_{R,4} \; : \; c_2c_3 h_{R,2}h_{S,3}h_{S,1}h_{R,3} \; \Big).$$

We see that $(1/A : 1/B : 1/C : 1/D)$ requires no inversions, and only needs operations in $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$, assuming we have $\mathbb{F}_p$- or $\mathbb{F}_{p^2}$-rational 4-torsion (respectively) and depending on the constants c_i, which are given in Section D.

5.3 (2, 2)-ISOGENIES DEFINED OVER $\mathbb{F}_p$

To obtain efficient protocols, we are interested in $(2, 2)$-isogenies that are defined over $\mathbb{F}_p$. Such isogenies arise from kernels where both α_G is defined over $\mathbb{F}_p$ and the scaling map $\mathtt{C}_{\mathtt{Inv}(A:B:C:D)}$ can be computed using $\mathbb{F}_p$-operations. The latter happens when the constants c_i lie in $\mathbb{F}_p$ and we have $\mathbb{F}_p$-rational points R, S lying above $L_{i,j}$, $L_{k,l}$, respectively. Assuming we have rational 4-torsion points lying above, the isogeny φ_G is thus defined over $\mathbb{F}_p$ for kernels $G = \langle L_{i,j}, L_{k,l} \rangle$ where

$$(i, j, k, l) \in \{(1, 2, 3, 4), (2, 3, 4, 5), (1, 4, 3, 6)\}.$$

These kernels coincide precisely with the isogenies derived by Costello [125] which we will require in Section 7 to define SQIsign verification using squared Kummer surfaces. However, when using elliptic Kummer surfaces $\mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\lambda\mu}$ arising from Scholten's construction, these kernels can surprisingly be generated by only a *single* point rather than a subgroup G of $\mathcal{K}^{\mathrm{Sqr}}[2]$. We discuss this in more depth in Section 6.

COST OF COMPUTING A $(2, 2)$-ISOGENY OVER $\mathbb{F}_p$. The maps $\mathtt{H}$, $\mathtt{C}$, $\mathtt{Inv}$ and $\mathtt{S}$ together cost $14\mathbf{M}$ and $8\mathbf{a}$. For kernel $\langle L_{1,2}, L_{3,4} \rangle$ we compute $(A^2 : B^2 : C^2 : D^2)$ as $\mathtt{H}(\mathbf{o})$ using $8\mathbf{a}$, meaning the corresponding $(2, 2)$-isogeny requires $14\mathbf{M}$ and $16\mathbf{a}$ to compute. For kernels $\langle L_{2,3}, L_{4,5} \rangle$ and $\langle L_{1,4}, L_{3,6} \rangle$, computeing the scaling

point using the 4-torsion points, as shown in Equation (28), requires (at most) 6**M** and 16**a**.

§6 (2, 2)-Isogenies on Elliptic Kummer Surfaces

In this section, we specialise the $(2^n, 2^n)$-isogenies from the previous section to isogenies between *elliptic* Kummer surfaces as derived from Scholten's construction. Surprisingly, as these isogenies are induced by 2^n-isogenies between elliptic curves, we can describe these using a single point $P \in \mathcal{K}[2^n]$ We refer to such isogenies as *elliptic* isogenies.

We first describe the elliptic $(2,2)$-isogenies $\varphi_i : \mathcal{K}_\alpha \to \mathcal{K}'_\alpha$ that correspond to the elliptic curve 2-isogenies $\phi : E_\alpha \to E'_\alpha = E_\alpha/\langle D \rangle$, where $D \in E[2] \setminus \{\mathcal{O}_E\}$ thus recovering the isogenies given by Costello [125]. In particular, these elliptic $(2,2)$-isogenies can be computed more efficiently than general $(2,2)$-isogenies between Kummer surfaces arising from theta coordinates [142]. We then discuss how to compute *chains* of elliptic isogenies between elliptic Kummer surfaces. This will be key to our application in Section 7 on SQIsign verification, which relies heavily on $(2^n, 2^n)$-isogenies.

6.1 elliptic $(2,2)$-isogenies

Let $D_0 = (0,0)$, $D_\alpha = (\alpha, 0)$ and $D_{\tilde{\alpha}} = (1/\alpha, 0)$ denote the three points of order 2 on a supersingular Montgomery curve $E_\alpha : y^2 = x(x-\alpha)(x-1/\alpha)$. Let $D \in E_\alpha[8]$ be a point of order 8 so that $[4]D \in \{D_0, D_\alpha, D_{\tilde{\alpha}}\}$. We assume throughout this section that D is $\mathbb{F}_{p^2}$-rational so that the image of D on $\mathcal{K}_\alpha$ is $\mathbb{F}_p$-rational. We remark that as E_α is supersingular, there is always $\mathbb{F}_{p^2}$-rational 8-torsion as $8 \mid \#E_\alpha(\mathbb{F}_{p^2}) = (p \pm 1)^2$ for any odd prime p. Recall that $\bar{\eta} : E_\alpha \to \mathcal{K}_\alpha$ from Section 2.9 is itself a $(2,2)$-isogeny, thus, when we map D down to the Kummer surface $\mathcal{K}_\alpha$, we obtain a 4-torsion point $P' := \bar{\eta}(D)$. Furthermore, $P = [2]P'$ completely describes the $(2,2)$-isogeny corresponding to the elliptic curve isogeny φ, which we depict in Figure 23.

In the lemma that follows, we give explicit equations for the 3 possible elliptic $(2,2)$-isogenies induced by the 2-isogenies generated by $D_0, D_\alpha, D_{\tilde{\alpha}}$ on E_α In particular, we see that the isogenies defined over $\overline{\mathbb{F}}_p$ given in Section 5.3 collapse to the isogenies given by Costello [125] in this special setting when $\nu = \lambda\mu$.

Lemma 15 (Eq. 18, [125])**.** Let $D \in E_\alpha[8]$ such that $[4]D \in \{D_0, D_\alpha, D_{\tilde{\alpha}}\}$, and let $\mathcal{K}_\alpha$ be the associated elliptic Kummer surface. Let $P' := \bar{\eta}(D) \in \mathcal{K}_\alpha[4]$ and

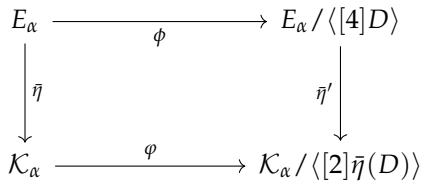


Figure 23: The elliptic $(2,2)$-isogeny φ induced by a 2-isogeny ϕ.

$P := [2]P' \in \mathcal{K}_\alpha[2]$, and denote their images under the Hadamard map by $\mathtt{H}(P) = (h_1 : h_2 : h_3 : h_4)$ and $\mathtt{H}(P') = (h'_1 : h'_2 : h'_3 : h'_4)$. Then:

1. If $D = D_0$, then $P \in \{L_{5,6}, L_{3,4}\}$ describes the isogeny given by $\varphi_0 = \mathtt{C}_S \circ \mathtt{S} \circ \mathtt{H}$, where $S = \mathtt{Inv}(\mathtt{H}(\mathbf{o}))$.
2. If $D = D_\alpha$, then $P \in \{L_{2,3}, L_{1,6}\}$ describes $\varphi_\alpha = \mathtt{S} \circ \mathtt{H} \circ \mathtt{C}_S \circ \mathtt{H}$, where
   - $S = (h_2h'_4 : h_1h'_4 : h_2h'_1 : h_2h'_1)$, if $P = L_{1,6}$, or
   - $S = (h_2h'_3 : h_1h'_3 : h_2h'_1 : h_2h'_1)$, if $P = L_{2,3}$.
3. If $D = D_{\tilde{\alpha}}$, then $P \in \{L_{1,4}, L_{2,5}\}$ describes $\varphi_{\tilde{\alpha}} = \mathtt{S} \circ \mathtt{H} \circ \mathtt{C}_S \circ \mathtt{H}$, where
   - $S = (h_2h'_4 : h_1h'_4 : h_2h'_1 : h_2h'_1)$, if $P = L_{2,5}$, or
   - $S = (h_2h'_3 : h_1h'_3 : h_2h'_1 : h_2h'_1)$, if $P = L_{1,4}$.

We briefly describe how the isogenies given in Section 5.3 that can be defined over $\mathbb{F}_p$ collapse to the isogenies in Lemma 15 in this specific setting.

THE ISOGENY φ_0 follows directly noting that $\mathbf{A}_G$ is the identity map.

THE ISOGENY φ_α has $\mathbf{A}_G = \mathtt{H}$ and $S = \mathtt{Inv}(A : B : C : D)$, where S is as in the statement of the lemma. Indeed, observing that for elliptic Kummer surfaces we have $C = D = 1$, we have that $\mathtt{H}(P) = (A : B : B : A)$ and so $h_1/h_2 = A/B$.

When $P = L_{2,3} = (1 : 0 : \star : 0)$, using Equation (28) we have that $h'_1/h'_3 = A$. Therefore $S = (1 : h_1/h_2 : h'_1/h'_3 : h'_1/h'_3)$. In the other case, $P = (1 : 0 : 0 : \star)$ and $h'_1/h'_4 = A$. Therefore $S = (1 : h_1/h_2 : h'_1/h'_4 : h'_1/h'_4)$.

THE ISOGENY $\varphi_{\tilde{\alpha}}$ has $\mathbf{A}_G : (X_1 : X_2 : X_3 : X_4) \mapsto \mathtt{H}(-X_1 : X_2 : X_3 : X_4)$. We can verify that $S = \mathtt{Inv}(-A : B : C : D)$ in a similar method to above, and so $\mathbf{A}_G \circ \mathtt{C}_{\mathtt{Inv}(A:B:C:D)} = \mathtt{H} \circ \mathtt{C}_S$, as required.

In this way, the isogenies are fully described by a single point P, rather than two points as in the previous sections.

6.2 IMAGES OF $(2,2)$-ISOGENIES

In what follows, we describe how we compute the codomain of elliptic isogenies, the Rosenhain invariants of the codomain, and how to evaluate points under an elliptic isogeny. We give explicit counts for each procedure, which refer to the optimised implementation of each routine found in the accompanying code.

COMPUTING THE CODOMAIN OF AN ELLIPTIC ISOGENY. To compute the codomain of an elliptic $(2,2)$-isogeny φ, it suffices to compute the constants defining the codomain, given by $\varphi(\mathbf{o})$. For isogeny φ_0, we have that $\varphi(\mathbf{o}) = \mathtt{H}(\mathbf{o})$, which can be computed in $8\mathbf{a}$. For isogenies φ_α and $\varphi_{\tilde{\alpha}}$, we compute the constants by evaluating the isogeny at $\mathbf{o}$ in $11\mathbf{M}$ and $32\mathbf{a}$.

COMPUTING THE ROSENHAIN INVARIANTS. Key to many of the algorithms in Sections 3 and 4 is the knowledge of the Rosenhain invariants of the curve $\mathcal{C}_{\alpha'}$ associated to the codomain Kummer surface $\mathcal{K}_{\alpha'}$. Gaudry [192, §4.2] shows that for elliptic Kummer surfaces, we have

$$\lambda = \frac{\mu_1}{\mu_2}, \quad \mu = \frac{\tau\mu_2 - 1}{\tau\lambda\mu_2 - 1}, \quad \nu = \lambda\mu, \tag{29}$$

with τ as defined in Section 2.5. Thus, to compute the Rosenhain invariants of the codomain, we need to compute τ, which we can extract from the 2-torsion points on $\mathcal{K}_{\alpha'}$ in the following way:

Consider the elliptic isogeny $\varphi : \mathcal{K}_\alpha \to \mathcal{K}_{\alpha'}$ described by $K \in \mathcal{K}_\alpha[2]$. Let R be the point of order 4 such that $K = [2]R$. Then, $\varphi(R)$ is a point of order 2 on $\mathcal{K}'$. If R is the kernel of φ_α or $\varphi_{\tilde{\alpha}}$ on $\mathcal{K}_{\alpha'}$, using the description of the 2-torsion on elliptic Kummer surfaces in Section 2.5, we find that the non-zero coordinates of $\varphi(R)$ will be 1 and τ or $1/\tau$ after normalisation, respectively. We use one bit of information to communicate whether we compute τ or $1/\tau$, which we then use to compute the Rosenhain invariants of the codomain from Equation (29).

EVALUATING POINTS UNDER ELLIPTIC ISOGENIES. This amounts to evaluating the maps from Lemma 15 at a point $(X_1 : X_2 : X_3 : X_4)$. We assume we have the point S used in the scaling map $\mathtt{C}_S$ defined in Lemma 15. For φ_0, this then takes $8\mathbf{M}$ and $8\mathbf{a}$. For φ_α and $\varphi_{\tilde{\alpha}}$, this takes of $8\mathbf{M}$ and $16\mathbf{a}$.

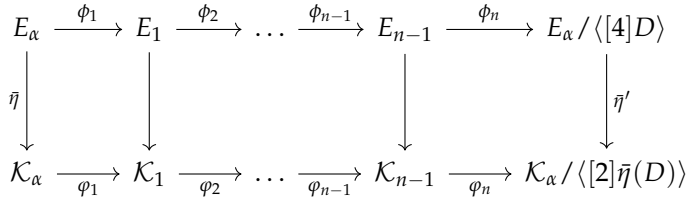


Figure 24: The elliptic $(2^n, 2^n)$-isogeny φ induced by the 2^n-isogeny ϕ .

6.3 CHAINS OF ELLIPTIC $(2,2)$-ISOGENIES

Following the blueprint from the previous section, given a 2^{n+2}-torsion point in $E_\alpha(\mathbb{F}_{p^2})$, we can compute an elliptic $(2^n, 2^n)$-isogeny over $\mathbb{F}_p$ between elliptic Kummer surfaces, by taking elliptic $(2,2)$-isogeny steps, using the 4-torsion on $\mathcal{K}_\alpha$ at each step to derive the isogeny efficiently. We depict this in Figure 24.

For each elliptic isogeny $\mathcal{K}_\alpha \to \mathcal{K}_{\alpha'}$, its dual isogeny is of type 1 in Lemma 15, with kernel given by $\langle L'_{1,2}, L'_{3,4}\rangle \subset \mathcal{K}_{\alpha'}$. Therefore, to avoid *backtracking*, after the first step we only take $(2,2)$-isogenies φ_α or $\varphi_{1/\alpha}$. We can therefore use the technique described in the previous section to compute the Rosenhain invariants of $\mathcal{K}_{\alpha'}$.

STRATEGIES. Naively, given a point $K \in \mathcal{K}_\alpha[2^{n+1}]$, we compute the $(2^n, 2^n)$-isogeny as a chain of elliptic $(2,2)$-isogenies of length n by deriving a 2- and 4-torsion point from K, and pushing K through the derived $(2,2)$-isogeny φ_1. After n repetitions, we find $\varphi = \varphi_n \circ \dots \circ \varphi_1$.

A better way to compute chains of isogenies is to use *optimal strategies* as originally introduced in the context of SIDH [177], easily adaptable to the Kummer surface setting [105, 119, 142]. We find that using these strategies in our implementation, we gain roughly a factor of two cost reduction compared to the naive strategy. We thank Michael Meyer for the implementation of this optimization.

§7 SQISIGN VERIFICATION ON KUMMER SURFACES

We now have the necessary tools in place to turn to our target example: performing SQIsign verification on Kummer surfaces. We refer to [100, 148], or Chapter II and Chapter IX for more details on SQIsign, in particular verification.

The essential tool we require is Scholten's construction, which allows us to construct an elliptic Kummer surface $\mathcal{K}_\alpha = \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\lambda\mu}$ defined over $\mathbb{F}_p$ corresponding to an elliptic curve E_α defined over $\mathbb{F}_{p^2}$. Beyond Scholten's construction, we require the techniques from Sections 3 and 4 to enable efficient compression of isogenies between Kummer surfaces, and we require the theory from Sections 5 and 6 to efficiently compute elliptic isogenies between Kummer surfaces. We emphasise that we initiate this construction for supersingular curves E_α which ensures superspecial elliptic Kummer surfaces $\mathcal{K}_\alpha$.

7.1 USING SCHOLTEN'S CONSTRUCTION

Let $E_\alpha : y = x(x-\alpha)(x-\frac{1}{\alpha})$ be a supersingular elliptic curve over $\mathbb{F}_{p^2}$. Let f be the largest integer such that $2^f \mid p+1$ so that $E_\alpha[2^f]$ is rational. A SQIsign response isogeny ϕ_{resp} is an isogeny of degree 2^e, with $e \approx 1000$, split up into $n = \lceil e/f \rceil$ *blocks* $\phi_i : E_i \to E_{i+1}$, such that $\phi_{\text{resp}} = \phi_n \circ \ldots \circ \phi_1$. The kernel of each ϕ_i is some $K_i \in E_i[2^f]$. We use Scholten's construction to compute the superspecial elliptic Kummer $\mathcal{K}_i^{\text{Sqr}}$ associated to the elliptic curve E_i, to get the following commutative diagram.

$$\begin{array}{ccccccc}
E_0 & \xrightarrow{\phi_1} & E_1 & \xrightarrow{\phi_2} & \ldots & \xrightarrow{\phi_n} & E_n \\
\downarrow{\scriptstyle \bar{\eta}_0} & & \downarrow{\scriptstyle \bar{\eta}_1} & & & & \downarrow{\scriptstyle \bar{\eta}_n} \\
\mathcal{K}_0^{\text{Sqr}} & \xrightarrow{\varphi_1} & \mathcal{K}_1^{\text{Sqr}} & \xrightarrow{\varphi_2} & \ldots & \xrightarrow{\varphi_n} & \mathcal{K}_n^{\text{Sqr}}
\end{array}$$

Thus, we can verify a SQIsign response $\phi_{\text{resp}} : E_0 \to E_n$ by computing the corresponding chain of elliptic $(2^f, 2^f)$-isogenies $\varphi : \mathcal{K}_0^{\text{Sqr}} \to \mathcal{K}_n^{\text{Sqr}}$, now defined over $\mathbb{F}_p$. By Lemma 6, one can also see the first and last steps of this diagram as gluing and splitting isogenies. Thus, $\phi_{\text{resp}} : E_0 \to E_n$ can similarly be viewed as a two-dimensional isogeny

$$\Phi_{\text{resp}} : E_0 \times E_0^{(p)} \to \mathcal{K}_0^{\text{Sqr}} \to \ldots \to \mathcal{K}_n^{\text{Sqr}} \to E_n \times E_n^{(p)}.$$

7.2 COMPUTING A SINGLE BLOCK

We first describe how to transport the computation of a single block $\varphi_i : E_i \to E_{i+1}$ to a computation involving Kummer surfaces.

We start with a point P on $E_i(\mathbb{F}_{p^2})$ of order 2^f. Pushing P down to $\mathcal{K}_i^{\mathrm{Sqr}}$ through the $(2,2)$-isogeny $\bar{\eta}_i$, the point $\bar{\eta}_i(P)$ has order 2^{f-1} on $\mathcal{K}_i^{\mathrm{Sqr}}(\mathbb{F}_p)$, which is the maximal power-of-two torsion on $\mathcal{K}_i$. As described in Section 6, the elliptic isogenies φ_i we derive from $\bar{\eta}_i(P)$ are those given by Lemma 15. It is most cost effective to use $\bar{\eta}_i(P)$ to perform $f-2$ isogenies, so that we can always use the 4-torsion lying above our kernel generators to compute (Rosenhain invariants of) the codomains. This elliptic $(2^{f-2}, 2^{f-2})$-isogeny φ_i then corresponds to the elliptic curve isogeny ϕ_i with kernel generator $[4]P$ of order 2^{f-2}.

In this way, by moving to the Kummers we lose 2 bits in the length of the isogeny per block. However, this should not have a large effect on performance as long as we perform SQIsign verification with the same number of blocks, as we have observed already in Chapters IX and X.

7.3 UNCOMPRESSED SQISIGN SIGNATURES

Uncompressed SQIsign is a variant of SQIsign where we assume that the kernel K_i of the i-th block $\varphi_i : E_i \to E_{i+1}$ is simply given as (the x-coordinate of) a point K_i, not using any compression techniques. SQIsign verification then consists of the recomputation of two isogenies: the above-mentioned *response* $\varphi_{\mathrm{resp}} : E_A \to E_2$ and (the dual of) the *challenge* $\varphi_{\mathrm{chall}} : E_1 \to E_2$.

The challenge isogeny.

As we only have considered efficient $(2,2)$-isogenies in this work, we require the challenge isogeny to be of degree 2^λ, where λ is the security parameter. Hence, we require $f \geq \lambda$ to be able to describe the challenge isogeny again using a single Kummer point $K \in \mathcal{K}$. Beyond that, the signer needs to be slightly more careful in constructing the challenge isogeny: they should not use the deterministic basis of E_1 to hash to a random isogeny φ_{chall}, as the verifier will only see the Kummer surface, on which the deterministically sampled basis is different. So, the signer computes the associated Kummer surface $\mathcal{K}_2^{\mathrm{Sqr}}$ and hashes to a challenge isogeny φ_{chall}, then lifts φ_{chall} to the elliptic curve to compute the curve E_2. Only then can the signer craft a response isogeny between E_A and E_2 and push this down to Kummer surfaces.

The response isogeny.

Again, we use Scholten's construction to push the full isogeny φ_{resp} to elliptic Kummer surfaces. We split up φ_{resp} into $n = \lceil e/(f-2) \rceil$ isogenies of degree 2^{f-2} to ensure that we can perform a single block on the Kummer isogenies when starting with a point of order 2^f on the elliptic curve. The response isogeny is then given as $(x_1, \ldots, x_n)$, where x_i is the x-coordinate of a point K_i generating the kernel of the i-block φ_i.

Remark 6. A minor difference between the isogeny on the elliptic curves and the Kummer surfaces is that we sometimes end up on the *twist* of $\mathcal{K}_{i+1}^{\text{Sqr}}$, which we need to correct for. However, we can simply communicate this in the signature at the cost of 1 bit, and the cost for twist correction is negligible (see Section 2.10). This does not impact security, as one could lexicographically decide on either Kummer surface to normalise this choice, and all information required is public.

7.4 COMPRESSED SQISIGN SIGNATURES

Compressing SQIsign signatures requires many of the general techniques described in Sections 3 and 4. More generally, we apply the theory we developed to improve the performance of isogeny-based cryptography on Kummer surfaces. Before describing compressed isogenies between Kummer surfaces, we recall the two core tools used for compression of elliptic curve SQIsign signatures:

1. An efficient, deterministic method to sample a basis P, Q for $E[2^f]$,
2. A recomputation of K_i as $P + [s]Q$ given a scalar $s \in \mathbb{Z}/2^f\mathbb{Z}$.

In Section 4, we gave a detailed analysis of the cost of both steps, with several optimizations which we can generalise to the higher-dimensional case.

Using the sampling method from Section 4.4, we can compress the kernel point K more efficiently than the general method sketched in PointCompression, by using that, in signing, we can compute the divisor D_K on the Jacobian $\mathcal{J}_\mathcal{C}$ corresponding to K as the image of the kernel point in the associated elliptic curve. This allows us to forgo Line 1 of Algorithm 25 (which maps the kernel point K to D_K), and instead start the point compression directly on $\mathcal{J}_\mathcal{C}$, with the caveat that we must ensure the results stay consistent for the verifier.

Our approach is as follows. Given the element $D_K \in \text{Im}(\eta) \subset \mathcal{J}_\mathcal{C}(\mathbb{F}_p)$, the signer samples $D \in \mathcal{J}_\mathcal{C}(\mathbb{F}_p)$ by sampling a deterministic sequence of points

$P \in \mathcal{C}_{\lambda,\mu,\nu}$ until $D = (P) + ((w_i, 0)) - D_\infty$ has the same profile as D_K and sets $D_P \leftarrow D$. The signer then samples D_Q similarly until we ensure $D_K \in \langle D_P, D_Q \rangle$ and computes a, b such that $D_K = [a][\frac{p+1}{2^f}]D_p + [b][\frac{p+1}{2^f}]D_Q$. As D_K has the same profile as D_P, we are ensured that a is odd and set $s = b/a \in \mathbb{Z}/2^f$. Thus, the signer updates $D_K \leftarrow [s^{-1}]D_K$.

The signer then pushes D_K, D_P and D_Q to $\mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$ as K, P and Q, and derives $D, S \in \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$ using PointDifference. Finally, the signer recomputes both K_D and K_S to find which one equals K, and adds this information as a bit b. The compression of K is then given as the pair (s, b). The verifier uses the same deterministic sampling procedure to derive $P, Q \in \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$, and only has to use efficient Kummer arithmetic, instead of expensive Jacobian arithmetic, to derive $K \in \mathcal{K}^{\text{Sqr}}_{\lambda,\mu,\nu}$.

7.5 PERFORMANCE OF SQISIGN ON KUMMER SURFACES

Our benchmarks, using the same cost model as AprèsSQI, show that SQIsign verification on Kummer surfaces, in comparison to elliptic curves, takes less than $1.5\times$ the number of $\mathbb{F}_p$-operations for both the uncompressed variant and the compressed variant. As shown in [39], the core Kummer arithmetic ($\mathtt{H}$, $\mathtt{S}$ and $\mathtt{C}_P$) can be very efficiently vectorised on larger CPUs. If such efficient vectorisation scales to the primes $p = f \cdot 2^k - 1$ used in SQIsign, this may potentially allow for faster SQIsign verification on Kummer surfaces than elliptic curves. Furthermore, many of the proposed algorithms in this work are new, and although we have optimised these to the best of our knowledge, we assume further optimizations are possible.

§8 CONCLUSIONS

We have described, used, and implemented several techniques and tools to advance the toolbox of isogeny-based cryptography in higher dimensions. Using Scholten's construction, we have shown that SQIsign verification can also be viewed as a very unique $(2^n, 2^n)$-isogeny between products of elliptic curves, with relatively little overhead. With vectorised arithmetic [39], this can potentially outperform SQIsign verification between elliptic curves, and compete with the two-dimensional approaches over $\mathbb{F}_{p^2}$ [34, 166, 288] in terms of verification speed.

Two-dimensional approaches over $\mathbb{F}_{p^2}$ seem to achieve close-to-optimal results for the length of the response isogeny, and therefore potential improvements most likely come from lower-level improvements, such as better isogeny formulas and optimised finite-field arithmetic. Our approach, however, can still improve in several aspects. First, the techniques developed in this work are novel, and closer analysis might significantly improve their performance. Second, the overall length of the response isogeny is far off from the theoretical best. Improvements to KLPT, or other approaches to achieve a shorter response therefore potentially allow for drastic improvements to verification.

In a more general sense, our work shows that isogeny-based cryptography in higher dimensions has access to a similar toolbox as isogeny-based elliptic-curve cryptography, and, with the use of Scholten's construction, we can transport primitives to higher dimensions. Using the tools developed in this work, it is not out of reach to similarly analyze higher-dimensional analogues of other isogeny-based cryptosystems.

§A Addition Matrices for Kummer Surfaces

The addition by points of order 2 is well-defined on Kummer surfaces, and can be described by a 4×4-matrix. This appendix describes these matrices for the general Kummer surface, as described by Cassels and Flynn [84] as well as for the canonical, squared, and elliptic Kummer surfaces. These latter two are original work.

THE GENERAL KUMMER SURFACE. Let $L_{i,j} \in \mathcal{K}[2]$, whose x-part of the Mumford representation on $\mathcal{J}$ is given by $a(x) = x^2 - (w_i + w_j)x + w_i \cdot w_j$. Then $a(x)$ divides the defining polynomial $f(x)$ of $\mathcal{C}$. Write $f = a \cdot h$ with $h(x) = \sum h_i x^i$, that is, $h(x) = \prod_{k \neq i,j}(x - w_k)$. Then the matrix $W_{i,j}$ is given by the composition of the 4×3 matrix

$$\begin{pmatrix} g_2^2 h_0 + g_0 g_2 h_2 - g_0^2 h_4 & g_0 g_2 h_3 - g_0 g_1 h_4 & g_1 g_2 h_3 - g_1^2 h_4 + 2 g_0 g_2 h_4 \\ -g_0 g_2 h_1 - g_0 g_1 h_2 + g_0^2 h_3 & g_2^2 h_0 - g_0 g_2 h_2 + g_0^2 h_4 & g_2^2 h_1 - g_1 g_2 h_2 - g_0 g_2 h_3 \\ -g_1^2 h_0 + 2 g_0 g_2 h_0 + g_0 g_1 h_1 & -g_1 g_2 h_0 + g_0 g_2 h_1 & -g_2^2 h_0 + g_0 g_2 h_2 + g_0^2 h_4 \\ w_{41} & w_{42} & w_{43} \end{pmatrix}$$

adjoined on the right by the column $(g_2, -g_1, g_0, w_{44})^T$, with

$$\begin{aligned} w_{41} = &- g_1 g_2^2 h_0 h_1 + g_1^2 g_2 h_0 h_2 + g_0 g_2^2 h_1^2 - 4 g_0 g_2^2 h_0 h_2 \\ &- g_0 g_1 g_2 h_1 h_2 + g_0 g_1 g_2 h_0 h_3 - g_0^2 g_2 h_1 h_3, \\ w_{42} = &- g_1^3 h_0 h_4 + g_1^2 g_2 h_0 h_3 - 2 g_0 g_2^2 h_0 h_3 - g_0 g_1 g_2 h_1 h_3 \\ &+ 4 g_0 g_1 g_2 h_0 h_4 + g_0 g_1^2 h_1 h_4 - 2 g_0^2 g_2 h_1 h_4, \\ w_{43} = &- g_0 g_2^2 h_1 h_3 - g_0 g_1 g_2 h_2 h_3 + g_0 g_1 g_2 h_1 h_4 + g_0 g_1^2 h_2 h_4 \\ &+ g_0^2 g_2 h_3^2 - 4 g_0^2 g_2 h_2 h_4 - g_0^2 g_1 h_3 h_4, \\ w_{44} = &- g_2^2 h_0 - g_0 g_2 h_2 - g_0^2 h_4. \end{aligned}$$

THE CANONICAL KUMMER SURFACE. For the canonical Kummer surface, addition by a point of order 2 is very simple. As described by Gaudry [192], each point of order 2 is some permutation of $\mathbf{0}$, followed by multiplication by -1 for either 0 or 2 of the coordinates. Similarly, addition of a point of order 2 to $P = (T_1 : T_2 : T_3 : T_4)$ is computed by applying the same permutation to the T_i, and multiplying the same coordinates by -1. For example, adding a point

$P = (T_1 : T_2 : T_3 : T_4)$ by $(c : -d : -a : b)$ we get $(T_3 : -T_4 : -T_1 : T_2)$, and the addition matrix for $(c : -d : -a : b)$ is given by

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

THE SQUARED KUMMER SURFACE. For squared Kummer surfaces, the addition matrices $W_{i,j}$ that represent $P \mapsto P + L_{i,j}$ can be computed using the algebraic method sketched in Section 2.4. This gives 4×4-matrices in terms of the Rosenhain values λ, μ, ν and the theta constants μ_i. As these matrices look rather daunting in general form yet are easily derivable from the given values, we do not put them in full form here. Their derivation and their description are given in Magma code in the file wij_squared_kummer.m.

THE ELLIPTIC KUMMER SURFACE. For elliptic Kummer surfaces we can specialise the derived $W_{i,j}$ for the squared Kummer surface to the case $\mu_3 = \mu_4 = 1$ and $\nu = \lambda \cdot \mu$, which greatly improves their visual appearance. As we use these throughout the work, we give their full versions here.

Let τ and $\tilde{\tau}$ be the roots of $x^2 - Gx + 1$. In particular, $\tilde{\tau} = 1/\tau$, and $\tau + \tilde{\tau} = \mu_1 + \mu_2$. The terms $\frac{\mu_1 - \tau}{\mu_2 - \tau}$, and their $\tilde{\tau}$ variants, appear often in these matrices. For brevity and clarity, we denote them by

$$\gamma := \frac{\mu_1 - \tau}{\mu_2 - \tau}, \quad \tilde{\gamma} := \frac{\mu_1 - \tilde{\tau}}{\mu_2 - \tilde{\tau}}.$$

Then the addition matrices are given by

$$W_{1,2} := \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad W_{3,4} := \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad W_{5,6} := \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix},$$

$$W_{1,3} := \begin{pmatrix} 1 & -\gamma & \tau \cdot \gamma & -\tau \\ \gamma & -1 & \tau & -\tau \cdot \gamma \\ \tau \cdot \gamma & -\tau & 1 & -\gamma \\ \tau & -\tau \cdot \gamma & \gamma & -1 \end{pmatrix}, \quad W_{1,4} := \begin{pmatrix} 1 & -\gamma & \tilde{\tau} \cdot \gamma & -\tilde{\tau} \\ \gamma & -1 & \tilde{\tau} & -\tilde{\tau} \cdot \gamma \\ \tilde{\tau} \cdot \gamma & -\tilde{\tau} & 1 & -\gamma \\ \tilde{\tau} & -\tilde{\tau} \cdot \gamma & \gamma & -1 \end{pmatrix},$$

$$W_{1,5} := \begin{pmatrix} 1 & -\tilde{\gamma} & -\tilde{\tau} & \tilde{\tau}\cdot\tilde{\gamma} \\ \tilde{\gamma} & -1 & -\tilde{\tau}\cdot\tilde{\gamma} & \tilde{\tau} \\ \tilde{\tau} & -\tilde{\tau}\cdot\tilde{\gamma} & -1 & \tilde{\gamma} \\ \tilde{\tau}\cdot\tilde{\gamma} & -\tilde{\tau} & -\tilde{\gamma} & 1 \end{pmatrix}, \; W_{1,6} := \begin{pmatrix} 1 & -\tilde{\gamma} & -\tau & \tau\cdot\tilde{\gamma} \\ \tilde{\gamma} & -1 & -\tau\cdot\tilde{\gamma} & \tau \\ \tau & -\tau\cdot\tilde{\gamma} & -1 & \tilde{\gamma} \\ \tau\cdot\tilde{\gamma} & -\tau & -\tilde{\gamma} & 1 \end{pmatrix},$$

$$W_{2,3} := \begin{pmatrix} 1 & -\tilde{\gamma} & \tau\cdot\tilde{\gamma} & -\tau \\ \tilde{\gamma} & -1 & \tau & -\tau\cdot\tilde{\gamma} \\ \tau\cdot\tilde{\gamma} & -\tau & 1 & -\tilde{\gamma} \\ \tau & -\tau\cdot\tilde{\gamma} & \tilde{\gamma} & -1 \end{pmatrix}, \; W_{2,4} := \begin{pmatrix} 1 & -\tilde{\gamma} & \tilde{\tau}\cdot\tilde{\gamma} & -\tilde{\tau} \\ \tilde{\gamma} & -1 & \tilde{\tau} & -\tilde{\tau}\cdot\tilde{\gamma} \\ \tilde{\tau}\cdot\tilde{\gamma} & -\tilde{\tau} & 1 & -\tilde{\gamma} \\ \tilde{\tau} & -\tilde{\tau}\cdot\tilde{\gamma} & \tilde{\gamma} & -1 \end{pmatrix},$$

$$W_{2,5} := \begin{pmatrix} 1 & -\gamma & -\tilde{\tau} & \tilde{\tau}\cdot\gamma \\ \gamma & -1 & -\tilde{\tau}\cdot\gamma & \tilde{\tau} \\ \tilde{\tau} & -\tilde{\tau}\cdot\gamma & -1 & \gamma \\ \tilde{\tau}\cdot\gamma & -\tilde{\tau} & -\gamma & 1 \end{pmatrix}, \; W_{2,6} := \begin{pmatrix} 1 & -\gamma & -\tau & \tau\cdot\gamma \\ \gamma & -1 & -\tau\cdot\gamma & \tau \\ \tau & -\tau\cdot\gamma & -1 & \gamma \\ \tau\cdot\gamma & -\tau & -\gamma & 1 \end{pmatrix},$$

$$W_{3,5} := \begin{pmatrix} 1 & 1 & -\tau & -\tilde{\tau} \\ 1 & 1 & -\tilde{\tau} & -\tau \\ \tau & \tilde{\tau} & -1 & -1 \\ \tilde{\tau} & \tau & -1 & -1 \end{pmatrix}, \qquad W_{3,6} := \begin{pmatrix} 1 & \tilde{\tau}^2 & -\tilde{\tau} & -\tilde{\tau} \\ \tilde{\tau}^2 & 1 & -\tilde{\tau} & -\tilde{\tau} \\ \tilde{\tau} & \tilde{\tau} & -1 & -\tilde{\tau}^2 \\ \tilde{\tau} & \tilde{\tau} & -\tilde{\tau}^2 & -1 \end{pmatrix},$$

$$W_{4,5} := \begin{pmatrix} 1 & \tau^2 & -\tau & -\tau \\ \tau^2 & 1 & -\tau & -\tau \\ \tau & \tau & -1 & -\tau^2 \\ \tau & \tau & -\tau^2 & -1 \end{pmatrix}, \qquad W_{4,6} := \begin{pmatrix} 1 & 1 & -\tilde{\tau} & -\tau \\ 1 & 1 & -\tau & -\tilde{\tau} \\ \tilde{\tau} & \tau & -1 & -1 \\ \tau & \tilde{\tau} & -1 & -1 \end{pmatrix}.$$

§B Kummer Pairings à la Robert

This section details a concrete instantiation to computing pairings of degree 2 on (squared) Kummer surfaces using Algorithm 5.2 by Robert [331]. We assume we want to compute the Tate pairing $t_2(L_{i,j}, Q)$ with $L_{i,j} \in \mathcal{K}[2]$ and $Q \in \mathcal{K}$, which we for now denote $t_{i,j}(Q)$. By Section A, we have $W_{i,j}$, the addition matrix with respect to $L_{i,j}$.

We first normalise the point $L := L_{i,j}$ by its first non-zero coefficient, whose index we denote $n_{i,j}$. We then apply $W_{i,j}$ to get $\widetilde{L}$, and set $\lambda_{i,j} = \widetilde{L}_{n_{i,j}} / \mu_{n_{i,j}}$. The pairing value $t_{i,j}(Q)$ for some $Q \in \mathcal{K}$ can then be computed by computing $D = L_{i,j} \pm Q$ as $W_{i,j} \cdot Q \; \lambda_Q = D_{n_{i,j}} / Q_{n_{i,j}}$. This gives us Algorithm 27.

This approach is elegant once $W_{i,j}$ and $\lambda_{i,j}$ are computed, and only requires a matrix multiplication and a Legendre symbol. The divisions can be replaced

Algorithm 27 Monodromy pairing computation.

Input: A Kummer surface $\mathcal{K}$, an index (i,j) with $1 \leq i < j \leq 6$, a point $Q \in \mathcal{K}$ normalised to $n_{i,j}$ and the precomputed $\lambda_{i,j}$.
Output: The reduced 2-Tate pairing $t_{i,j}(Q) = t_2(L_{i,j}, Q)$.
1: $D \leftarrow W_{i,j} \cdot Q$
2: $\lambda_Q \leftarrow D_{n_{i,j}} / Q_{n_{i,j}}$
3: **return** `IsSquare`$(\frac{\lambda_Q}{\lambda_{i,j}})$

by multiplications for improved performance, as this does not affect the final Legendre symbol.

All in all, to perform a monodromy pairing computation of degree 2 on a given Kummer surface, we require only the translation-by-$L_{i,j}$ maps $W_{i,j}$. For both the general Kummer surface as well as the squared Kummer surfaces, these are given in this work, and more generally they can be computed given the biquadratic forms of a Kummer surface.

§C Algebraic Derivations

In this section, we briefly describe the derivation of the polynomials F_0, F_1, F_2 and G that give a more efficient map to recover u_0, u_1 and v_0^2 on $\mathcal{J}_{\lambda,\mu,\nu}$ given a point $P \in \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$, as described in Section 2.7[14]. Any such point $P = (X_1 : X_2 : X_3 : X_4)$ in the image of $\rho^{\mathrm{Sqr}} : \mathcal{J}_{\lambda,\mu,\nu} \to \mathcal{K}^{\mathrm{Sqr}}_{\lambda,\mu,\nu}$ is given by a scalar multiple $\omega \in \mathbb{F}_p$ of the image of some $D \in \mathcal{J}_{\lambda,\mu,\nu}$ as in Equation (19). Hence, we have a system of equations

$$
\begin{aligned}
X_1 &= \omega\mu_1 \cdot \left(u_0(w_3w_5 - u_0)(w_4 + w_6 + u_1) - v_0^2\right), \\
X_2 &= \omega\mu_2 \cdot \left(u_0(w_4w_6 - u_0)(w_3 + w_5 + u_1) - v_0^2\right), \\
X_3 &= \omega\mu_3 \cdot \left(u_0(w_3w_6 - u_0)(w_4 + w_5 + u_1) - v_0^2\right), \\
X_4 &= \omega\mu_4 \cdot \left(u_0(w_4w_5 - u_0)(w_3 + w_6 + u_1) - v_0^2\right).
\end{aligned}
$$

14 The derivation in Magma code can be found in the file v2_derive.m.

with known values X_i, Kummer coefficients μ_i and Rosenhain invariants w_i, and unknowns u_0, u_1, v_0 and ω. Let $\tilde{X}_i = X_i/\mu_i$, then get

$$\begin{aligned}
f_1 &: u_0(w_3w_5 - u_0)(w_4 + w_6 + u_1) - \tilde{X}_1/\omega = v_0^2,\\
f_2 &: u_0(w_4w_6 - u_0)(w_3 + w_5 + u_1) - \tilde{X}_2/\omega = v_0^2,\\
f_3 &: u_0(w_3w_6 - u_0)(w_4 + w_5 + u_1) - \tilde{X}_3/\omega = v_0^2,\\
f_4 &: u_0(w_4w_5 - u_0)(w_3 + w_6 + u_1) - \tilde{X}_4/\omega = v_0^2.
\end{aligned}$$

with the differences $f_{i-j} = f_i - f_j = 0$ for all $1 \le i < j \le 4$ give us six equations. By assuming u_0^2, u_0u_1, u_0 and ω as independent linear variables, this gives us a matrix F such that $F(u_0^2, u_0u_1, u_0, \omega)^T = (f_{i-j})_{1\le i<j\le 4}$. After row-echelon reduction of F, we find simple equations

$$\begin{aligned}
u_0^2 &= h_1(\tilde{X}_i, w_i)\cdot 1/\omega,\\
u_0u_1 &= h_2(\tilde{X}_i, w_i)\cdot 1/\omega,\\
u_0 &= h_3(\tilde{X}_i, w_i)\cdot 1/\omega
\end{aligned}$$

for some polynomials h_i in the coefficients $\tilde{X}_i$ and w_i, and as we must have $(u_0^2) = (u_0)^2$, the first as the squared variable – the second as the variable squared, we get $\omega = h_3^2/h_1$. Thus, we derive equations for u_0, u_1 and ω from h_1, h_2 and h_3. Finally, by any of the f_i, we then similarly recover an equation for v_0^2. By gathering common factors, we find the polynomials F_0, F_1, F_2 and G defined in the coefficients μ_i, X_i, w_i.

§D Constants for Scaling Maps

For kernels $\langle L_{i,j}, L_{k,\ell}\rangle$ where

$$(i,j,k,\ell) \in \left\{ \begin{array}{l} (2,3,4,5),(1,3,4,5),(1,4,3,6),(2,4,3,6),\\ (2,3,4,6),(1,3,4,6),(1,4,3,5),(2,4,3,5) \end{array} \right\},$$

the required square roots can be extracted from the 4-torsion points lying above the kernel points. More precisely, let R, S be such that

$$[2]R = L_{i,j},\ [2]S = L_{k,\ell},$$

and let $h_{P,i}$ be the i-th coordinate of $\mathtt{H}(P)$. For the sake of clarity, we suppose that $L_{i,j}$ and $L_{k,\ell}$ are of the form $(1\colon \star\colon 0\colon 0)$ and $(1\colon 0\colon \star\colon 0)$, respectively. The other cases follow similarly.

We compute the scaling value $(1/A : 1/B : 1/C : 1/D)$ using the points lying above the kernel by noting that

$$\frac{A}{B} = c_1 \frac{h_{R,1}}{h_{R,2}}, \ \frac{A}{C} = c_2 \frac{h_{S,1}}{h_{S,3}}, \text{ and } \frac{C}{D} = c_3 \frac{h_{R,3}}{h_{R,4}},$$

for some constants $c_i \in \{1, -i\}$. From this we can derive $\frac{A}{B}$, $\frac{A}{C}$, and $\frac{A}{D}$ to get

$$\left(\frac{1}{A} : \frac{1}{B} : \frac{1}{C} : \frac{1}{D}\right) = \Big(h_{R,2}h_{R,4}h_{S,3}\colon c_1 h_{R,1}h_{R,4}h_{S,3}\colon \\ \colon c_2 h_{R,2}h_{R,4}h_{S,1}\colon c_2c_3 h_{R,2}h_{R,3}h_{S,1}\Big).$$

The constants for each kernel are then given as follows:

- If $(i,j,k,\ell) = (2,3,4,5)$ or $(1,4,3,6)$, then $(c_1, c_2, c_2c_3) = (+1, +1, +1)$.
- If $(i,j,k,\ell) = (1,3,4,5)$ or $(2,4,3,6)$, then $(c_1, c_2, c_2c_3) = (+1, -i, -i)$.
- If $(i,j,k,\ell) = (2,3,4,6)$ or $(1,4,3,5)$, then $(c_1, c_2, c_2c_3) = (-i, +1, -i)$.
- If $(i,j,k,\ell) = (1,3,4,6)$ or $(2,4,3,5)$, then $(c_1, c_2, c_2c_3) = (-i, -i, +1)$.

TOWARDS A BRIGHTER FUTURE!

At the time of writing this thesis, 2D-variants of SQIsign have made a meteoric rise: almost certainly 2D-SQIsign will be here to stay, making it a solid contender in the second round of the NIST competition. It is unrivaled in combined public-key and signature size, and we may suspect signing and verification times to still improve, e.g. by including some ideas from Chapter X. We describe a few movements that may impact SQIsign's slope of tomorrow.

Constant-time Signing.

The biggest challenge for the current signing procedure is making it constant time. There are many challenges ahead, and initial research has already shown the vulnerability of current approaches [219]. This is not surprising, as many of the techniques used in SQIsign using quaternions are completely new in the isogeny landscape. Nevertheless, we should not underestimate this challenge: we have seen in Part 1 that some techniques or even complete schemes may suffer a dramatic hit in performance when we require them to be constant time.

One-dimensional SQIsign.

With current signing times, 1D-SQIsign simply cannot compete in practicality with 2D-SQIsign. However, as we have seen in Chapter IX and Chapter X, research in signing for 1D-SQIsign is a useful endeavour: signing for Après-primes has suddenly become much more practical [303] making current verification times practically equal between 1D- and 2D SQIsign, even though 1D-SQIsign has not yet reached the theoretical lower limit of the degree of the verification isogeny. Future research on KLPT could potentially lower the length of this isogeny by a significant factor, giving 1D-SQIsign an edge over 2D-SQIsign. Furthermore, the parallelizability of 1D-verification has significant impact on 1D-SQIsign's practicality. With the techniques from Chapter XI, we may explore similar approaches for 2D-SQIsign.

Probing Techniques.

The probing techniques using Tate pairing profiles, realised independently by Robert [333] and us (Chapter XI), seem stronger than the community currently realises. For example[15], a point $P \in E[\ell]$ with ℓ a large prime defines an isogeny $\varphi : E \to E/\langle P \rangle$. Even though we cannot compute φ, we can rather easily compute the generalized Tate pairing t_φ and gain information on $E/\langle P \rangle$, such as the volcano level [216]!

Furthermore, the recasting of 'well-known' results in terms of profiles of Tate pairings, as done in Chapter IX, Chapter X and Chapter XI but applicable also to Chapter VIII, explores the deeper connection between kernels and cokernels on a very practical level. We hope to see many more applications of these techniques, beyond the current literature [216, 331, 333] that we are aware of.

We may therefore draw the following conclusion for this part of the thesis: SQIsign has a bright two-dimensional future ahead, with current developments going at breakneck speed. Still, this should not cause tunnel vision in the isogeny community, as research into one-dimensional signing techniques (using higher-dimensional tools) could potentially be a gamechanger. For both variants, especially if signing can be made constant-time without significantly degrading performance, SQIsign will be a major player for years to come.

15 This example was communicated to me by Damien Robert.

Part 3

ISOMETRIES (MCE)

THE LANDSCAPE OF CODE EQUIVALENCE

Love your team,
not your scheme.

At the start of this thesis, Matrix Code Equivalence, as defined in Problem III.1, was not well-studied for cryptographic applications. Linear Code Equivalence, using the Hamming metric, sprouted LESS [56] and later LESS-FM [30]. By studying the hardness of MCE, a similar scheme should be possible using the rank metric, which could then potentially have better characteristics than its little brother LESS.

Some works [38, 138] have studied the hardness of this problem. Crucially, Couvreur, Debris-Alazard, and Gaborit [138] conclude that the *vector rank metric* seems unsuitable for cryptographic applications whose hardness relies on isometries, as the associated equivalence problem is solvable with a probabilistic polynomial-time algorithm.

We thus turn ourselves solely to the *matrix rank metric*. In Chapter XII, we extend the analysis of MCE: We show how MCE fits in the framework of other code-based and multivariate-based problems[16], and use this connection to apply graph-based algorithms with complexity $\tilde{O}(q^{\min(m,n,k)})$ for k-dimensional random codes $\mathcal{C} \subseteq \mathbb{F}_q^{m \times n}$.

Using this improved upper bound from Chapter XII, we design a signature scheme using MCE as a group action in Chapter XIII, as sketched in Section I.2. Applying both general Sigma protocol optimisations, as well as optimisations specific to MCE, we construct Matrix Equivalence Digital Signature (MEDS). We furthermore improve cryptanalysis on MCE with new algebraic attacks.

MEDS, as specified in Chapter XIII, is close to the scheme submitted[17] to the NIST Call for Additional Signatures [357]. There are a few advances: cryptanalysis improved, using more algebraic structure [289, 318] and signature size decreased, by a clever technique introduced by Chou, Niederhagen, Ran, and Samardjiska [111]. We are excited to see further developments in both cryptanalysis and scheme optimisation.

16 A result obtained independently by Grochow and Qiao [202].
17 Specifications available at https://meds-pqc.org.

The general Sigma protocol optimizationss applied to MEDS can be studied for any cryptographic group action. There are many works that discuss such generic techniques [30, 53, 144, 227, 362] but it is often hard to compare the usefulness of the combinations of these techniques for a specific scheme. We tackle this problem in Chapter XIV, by presenting a unified taxonomy of these techniques, including novel ones, and analyzing the performance of their combinations. We can then easily apply this framework to existing schemes, including MEDS, to optimize their performance.

XII

HARDNESS ESTIMATES OF THE CODE EQUIVALENCE PROBLEM IN THE RANK METRIC

In this chapter, we analyze the hardness of the Matrix Code Equivalence (MCE) problem for matrix codes endowed with the rank metric, and provide the first algorithms for solving it. We do this by making a connection to another well-known equivalence problem from multivariate cryptography - the Isomorphism of Polynomials (IP). Under mild assumptions, we give tight reductions from MCE to the homogenous version of the Quadratic Maps Linear Equivalence (QMLE) problem, and vice versa. Furthermore, we present reductions to and from similar problems in the sum-rank metric, showing that MCE is at the core of code equivalence problems. On the practical side, using birthday techniques known for IP, we present two algorithms: a probabilistic algorithm for MCE running in time $q^{\frac{2}{3}(n+m)}$ up to a polynomial factor, and a deterministic algorithm for MCE with roots, running in time $q^{\min\{m,n,k\}}$ up to a polynomial factor. Lastly, to confirm these findings, we solve randomly-generated instances of MCE using these two algorithms.

§1 Introduction

Given two mathematical objects of the same type, an equivalence problem asks the question whether there exists an equivalence map between these objects –

This chapter is based on the paper

> Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. "Hardness estimates of the code equivalence problem in the rank metric". In: *Designs, Codes and Cryptography* 92.3 (2024), pp. 833–862.

I have only made small editorial changes to ensure consistency and improve clarity. At the time of writing this thesis, cryptanalysis of MCE has significantly advanced [109, 289, 318].

and how to find it – that preserves some important property of the objects. These kind of problems come in different flavors depending on the objects – groups, graphs, curves, codes, quadratic forms, etc. – and quite often the interesting maps are isomorphisms or isometries. Interestingly, equivalence problems are one of the core hard problems underlying the security of many public-key cryptosystems, especially post-quantum ones. Many multivariate and code-based systems employ an equivalence transformation as a hiding technique, and thus intrinsically rely on the assumption that a particular equivalence problem is intractable, for example [55, 83, 153, 160, 269, 295, 305] . In addition, quite remarkably, a hard equivalence problem gives rise to a Sigma protocol and, through the Fiat-Shamir transform, a provably secure digital signature scheme [178]. This idea has been revisited many times, being the basis of several signature schemes [30, 56, 144, 147, 195, 305]. Three such schemes actually appeared during the writing of this manuscript [56, 165, 362] as a result of NIST's announcement for an additional fourth round on signatures in the post quantum standardization process [291]. Understanding the hardness of these equivalence problems is an essential task in choosing appropriate parameters that attain a certain security level of these cryptographic schemes.

One of these problems is the Linear Code Equivalence problem, which given two codes (with the Hamming metric), asks for an isometry (equivalence transformation that preserves the metric) that maps one code to the other. It was first studied by Leon [256] who proposed an algorithm that takes advantage of the Hamming weight being invariant under monomial permutations. It was improved very recently by Beullens [51] using collision-based techniques. Sendrier [349] proposed another type of algorithm, the Support Splitting Algorithm (SSA), that is exponential in the dimension of the hull (the intersection of a code and its dual). Interestingly, in low characteristic, random codes have very small hull, rendering the problem easy.

In this work, we focus on the code equivalence problem, for matrix codes endowed with the rank metric - *Matrix Code Equivalence* (MCE). Evaluating the hardness of this problem is only natural – rank-based cryptography has become serious competition for its Hamming-based counterpart, showing superiority in key sizes for the same security level [12, 13, 37, 271]. MCE, and variations of it, has been introduced by Berger [38], but it was only recently that the first concrete statements about its hardness were shown in two concurrent independent works publicly available as preprints [138, 202][1]. Couvreur, Debris-Alazard,

1 The two works use different techniques and terminology, and seem to be mutually unaware of the line of work preceding the other. In [202] the MCE problem is referred to as Matrix Space Equivalence problem and 3-Tensor Isomorphism problem.

and Gaborit [138] show that MCE is at least as hard as the (Monomial) Code Equivalence problem in the Hamming metric, while for only right equivalence, or when the codes are $\mathbb{F}_{q^m}$-linear, the problem becomes easy. Grochow and Qiao [202] show the same reduction from (Monomial) Code Equivalence to MCE but using a completely different technique of linear algebra coloring gadgets which makes the reduction looser than the one in [138].

Contributions

In this paper, we investigate the theoretical and practical hardness of the Matrix Code Equivalence (MCE) problem. Our contributions can be summarized as follows:

First, we link in a straightforward manner the MCE problem to hard problems on systems of polynomials by showing that MCE is polynomial-time equivalent to the Bilinear Maps Linear Equivalence (BMLE) problem. We then extend this result by proving that MCE is polynomial-time equivalent to the Quadratic Maps Linear Equivalence (QMLE) problem, under a mild assumption of trivial automorphism groups of the codes in question. While our technique fails to give a proof without this assumption, we consider it to be reasonable for randomly generated codes and for cryptographic purposes. As the QMLE problem is considered to be the hardest equivalence problem for systems of multivariate polynomials, it is essential to understand under which conditions MCE and QMLE reduce to one another. Note that previous work[2] requires much stronger assumptions for related results [36, 182, 202], such as algebraically closed fields or the existence of square or third roots. Our reduction to QMLE is tight and gives a tight upper bound on the hardness of MCE. Furthermore, it is very simple, thus establishing a connection between code equivalence problems and polynomial equivalence problems that is usable in practice. This is the basis of our contributions to the practical hardness of MCE.

Second, using similar techniques, and under the same assumptions, we show that MCE is polynomial-time equivalent to other code equivalence problems, such as Matrix Sum-Rank Code Equivalence Problem, and at least as hard as the Vector Sum-Rank Code Equivalence Problem. All these connections and our results are visualized in Figure 25.

On the practical side, we provide the first two non-trivial algorithms for solving MCE using the connection to QMLE. The first algorithm is a general-

2 We were made aware of this line of work by one of the authors after our results were first presented at WCC 2022.

ization of a known birthday-based algorithm for QMLE [68, 69] for systems of polynomials with the same number of variables as equations. We show that this algorithm extends to different invariance properties and code dimensions, which helps us prove complexity of $q^{\frac{2}{3}(n+m)}$ up to a polynomial factor for MCE for $m \times n$ matrix codes. The algorithm is probabilistic with success probability that can be made arbitrarily close to 1, and can be used for code dimensions up to $2(m+n)$. For larger dimensions, the complexity becomes $q^{(n+m)}$ up to a polynomial factor, but the algorithm is deterministic. The birthday-based algorithm for QMLE [69] assumed existence of a polynomial-time solver for the inhomogeneous variant of QMLE to achieve these complexities. Interestingly, due to the specific instances of the inhomogeneous QMLE arising from the collision search, the problem seems to be much harder than for random instances – a fact previously overlooked in [69]. In contrast, [68] uses a non-polynomial estimate for this solver. We analyse the most recent results regarding such solvers, and show that for parameter sets of cryptographical interest the above complexities hold, even if such solvers do not achieve polynomial time.

Our second algorithm uses the bilinear structure of the polynomials arising from MCE. Because matrix codes show symmetry between the parameters, as given in Lemma 12, the complexity of solving MCE using this result and Algorithm 29 becomes $q^{\min\{m,n,k\}}$ up to a polynomial factor. The algorithm is deterministic and does not require a polynomial-time solver for the inhomogeneous QMLE instance, but the weaker assumption that the solver has a complexity of $\mathcal{O}(q^{\min\{m,n,k\}})$ at most. This general result, valid for any m,n, and k, is summarized in our main result Theorem 25.

Lastly, to verify the results and performance of these algorithms in practice, we have implemented both and solved randomly generated instances of MCE for different parameter sets. The results of these experiments show that our assumptions are reasonable and the above complexities hold. Our implementations are open source and available at:

https://github.com/mtrimoska/matrix-code-equivalence

A NOTE ON REDUCTIONS

Our results in Section 3 use the following standard notion of Turing reduction.

Definition 1. Given two computational problems $\mathsf{P}(\mathsf{p}, S)$ and $\mathsf{P}'(\mathsf{p}', S')$, with inputs coming from sets S and S' respectively, we say that $\mathsf{P}(\mathsf{p}, S)$ reduces to $\mathsf{P}'(\mathsf{p}', S')$ if there exists a probabilistic polynomial-time oracle machine $\mathcal{B}$ such

that for every oracle $\mathcal{A}$ that solves $\mathsf{P}'(\mathsf{p}', S')$ on all inputs from S', $\mathcal{B}^{\mathcal{A}}$ ($\mathcal{B}$ given access to $\mathcal{A}$) solves $\mathsf{P}(\mathsf{p}, S)$ on all inputs from S.

Note that our reductions are meaningful only as worst-case to worst-case, and therefore in the definition we include the statement that the oracles solve the problems on all inputs. On the other hand, we do not always require the oracle $\mathcal{A}$ to be able to solve P' on the entire universe U' of inputs in order for $\mathcal{B}^{\mathcal{A}}$ to be able to solve P on the entire universe U of inputs. When this is the case, it will be emphasized through the definition of the input sets S and S'. These restrictions, however, can not be used to show a stronger statement such as worst-case to average-case reduction.

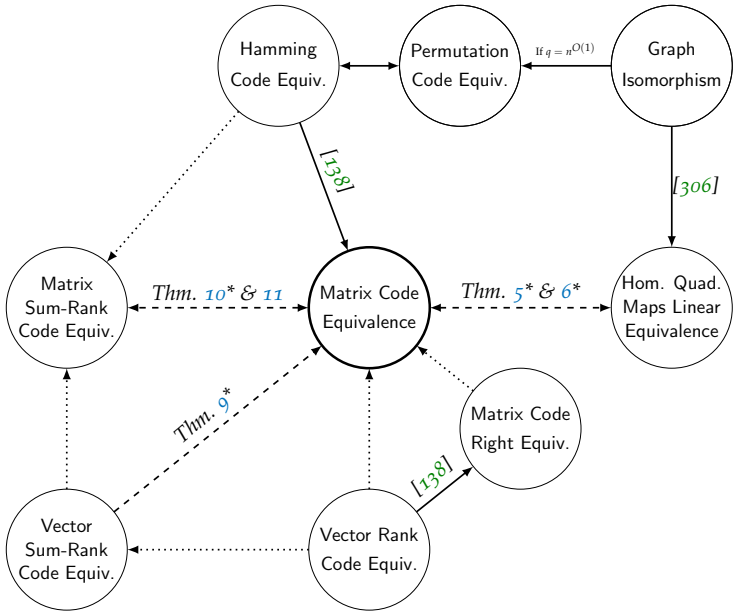


Figure 25: Reductions around Matrix Code Equivalence. Dashed arrows are contributions from this work, dotted arrows are trivial reductions. "A $\longrightarrow$ B" means that "Problem A reduces to Problem B in polynomial time". Results with * assume trivial automorphism groups.

§2 Preliminaries

Chapter III gives a general introduction to codes and their isometries. We repeat here the statement of MCE, to provide context to subsequent definitions.

Problem 1. MCE$(n, m, k, \mathbb{F}_q^{m\times n\times k})$:
Input: Two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subseteq \mathbb{F}_q^{m\times n}$
Question: Find – if any – a map $(\mathbf{A}, \mathbf{B})$, where $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mathbf{ACB} \in \mathcal{D}$.

This is the computational version of MCE which, similarly to its counterpart in the Hamming metric [28, 30, 56], seems to be more interesting for cryptographic applications than its decisional variant. We will thus be interested in evaluating the practical hardness only of MCE, and present algorithms only for MCE and not its decisional variant.

When $n = m$, if there exists an isometry of the form $(\mathbf{A}, \mathbf{A}^\top) : \mathbf{C} \to \mathbf{ACA}^\top$, where $\mathbf{A} \in \mathrm{GL}_m(q)$, we will say that $\mathcal{C}$ and $\mathcal{D}$ are *congruent*, which is a direct generalization of the notion of congruence for matrices.

It is also interesting to consider the following variant of MCE:

Problem 2. MCEbase$(n, m, k, \mathbb{F}_q^{m\times n\times k})$:
Input: The bases $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ of two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subseteq \mathbb{F}_q^{m\times n}$
Question: Find – if any – a map $(\mathbf{A}, \mathbf{B})$, where $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{C}^{(i)}$, it holds that $\mathbf{AC}^{(\mathbf{i})}\mathbf{B} = \mathbf{D}^{(i)}$.

Intuitively, MCEbase seems easier than MCE, and as a matter of fact, we will show later that most random instances are solvable in polynomial time. Another variant of the MCE problem is the *Matrix Codes Right Equivalence* problem (MCRE) (left equivalence could be defined similarly):

Problem 3. MCRE$(n, m, k, \mathbb{F}_q^{m\times n\times k})$:
Input: Two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subseteq \mathbb{F}_q^{m\times n}$
Question: Find – if any – $\mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mathbf{CB} \in \mathcal{D}$.

It has been shown in [138] that MCE is at least as hard as code equivalence in the Hamming metric, *Hamming Code Equivalence* (HCE), also known as Linear or Monomial Equivalence. Interestingly, the same paper shows that MCRE is actually easy and can always be solved in probabilistic-polynomial time.

For *vector rank codes* $\mathcal{C} \subset \mathbb{F}_{q^m}^n$, isometries are similar to the case of matrix codes. We get the *Vector Rank Code Equivalence* (VRCE) problem.

Problem 4. $\mathsf{VRCE}(n, m, k, \mathbb{F}_q^{m\times n\times k})$:
Input: Two k-dimensional vector rank codes $\mathcal{C}, \mathcal{D} \subset \mathbb{F}_{q^m}^n$
Question: Find – if any – a matrix $\mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{c} \in \mathcal{C}$, it holds that $\mathbf{cB} \in \mathcal{D}$.

Given a vector rank code $\mathcal{C} \subset \mathbb{F}_{q^m}^n$ and a basis Γ for $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, each vector $\mathbf{c} \in \mathcal{C}$ can be expanded to a matrix $\Gamma(\mathbf{c}) \in \mathbb{F}_q^{m\times n}$, giving rise to a matrix code $\Gamma(\mathcal{C})$. For any two bases Γ and Γ', an equivalence between two vector rank codes $\mathcal{C}$ and $\mathcal{D}$ implies an equivalence between the matrix codes $\Gamma(\mathcal{C})$ and $\Gamma'(\mathcal{D})$ [198], so VRCE is trivially a subproblem of MCE. However, using the $\mathbb{F}_{q^m}$-linearity of vector rank codes, VRCE reduces *non-trivially* to MCRE [138].

2.1 SYSTEMS OF QUADRATIC POLYNOMIALS

Let $\mathcal{P} = (p_1, p_2, \ldots, p_k) : \mathbb{F}_q^N \to \mathbb{F}_q^k$ be a vectorial function of k quadratic polynomials in N variables $x_1, \ldots, x_N$, where

$$p_s(x_1, \ldots, x_N) = \sum_{1\leqslant i\leqslant j\leqslant N} \gamma_{ij}^{(s)} x_i x_j + \sum_{i=1}^{N} \beta_i^{(s)} x_i + \alpha^{(s)},$$

with $\gamma_{ij}^{(s)}, \beta_i^{(s)}, \alpha^{(s)} \in \mathbb{F}_q$ for $1 \leqslant s \leqslant k$.

It is common to represent the quadratic homogeneous part of the components of $\mathcal{P}$ using symmetric matrices, but unfortunately, a natural correspondence only exists for finite fields of odd characteristic. For the case of even characteristic, we will adopt a technical representation that is a common workaround in the literature of multivariate cryptography and will still be good for our purposes.

Let $p(x_1, \ldots, x_N) = \sum_{1\leqslant i\leqslant j\leqslant N} \gamma_{ij} x_i x_j$ be a quadratic form over $\mathbb{F}_q$. Then, for fields of odd characteristic, we can associate to p a symmetric matrix $\mathbf{P} = \overline{\mathbf{P}} + \overline{\mathbf{P}}^\top$, where $\overline{\mathbf{P}}$ is an upper triangular matrix with coefficients $\overline{\mathbf{P}}_{ij} = \gamma_{ij}/2$ for $i \leqslant j$. Clearly, there is a one-to-one correspondence between quadratic forms and symmetric matrices, since for $\mathbf{x} = (x_1, \ldots, x_N)$ it holds that

$$p(x_1, \ldots, x_N) = \mathbf{x}\mathbf{P}\mathbf{x}^\top. \tag{30}$$

Now, all operations on quadratic forms naturally transform into operations on matrices since the one-to-one correspondence between quadratic forms and

symmetric matrices is in fact an isomorphism. Note that, in matrix form, a change of variables (basis) works as:

$$p(\mathbf{xS}) = \mathbf{xSPS}^\top \mathbf{x}^\top. \tag{31}$$

In what follows, we will interchangeably work with both the quadratic form p and its matrix representation $\mathbf{P}$.

Over fields $\mathbb{F}_q$ of even characteristic, the relation (30) does not hold, since for a symmetric matrix $\mathbf{P}$ we have $(\mathbf{P}_{ij} + \mathbf{P}_{ji})x_i x_j = 2\mathbf{P}_{ij} x_i x_j = 0$. The nice correspondence between quadratic forms and symmetric matrices is broken, but we would still like to be able to use some sort of matrix representation for quadratic forms. Thus, in even characteristic we associate to p a symmetric matrix $\mathbf{P} = \overline{\mathbf{P}} + \overline{\mathbf{P}}^\top$, where $\overline{\mathbf{P}}$ is an upper triangular matrix with coefficients $\overline{\mathbf{P}}_{ij} = \gamma_{ij}$ for $i \leqslant j$.

This representation can also be used in odd characteristic when it comes to linear operations and changes of basis, as the correspondence $p \mapsto \mathbf{P}$ is a homomorphism. However, it is not a bijection, since all the quadratic forms in the set $\{ \sum_{1 \leqslant i < j \leqslant N} \gamma_{ij} x_i x_j + \sum_{1 \leqslant i \leqslant N} \gamma_{ii} x_i^2 \mid \gamma_{ii} \in \mathbb{F}_q \}$ map to the same symmetric matrix (note that it has zeros on the diagonal). In practical, cryptographic applications, this typically does not pose a problem, and can be overcome. The same holds for our purpose of solving equivalence problems for systems of quadratic polynomials.

Differential of quadratic functions.

Given a non-zero $\mathbf{a} \in \mathbb{F}_q^N$, an object directly related to the symmetric matrix representation of quadratic forms is the *differential* of $\mathcal{P}$ at $\mathbf{a}$ (see [163, 179]):

$$D_{\mathbf{a}}\mathcal{P} : \mathbb{F}_q^N \to \mathbb{F}_q^k, \quad \mathbf{x} \mapsto \mathcal{P}(\mathbf{x} + \mathbf{a}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{a}).$$

Note that the differential of a quadratic function is closely related to the bilinear form $\beta(\mathbf{x}, \mathbf{y}) = q(\mathbf{x} + \mathbf{y}) - q(\mathbf{x}) - q(\mathbf{y})$ associated to a quadratic form q. In this work we are especially interested in the kernel of $D_{\mathbf{a}}\mathcal{P}$, as $D_{\mathbf{a}}\mathcal{P}(\mathbf{x}) = 0$ implies $\mathcal{P}(\mathbf{x} + \mathbf{a}) = \mathcal{P}(\mathbf{x}) + \mathcal{P}(\mathbf{a})$, that is, $\mathcal{P}$ acts linearly on the kernel of $D_{\mathbf{a}}\mathcal{P}$.

2.2 ISOMORPHISM OF POLYNOMIALS

The Isomorphism of Polynomials (IP) problem (or Polynomial Equivalence (PE) [173]) was first defined by Patarin in [305] for the purpose of designing a

"graph isomorphism"-like identification scheme and a digital signature scheme using the Fiat-Shamir transform [178]. It is defined as follows.

Problem 5. $\mathsf{IP}(N, k, \mathbb{F}_q[x_1, \ldots, x_N]^k \times \mathbb{F}_q[x_1, \ldots, x_N]^k)$:
Input: Two k-tuples of multivariate polynomials $\mathcal{F} = (f_1, f_2, \ldots, f_k)$, $\mathcal{P} = (p_1, p_2, \ldots, p_k) \in \mathbb{F}_q[x_1, \ldots, x_N]^k$.
Question: Find – if any – $(\mathbf{S}, \mathbf{s}) \in \mathrm{AGL}_N(q), (\mathbf{T}, \mathbf{t}) \in \mathrm{AGL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}) = \mathcal{F}(\mathbf{x}\mathbf{S} + \mathbf{s})\mathbf{T} + \mathbf{t}. \tag{32}$$

The variant of the problem where $(\mathbf{T}, \mathbf{t})$ is trivial is known as the Isomorphism of Polynomials with one secret (IP1S), whereas if $\mathcal{P}$ and $\mathcal{F}$ are quadratic and both $\mathbf{s}$ and $\mathbf{t}$ are the null vector, the problem is known as the Quadratic Maps Linear Equivalence (QMLE) problem.

The decisional version of IP is not $\mathcal{NP}$-complete [306], but it is known that even IP1S is at least as difficult as the Graph Isomorphism problem [306]. The IP problem has been investigated by several authors, initially for the security of the C^* scheme [306]. In [309], it was shown that IP1S is polynomially solvable for most of the instances with $k \geq N$, and Bouillaguet, Faugère, Fouque, and Perret [67] gave an algorithm with running time of $\mathcal{O}(N^6)$ for random instances of the IP1S problem, thus fully breaking Patarin's identification scheme [305]. Patarin, Goubin, and Courtois [306] gave an algorithm for solving the general IP, called To-and-Fro, that runs in time $\mathcal{O}(q^{2N})$ for $q > 2$ and $\mathcal{O}(q^{3N})$ for $q = 2$. It was noted in [69] that the algorithm is only suited for bijective mappings $\mathcal{F}$ and $\mathcal{P}$. Getting rid of the bijectivity constraint has been explored in [68] with the conclusion that the proposed workarounds either have a non-negligible probability of failure or it is unclear how greatly they affect the complexity of the algorithm.

Regarding QMLE, the linear variant of IP, an empirical argument was given in [173] that random inhomogeneous instances are solvable in $\mathcal{O}(N^9)$ time, but a rigorous proof for this case still remains an open problem. Under this assumption, the same paper provides an algorithm of complexity $\mathcal{O}(N^9 q^N)$ for the homogeneous case which is considered the hardest, that was subsequently improved to $\mathcal{O}(N^9 q^{2N/3})$ in [69]. Both works reduce a homogenous instance to an inhomogenous instance and assume the obtained inhomogeneous instance behaves as a random instance. This, however, is a wrong assumption which questions the claimed complexity of the algorithm.

In this work, we will be interested in the homogeneous variant of QMLE, that we denote hQMLE, as the hardest and most interesting instance of QMLE.

Formally, the hQMLE problem is defined as follows.

Problem 6. hQMLE$(N, k, \mathbb{F}_q[x_1, \ldots, x_N]^k \times \mathbb{F}_q[x_1, \ldots, x_N]^k)$:
Input: Two k-tuples of homogeneous multivariate polynomials of degree 2

$$\mathcal{F} = (f_1, f_2, \ldots, f_k),\ \mathcal{P} = (p_1, p_2, \ldots, p_k) \in \mathbb{F}_q[x_1, \ldots, x_N]^k.$$

Question: Find – if any – a map $(\mathbf{S}, \mathbf{T})$ where $\mathbf{S} \in \mathrm{GL}_N(q), \mathbf{T} \in \mathrm{GL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}) = (\mathcal{F}(\mathbf{xS}))\mathbf{T}. \tag{33}$$

Interestingly, the case of $k = 1$, which we will call Quadratic Form Equivalence (QFE) has been completely solved for more than 80 years already in the works of "Theorie der quadratischen Formen in beliebigen Korpern" [365] and "Untersuchungen über quadratische Formen in Korpern der Charakteristik 2, I" [369]. It is known that every quadratic form is equivalent to a unique canonical diagonal (for odd characteristic) or block diagonal (for even characteristic) form which can be obtained in time $\mathcal{O}(N^3)$. Thus, QFE can also be solved in time $\mathcal{O}(N^3)$ by first calculating the transformations to the canonical forms of the two quadratic forms. If the canonical forms are the same, by composition, one can find the equivalence. If the canonical forms are not the same, the two quadratic forms are not equivalent.

In this work, we also consider a variant of QMLE where $\mathcal{F}$ and $\mathcal{P}$ are bilinear forms. We call this problem Bilinear Maps Linear Equivalence (BMLE). In this variant, $\mathcal{F}$ and $\mathcal{P}$ are k-tuples of homogeneous polynomials of degree 2 in two sets of variables $\mathbf{x} = [x_1, \ldots, x_n]$ and $\mathbf{y} = [y_1, \ldots, y_m]$, where each monomial is of the form $x_i y_j$. Formally, the BMLE problem is defined as follows.

Problem 7. BMLE$(n, m, k, \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k \times \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k)$:
Input: Two k-tuples of bilinear forms

$$\mathcal{F} = (f_1, f_2, \ldots, f_k),\ \mathcal{P} = (p_1, p_2, \ldots, p_k) \in \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k$$

Question: Find – if any – a triplet $(\mathbf{S}_1, \mathbf{S}_2, \mathbf{T})$ where $\mathbf{S}_1 \in \mathrm{GL}_n(q), \mathbf{S}_2 \in \mathrm{GL}_m(q)$, $\mathbf{T} \in \mathrm{GL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}, \mathbf{y}) = (\mathcal{F}(\mathbf{xS}_1, \mathbf{yS}_2))\mathbf{T}. \tag{34}$$

The inhomogenous versions of QMLE and BMLE will be referred to as inhQMLE and inhBMLE respectively. We write inh(Q/B)MLE when it does not matter if we are referring to the quadratic or the bilinear version.

§3 How Hard is MCE?

In this section, we investigate the relation of the MCE problem to other known problems that we notably split in two groups – equivalence problems for systems of multivariate quadratic polynomials and equivalence problems for codes.

3.1 RELATIONS TO EQUIVALENCE PROBLEMS FOR QUADRATIC POLYNOMIALS

We start with establishing a straightforward link between MCE and polynomial equivalence problems by proving that the MCE and BMLE problems are equivalent.

Theorem 2. The MCE problem is at least as hard as the BMLE problem.

Proof. In order to prove our claim, we need to show that an oracle $\mathcal{A}$ solving any instance of the MCE problem can be transformed in polynomial time to an oracle $\mathcal{B}$ solving any instance of the BMLE problem.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{BMLE}}(\mathcal{F}, \mathcal{P})$ of $\mathsf{BMLE}(n, m, k, \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k \times \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k)$, where $\mathcal{F} = (f_1, f_2, \ldots, f_k)$, $\mathcal{P} = (p_1, p_2, \ldots, p_k) \in \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k$ are k-tuples of bilinear forms. Without loss of generality, we assume $f_1, f_2, \ldots, f_k$ (respectively $p_1, p_2, \ldots, p_k$) to be linearly independent. $\mathcal{B}$ can efficiently construct an instance of the MCE problem as follows.

$\mathcal{B}$ represents the components f_s and p_s, $s \in \{1, \ldots, k\}$ of the mappings $\mathcal{F}$ and $\mathcal{P}$ as $m \times n$ matrices $\mathbf{F}^{(s)}$ and $\mathbf{P}^{(s)}$, where $\mathbf{F}^{(s)}_{i,j}$ equals the coefficient of $x_i y_j$ in f_s and $\mathbf{P}^{(s)}_{i,j}$ equals the coefficient of $x_i y_j$ in p_s. Taking $(\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(k)})$ to be a basis of a matrix code $\mathcal{C}$ and $(\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(k)})$ a basis of a matrix code $\mathcal{D}$, $\mathcal{B}$ obtains an instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ of $\mathsf{MCE}(n, m, k, \mathbb{F}_q^{m \times n \times k})$.

$\mathcal{B}$ gives the instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ as an input to $\mathcal{A}$, who outputs either a solution $(\mathbf{A}, \mathbf{B})$ to the MCE instance (in the case of a positive instance) or outputs that there is no solution (in the case of a negative instance). In the latter case, $\mathcal{B}$ immediately outputs: no solution. In the former case, $\mathcal{B}$ constructs the matrices $\mathbf{R}^{(s)} = \mathbf{A}\mathbf{F}^{(s)}\mathbf{B} \in \mathcal{D}$ and solves the following system of equations in the variables $t_{i,j}$:

$$\sum_{j=1}^{k} t_{j,i} \cdot \mathbf{R}^{(j)} = \mathbf{P}^{(i)}, \quad \text{for all } i \in \{1, \ldots, k\} \tag{35}$$

The system always has a solution, since $(\mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(k)})$ is a basis of the code $\mathcal{D}$.

$\mathcal{B}$ sets $\mathbf{T} = \left(t_{i,j}\right)$, and outputs $(\mathbf{A}, \mathbf{B}^\top, \mathbf{T})$ as the solution to $\mathcal{I}_{\mathsf{BMLE}}(\mathcal{F}, \mathcal{P})$. $\mathcal{B}$ succeeds whenever $\mathcal{A}$ succeeds and the reduction runs in time $\mathcal{O}(k^6)$.

□

Theorem 3. BMLE is at least as hard as MCE.

Proof. We proceed similarly as in the other direction – Given an oracle $\mathcal{A}$ solving any instance of BMLE, we can construct in polynomial time an oracle $\mathcal{B}$ with access to $\mathcal{A}$ that can solve any instance of MCE.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ of $\mathsf{MCE}(n, m, k, \mathbb{F}_q^{m \times n \times k})$. $\mathcal{B}$ takes arbitrary bases $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ of the codes $\mathcal{C}$ and $\mathcal{D}$ respectively. For each of the matrices $\mathbf{C}^{(s)}$, $\mathcal{B}$ constructs the bilinear forms $c_s(\mathbf{x}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n} \mathbf{C}_{ij}^{(s)} x_i y_j$ and for the matrices $\mathbf{D}^{(s)}$ the bilinear forms $d_s(\mathbf{x}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n} \mathbf{D}_{ij}^{(s)} x_i y_j, \forall s, 1 \leqslant s \leqslant k$. Taking $\mathcal{F} = (c_1, c_2, \ldots, c_k)$ and $\mathcal{P} = (d_1, d_2, \ldots, d_k)$ we obtain an instance $\mathcal{I}_{\mathsf{BMLE}}(\mathcal{F}, \mathcal{P})$ of $\mathsf{BMLE}(n, m, k, \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k \times \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k)$.

$\mathcal{B}$ queries $\mathcal{A}$ with the instance $\mathcal{I}_{\mathsf{BMLE}}(\mathcal{F}, \mathcal{P})$ and $\mathcal{A}$ outputs a solution $(\mathbf{S}_1, \mathbf{S}_2, \mathbf{T})$ to the BMLE instance, or no solution if there is no solution. In the first case, this immediately gives a solution $(\mathbf{S}_1, \mathbf{S}_2^\top)$ to the MCE instance. In the second case, there is no solution to the MCE instance. □

To prove the connection of MCE to the more general problem hQMLE, we first need to establish some properties of matrix codes.

Lemma 4. Let $\mathcal{C}$ and $\mathcal{D}$ be matrix codes generated by the bases $(\mathbf{C}_1, \ldots, \mathbf{C}_k)$ and $(\mathbf{D}_1, \ldots, \mathbf{D}_k)$ of (skew-)symmetric matrices, and assume that $\mathcal{C}$ and $\mathcal{D}$ have trivial automorphism groups. Then $\mathcal{C}$ is equivalent to $\mathcal{D}$ if and only if $\mathcal{C}$ is congruent to $\mathcal{D}$.

Proof. Clearly, by definition if $\mathcal{C}$ is congruent to $\mathcal{D}$, then $\mathcal{C}$ is equivalent to $\mathcal{D}$. For the opposite direction, let $\mathcal{C}$ be equivalent to $\mathcal{D}$. Then there exist nonsingular matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{T}$ such that

$$\sum_{i=1}^{k} t_{j,i} \mathbf{D}_i = \mathbf{A} \mathbf{C}_j \mathbf{B} \, .$$

Since $\mathbf{C}_i$ and $\mathbf{D}_i$ are (skew-)symmetric, we can rewrite the right-hand side as

$$\sum_{i=1}^{k} t_{j,i}\mathbf{D}_i = \mathbf{B}^\top \mathbf{C}_j \mathbf{A}^\top \ .$$

Combining the two, and since $\mathbf{A}$ and $\mathbf{B}$ are non-singular, we obtain

$$\mathbf{C}_j = \mathbf{A}^{-1}\mathbf{B}^\top \mathbf{C}_j \mathbf{A}^\top \mathbf{B}^{-1} \ .$$

The automorphism group being trivial implies $\mathbf{A} = \lambda \mathbf{B}^\top$ for some $\lambda \in \mathbb{F}_q$ which in turn implies that $\mathcal{C}$ is congruent to $\mathcal{D}$. □

Remark 1. The result of Lemma 4 has already been known for algebraically closed fields of non-even characteristic [36, 350]. Since finite fields are not algebraically closed, this result is not useful in our context. On the other hand, requiring a trivial automorphism group for the codes is not a huge restriction, and we typically expect the automorphism group to be trivial for randomly chosen matrix codes. Specifically for cryptographic purposes with regards to MCE, one wants the orbit of $\mathcal{C}$ to be maximal under the action of suitable isometries, which happens when the automorphism group of $\mathcal{C}$ is trivial. Similar requirements for trivial or small automorphism groups occur in the Hamming metric, where weak keys may exist without this requirement [171, 172].

Theorem 5. Let $\mathcal{T}$ denote the subset of $\mathbb{F}_q^{m\times n\times k}$ of k-dimensional matrix codes of symmetric matrices with trivial automorphism groups. Further, let $\mathcal{T}'$ denote the subset of $\mathbb{F}_q[x_1,\ldots,x_N]^k$ of k-tuples of polynomials with trivial automorphism groups.

The $\mathsf{MCE}(\mathcal{T})$ problem is at least as hard as the $\mathsf{hQMLE}(\mathcal{T}')$ problem

Proof. We perform the reduction in a similar manner as previously.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F},\mathcal{P})$ of $\mathsf{hQMLE}(N,k,\mathcal{T}')$, where $\mathcal{F} = (f_1, f_2, \ldots, f_k)$, $\mathcal{P} = (p_1, p_2, \ldots, p_k) \in [x_1,\ldots,x_N]$ are k-tuples of linearly independent quadratic forms from $\mathcal{T}'$. $\mathcal{B}$ can efficiently construct an instance of the $\mathsf{MCE}(N,N,k,\mathcal{T})$ problem as follows.

$\mathcal{B}$ forms the $N \times N$ symmetric matrices $\mathbf{F}^{(s)}$ and $\mathbf{P}^{(s)}$ associated to the components f_s and p_s, $s \in \{1,\ldots,k\}$ of the mappings $\mathcal{F}$ and $\mathcal{P}$. Let $\mathcal{D}$ denote the matrix code with basis $(\mathbf{P}^{(1)},\ldots,\mathbf{P}^{(k)})$ and similarly $\mathcal{C}$ the matrix code with basis $(\mathbf{F}^{(1)},\ldots,\mathbf{F}^{(k)})$, so that $\mathcal{B}$ obtains an instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C},\mathcal{D})$ of MCE. Per assumption, the matrix codes $\mathcal{C}$ and $\mathcal{D}$ have trivial automorphism groups, hence the instance is from $\mathsf{MCE}(N,N,k,\mathcal{T})$.

$\mathcal{B}$ queries $\mathcal{A}$ with the instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ and $\mathcal{A}$ answers with a solution $(\mathbf{A}, \mathbf{B})$ to the MCE instance if it is positive, and no solution otherwise. In the former case, from Lemma 4, since the matrices are symmetric, $\mathbf{A} = \mathbf{B}^\top$. Now, $\mathcal{B}$ applies the change of variables $\mathbf{xA}$ to $\mathcal{F}$ and obtains $\mathcal{R}(\mathbf{x}) = \mathcal{F}(\mathbf{xA})$. It then solves the system

$$\sum_{j=1}^{k} t_{j,s} \cdot r_j = p_s, \quad \text{for all } s \in \{1, \ldots, k\} \tag{36}$$

The system has a solution if $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ is a positive instance. This is always the case in odd characteristic, because there is a one-to-one correspondence between polynomials and their symmetric matrix representation. Over characteristic 2, it may happen that the $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ is not a positive instance while its symmetric matrix representation $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ is. In this case, the system (36) does not have a solution and $\mathcal{B}$ outputs no solution.

If the system has a solution, $\mathcal{B}$ sets $\mathbf{T} = (t_{i,j})$, and outputs $(\mathbf{A}, \mathbf{T})$ as the solution to $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$. Thus, $\mathcal{B}$ succeeds whenever $\mathcal{A}$ succeeds and the reduction takes time $\mathcal{O}(k^6)$. □

For the following theorem, we define the symmetric matrix representation of a matrix code $\mathcal{C}$ as the code $\tilde{\mathcal{C}} = \{ \begin{bmatrix} \mathbf{0} & \mathbf{C}^\top \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \mid \mathbf{C} \in \mathcal{C} \}$.

Theorem 6. Let $\mathcal{T}_s$ denote the subset of $\mathbb{F}_q^{m \times n \times k}$ of k-dimensional matrix codes whose symmetric matrix representation has a trivial automorphism group. Similarly, let $\mathcal{T}_s'$ denote the subset of $\mathbb{F}_q[x_1, \ldots, x_N]^k$ of k-tuples of polynomials with trivial automorphism groups.

The $\mathsf{hQMLE}(\mathcal{T}_s')$ problem is at least as hard as the $\mathsf{MCE}(\mathcal{T}_s)$ problem.

Proof. We show that given any oracle $\mathcal{A}$ that solves $\mathsf{hQMLE}(\mathcal{T}_s')$ there exists an oracle $\mathcal{B}$ running in polynomial time that solves $\mathsf{MCE}(\mathcal{T}_s)$.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ of $\mathsf{MCE}(n, m, k, \mathcal{T}_s)$. $\mathcal{B}$ can efficiently construct an instance of the $\mathsf{hQMLE}(n + m, k, \mathcal{T}_s')$ problem as follows. $\mathcal{B}$ fixes bases $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ of the code $\mathcal{C}$ and $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ of the code $\mathcal{D}$. For each of the matrices $\mathbf{C}^{(s)}$, using $\mathbf{x} = (x_1, \ldots, x_{m+n})$ to denote the variables, $\mathcal{B}$ constructs the quadratic forms

$$c_s(\mathbf{x}) = \sum_{1 \leqslant i \leqslant m, m+1 \leqslant j \leqslant m+n} \mathbf{C}_{ij}^{(s)} x_i x_j$$

and for the matrices $\mathbf{D}^{(s)}$ the quadratic forms

$$d_s(\mathbf{x}) = \sum_{1 \leqslant i \leqslant m, m+1 \leqslant j \leqslant m+n} \mathbf{D}^{(s)}_{ij} x_i x_j,$$

for all $1 \leqslant s \leqslant k$. Taking $\mathcal{F} = (c_1, c_2, \ldots, c_k)$ and $\mathcal{P} = (d_1, d_2, \ldots, d_k)$ as quadratic maps, $\mathcal{B}$ obtains an instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ of $\mathsf{hQMLE}(n+m, k, \mathcal{T}'_s)$, with which queries $\mathcal{A}$, who outputs a solution $(\mathbf{S}, \mathbf{T})$ to the hQMLE instance.

We argue that, if the instance is positive, this solution can be transformed to a solution to the MCE instance. A basis of the codes $\tilde{C}$ and $\tilde{D}$ can be given by

$$\begin{bmatrix} \mathbf{0} & (\mathbf{C}^{(i)})^\top \\ \mathbf{C}^{(i)} & \mathbf{0} \end{bmatrix} \text{ and } \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(i)})^\top \\ \mathbf{D}^{(i)} & \mathbf{0} \end{bmatrix} \text{ for } i \in \{1, \ldots, k\}. \tag{37}$$

The solution $(\mathbf{S}, \mathbf{T})$ means

$$\sum \tilde{t}_{i,j} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(j)})^\top \\ \mathbf{D}^{(j)} & \mathbf{0} \end{bmatrix} = \mathbf{S} \begin{bmatrix} \mathbf{0} & (\mathbf{C}^{(i)})^\top \\ \mathbf{C}^{(i)} & \mathbf{0} \end{bmatrix} \mathbf{S}^\top, \text{ for } i \in \{1, \ldots, k\}. \tag{38}$$

If the given MCE instance is positive, then there exist matrices $\mathbf{A}, \mathbf{B}, \mathbf{L}$ such that $\mathbf{A}\mathbf{C}_i\mathbf{B} = \sum_j l_{i,j}\mathbf{D}_j$. This implies

$$\sum l_{i,j} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(j)})^\top \\ \mathbf{D}^{(j)} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{0} & (\mathbf{C}^{(i)})^\top \\ \mathbf{C}^{(i)} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^\top \end{bmatrix}, \tag{39}$$

for all $i \in \{1, \ldots, k\}$. These last two equations imply for every $i \in \{1, \ldots, k\}$ that

$$\sum \lambda_{i,j} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(j)})^\top \\ \mathbf{D}^{(j)} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \mathbf{S}^{-1} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(i)})^\top \\ \mathbf{D}^{(i)} & \mathbf{0} \end{bmatrix} \mathbf{S}^{-\top} \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^\top \end{bmatrix}. \tag{40}$$

By assumption, the automorphism group of $\tilde{\mathcal{D}}$ is trivial, which means $\mathbf{S}$ necessarily equals $\begin{bmatrix} \mathbf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix}$ up to scalar multiplication. For such an $\mathbf{S}$, the MCE solution can immediately be extracted, which $\mathcal{B}$ then outputs. In the other hand, if $\mathbf{S}$ is not of such block-diagonal form, $\mathcal{B}$ outputs no solution, as this implies the instance is not positive. □

Remark 2. Using the above reduction between MCE and hQMLE, we can reduce the MCEbase problem to and from a special case of IP known as IP1S. Inter-

estingly, Perret [309] shows IP1S is polynomially solvable for most instances $k \geq N$, and later work [67] gives an algorithm with a running time of $\mathcal{O}(N^6)$ for most random instances, although no rigorous proof that bounds the complexity of the problem to polynomial was given. This nevertheless implies that the MCEbase problem can practically be solved in polynomial time for most cryptographically interesting parameters.

3.2 RELATIONS TO EQUIVALENCE PROBLEMS FOR LINEAR CODES

In this section, we show that MCE is at the heart of various code equivalence problems. We show that equivalence problems for different metrics, such as the Hamming metric or the sum-rank metric, reduce to MCE, making the hardness analysis of MCE the more exciting.

Hamming code equivalence.

Codes $\mathcal{C} \subseteq \mathbb{F}_q^n$ equipped with the *Hamming metric* have isometries of the form

$$\tau : (c_1, \ldots, c_n) \mapsto (\alpha_1 c_{\pi^{-1}(1)}, \ldots, \alpha_n c_{\pi^{-1}(n)}), \quad \alpha_i \in \mathbb{F}_q^*, \ \pi \in S_n. \tag{41}$$

From this, we define *Hamming code equivalence* (HCE) as the problem of finding an isometry between two Hamming codes $\mathcal{C}$ and $\mathcal{D}$.

Problem 8. HCE(k, n):
Input: Two k-dimensional Hamming codes $\mathcal{C}, \mathcal{D} \subseteq \mathbb{F}_q^n$
Question: Find – if any – $\alpha \in \mathbb{F}_q^{*n}, \pi \in S_n$ such that $\alpha\pi(\mathbf{c}) \in \mathcal{D}$ holds for all $\mathbf{c} \in \mathcal{C}$.

The subproblem where α is trivial is called the *monomial equivalence problem*.

It is easy to turn an HCE instance into a MCE instance [138], given the description in Equation (41). First, define $\Phi : \mathbb{F}_q^n \to \mathbb{F}_q^{n \times n}$ by

$$\mathbf{x} = (x_1, \ldots, x_n) \mapsto \begin{pmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{pmatrix}.$$

The map Φ is an isometry from the Hamming metric to the rank metric: codewords with weigh t are mapped to matrices of rank t. From this, we quickly

get the reduction: Writing π as a matrix $\mathbf{P} \in \mathrm{GL}_n(q)$, the map Φ translates a Hamming isometry $\tau = (\alpha, \pi)$ to a rank-metric isometry $\Phi(\tau)$ by

$$\Phi(\tau) : \Phi(\mathbf{x}) \mapsto \mathbf{P}^{-1}\Phi(\mathbf{x})\mathbf{A}\mathbf{P}, \quad \text{where } \mathbf{A} = \begin{pmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_n \end{pmatrix} \in \mathrm{GL}_n(q).$$

A second reduction from HCE to MCE is given later in [138], which concerns the search variant of the problem, and is more explicit. Both reductions, however, do not help with solving HCE in practice: both the permutational ($\mathbf{A}$ is trivial) and the linear variant of code equivalence in the Hamming metric have algorithms [30, 311] that perform much better for an HCE instance τ than the algorithms we propose for solving $\Phi(\tau)$ as an MCE instance.

Sum-rank code equivalence.

The *sum-rank metric* [297] is a metric that is gaining in popularity in coding theory. It is commonly given as a generalization of the vector-rank metric, but one can also define a variant that generalizes matrix-rank metric. We will reduce both vector and matrix sum-rank equivalence problems to MCE. The idea is the same as for HCE: we find the right isometry from sum-rank metric to rank metric to get the reduction.

Definition 7. Let n be partitioned as $n = n_1 + \ldots + n_\ell$. Let $\mathbf{v}^{(i)} = (v_1^{(i)}, \ldots, v_{n_i}^{(i)}) \in \mathbb{F}_{q^m}^{n_i}$ and $\mathbf{v} = (\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(\ell)}) \in \mathbb{F}_{q^m}^{n}$. Let Γ be a basis for $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Then the *vector sum-rank* of $\mathbf{v}$ is defined as

$$\mathrm{SumRank}(\mathbf{v}) := \sum_{i=1}^{\ell} \mathrm{Rank}\, \Gamma(\mathbf{v}^{(i)}).$$

Let furthermore m be partitioned as $m = m_1 + \ldots + m_\ell$. Let $\mathbf{V}^{(i)} \in \mathbb{F}_q^{m_i \times n_i}$ and $\mathbf{V} = (\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(\ell)})$. Then the *matrix sum-rank* of $\mathbf{V}$ is defined as

$$\mathrm{SumRank}(\mathbf{V}) = \sum_{i=1}^{\ell} \mathrm{Rank}\, \mathbf{V}^{(i)}.$$

The sum-rank generalizes both the Hamming metric and the rank metric: taking $\ell = n$ gives the Hamming metric, whereas $\ell = 1$ gives the rank metric. We define isometries again as maps that preserve the sum-rank.

Vector sum-rank isometries are simple generalisations of vector-rank isometries (see Problem 4).

Proposition 8 (Thm. 3.7, [9])**.** Isometries with respect to the vector sum-rank metric are given by vector rank isometries

$$\mu^{(i)} : \mathbf{x}^{(i)} \mapsto \alpha^{(i)} \mathbf{x}^{(i)} \mathbf{B}^{(i)}$$

per 'block' with $\alpha^{(i)} \in \mathbb{F}_{q^m}^*$ and $\mathbf{B}^{(i)} \in \mathrm{GL}_{n_i}(q)$, and suitable permutations π of such blocks if $n_i = n_j$ for $i \neq j$. Thus,

$$\mu : (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell)}) \mapsto (\alpha^{(1)} \mathbf{x}^{\pi^{-1}(1)} \mathbf{B}^{(1)}, \ldots, \alpha^{(\ell)} \mathbf{x}^{\pi^{-1}(\ell)} \mathbf{B}^{(\ell)})$$

is a general description of a vector sum-rank isometry.

Generalizing to matrix sum-rank codes is achieved by simply replacing $\alpha^{(i)} \in \mathbb{F}_{q^m}^*$ with $\mathbf{A}^{(i)} \in \mathrm{GL}_{m_i}(q)$ [293, Prop. 4.25]. This gives us the *Vector Sum-Rank Code Equivalence* (VSRCE) and *Matrix Sum-Rank Code Equivalence* (MSRCE) problems.

Problem 9. VSRCE(n, m, k):
Input: Two k-dimensional vector sum-rank codes $\mathcal{C}, \mathcal{D} \subseteq \mathbb{F}_{q^m}^n$
Question: Find – if any – $\alpha^{(i)} \in \mathbb{F}_{q^m}^*, \mathbf{B}^{(i)} \in \mathrm{GL}_{n_i}(q)$ and a permuation π such that for all $\mathbf{c} \in \mathcal{C}$, it holds that $\mu(\mathbf{c}) \in \mathcal{D}$.

Problem 10. MSRCE(n, m, k):
Input: Two k-dimensional matrix sum-rank codes $\mathcal{C}, \mathcal{D} \subseteq \left(\mathbb{F}_q^{m_i \times n_i}\right)_i$
Question: Find – if any – $\mathbf{A}^{(i)} \in \mathrm{GL}_{m_i}(q), \mathbf{B}^{(i)} \in \mathrm{GL}_{n_i}(q)$ and a permuation π such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mu(\mathbf{C}) \in \mathcal{D}$.

In order to give a reduction from VSRCE to MCE, we use the same idea as for HCE. First, we define a 'nice' map $\Psi : \mathbb{F}_q^n \to \mathbb{F}_q^{\ell \cdot m \times n}$ by

$$\mathbf{x} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell)}) \mapsto \begin{pmatrix} \mathrm{Mat}(\mathbf{x}^{(1)}) & & \\ & \ddots & \\ & & \mathrm{Mat}(\mathbf{x}^{(\ell)}) \end{pmatrix}.$$

It is clear that Ψ is an isometry from the vector sum-rank metric to the rank metric, as it preserves the weight. We get the following reduction.

Theorem 9. Let $\mathcal{T}$ denote the subset of $\mathbb{F}_q^{m\times n\times k}$ of k-dimensional matrix codes with trivial automorphism groups. Let $\mathcal{T}'$ denote the subset of k-dimensional vector sum-rank codes that are in the preimage $\Psi^{-1}(\mathcal{T})$. Then $\mathsf{MCE}(\mathcal{T})$ is at least as hard as $\mathsf{VSRCE}(\mathcal{T}')$.

Proof. Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{VSRCE}}(\mathcal{C},\mathcal{D})$ of $\mathsf{VSRCE}(n,m,k,\mathcal{T}')$, where $\mathcal{C}$ and $\mathcal{D}$ are k-dimensional vector sum-rank codes. $\mathcal{B}$ can efficiently construct an instance of the $\mathsf{MCE}(\mathcal{T})$ problem as follows. By writing the permutation of the 'blocks' π by a matrix representation $\mathbf{P}$, $\mathcal{B}$ can translate a vector sum-rank isometry μ into a matrix code isometry $\Psi(\mu)$ by

$$\Psi(\mu):\Psi(\mathbf{x})\mapsto \mathbf{P}^{-1}\mathbf{A}\Psi(\mathbf{x})\mathbf{B}\mathbf{P}$$

where

$$\mathbf{A}=\begin{pmatrix}\alpha^{(1)} & & \\ & \ddots & \\ & & \alpha^{(\ell)}\end{pmatrix},\quad \mathbf{B}=\begin{pmatrix}\mathbf{B}^{(1)} & & \\ & \ddots & \\ & & \mathbf{B}^{(\ell)}\end{pmatrix}$$

with $\mathbf{A}\in \mathrm{GL}_\ell(q^m)$, $\mathbf{B}\in \mathrm{GL}_n(q)$. Hence, $\Psi(\mu)$ can be seen as an instance of $\mathsf{MCE}(n,m,k,\mathcal{T})$, with which $\mathcal{B}$ queries $\mathcal{A}$, who outputs a solution $(\mathbf{A}',\mathbf{B}')$. As the automorphism group is trivial, $\mathcal{B}$ computes $\lambda\mathbf{A}'=\mathbf{P}^{-1}\mathbf{A}$ and $\lambda\mathbf{B}'=\mathbf{B}\mathbf{P}$ for $\lambda\in\mathbb{F}_q$, and therefore solves the $\mathcal{I}_{\mathsf{VSRCE}}$ instance. □

Expanding this to a reduction from MSRCE to MCE is only a small step. Given a partition $m=m_1+\ldots+m_\ell$, the map we need is only slightly different from Ψ, namely $\tilde{\Psi}:\left(\mathbb{F}_q^{m_i\times n_i}\right)_i\to\mathbb{F}_q^{m\times n}$ by

$$\mathbf{X}=(\mathbf{X}^{(1)},\ldots,\mathbf{X}^{(\ell)})\mapsto\begin{pmatrix}\mathbf{X}^{(1)} & & \\ & \ddots & \\ & & \mathbf{X}^{(\ell)}\end{pmatrix}.$$

Theorem 10. Let $\mathcal{T}$ denote the subset of $\mathbb{F}_q^{m\times n\times k}$ of k-dimensional matrix codes with trivial automorphism groups. Let $\mathcal{T}'$ denote the subset of k-dimensional matrix sum-rank codes that are in the preimage $\tilde{\Psi}^{-1}(\mathcal{T})$. Then $\mathsf{MCE}(\mathcal{T})$ is at least as hard as $\mathsf{MSRCE}(\mathcal{T}')$.

Proof. This is a simple generalization of Theorem 9: Replace $\alpha^{(i)}$ by $\mathbf{A}^{(i)}\in \mathrm{GL}_{m_i}(q)$ so that $\mathbf{A}\in\mathrm{GL}_m(q)$. Then again, for a matrix sum-rank μ we get $\tilde{\Psi}(\mu)$ by $\Psi(\mathbf{x})\mapsto\mathbf{P}^{-1}\mathbf{A}\Psi(\mathbf{x})\mathbf{B}\mathbf{P}$ as an $\mathsf{MCE}(\mathcal{T})$ instance. □

The link between such MCE instances $\Psi(\mu)$ coming from vector sum-rank and $\tilde{\Psi}(\mu)$ coming from matrix sum-rank is given by a representation $\rho : \mathbb{F}_{q^m}^* \to \mathrm{GL}_m(q)$. We map a vector sum-rank instance to a matrix sum-rank instance by $\mathbf{A}^{(i)} = \rho(\alpha^{(i)})$, so that $\mathbf{A} \in \mathrm{GL}_{\ell \cdot m}(q)$.

To show the equivalences between the rank and sum-rank instances, we need to show that an MCE instance is also an MSRCE instance. But this is trivial: the sum-rank metric generalizes the rank metric, thus an MCE instance is an MSRCE instance with $\ell = 1$. Hence, we get the following theorem for free.

Theorem 11. MSRCE is at least as hard as MCE.

The results in this section are summarized in Figure 25.

§4 Solving Matrix Code Equivalence

In this section, we analyze the complexity of solving an instance of $\mathsf{MCE}(n, m, k)$. We start by establishing a useful lemma.

Lemma 12. An $\mathsf{MCE}(n, m, k)$ instance can in polynomial time be turned into an $\mathsf{MCE}(\sigma(n), \sigma(m), \sigma(k))$ instance for any permutation σ on the set $\{n, m, k\}$. Furthermore, they are either both positive, or both negative instances.

Proof. Let $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ be a given $\mathsf{MCE}(n, m, k)$ instance. Let $(\mathbf{C}^{(1)}, \dots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \dots, \mathbf{D}^{(k)})$ be bases of the codes $\mathcal{C}$ and $\mathcal{D}$ respectively. Without loss of generality, we will turn this instance into an $\mathsf{MCE}(m, k, n)$ instance, other permutations can be done analogously. We set $\bar{\mathbf{C}}_{j,t}^{(i)} = \mathbf{C}_{i,j}^{(t)}$ and $\bar{\mathbf{D}}_{j,t}^{(i)} = \mathbf{D}_{i,j}^{(t)}$, and we take $(\bar{\mathbf{C}}^{(1)}, \dots, \bar{\mathbf{C}}^{(n)})$ and $(\bar{\mathbf{D}}^{(1)}, \dots, \bar{\mathbf{D}}^{(n)})$ to be the bases of the codes $\bar{\mathcal{C}}$ and $\bar{\mathcal{D}}$ respectively. Clearly, $\bar{\mathcal{C}}$ and $\bar{\mathcal{D}}$ are equivalent if and only if $\mathcal{C}$ and $\mathcal{D}$ are equivalent. □

Without loss of generality, and with Lemma 12 in mind, we may assume $m = \min\{m, n, k\}$.

As a baseline we have a straightforward algorithm that uses a result from [138] that MCRE can be solved in polynomial time. By enumerating either $\mathbf{A}$ or $\mathbf{B}$, we obtain an instance of MCRE. This means the dominating complexity is the enumeration resulting in an overall complexity of $\tilde{\mathcal{O}}(q^{m^2})$ for MCE.

The approach we outline in this section makes use of the reduction of MCE to hQMLE (see Theorem 6). This means that we use techniques already applied for solving hQMLE, but generalize and improve them by making use of the specific structure that MCE instances show when viewed as hQMLE instances.

4.1 SOLVING MCE AS QMLE

Bouillaguet, Fouque, and Véber [69] propose an algorithm for solving hQMLE using techniques from graph theory. Their main idea is to reduce the homogeneous case to the inhomogeneous case, which they assume is efficiently solvable, for example using the heuristic algebraic approach of [173]. Starting from an instance of hQMLE, they build two exponentially-large graphs that correspond to the given maps $\mathcal{F}$ and $\mathcal{P}$ such that, finding an isomorphism between the two graphs is equivalent to finding an isomorphism between the two quadratic maps. Since the graphs are exponentially large, a technique is provided to *walk* through the graphs without constructing them. Walking through the graphs consists of finding adjacent nodes and computing the degree of a node, both in polynomial time. The algorithm consists in finding pairs of nodes from the first and the second graph that have the same degree and making queries to an inhomogenous QMLE solver. If the solver finds an isomorphism by which two nodes are related, then the isomorphism between the two graphs, and thus the isomorphism between the two quadratic maps, can be found.

4.2 FIRST ALGORITHM FOR SOLVING MCE

The algorithm for solving hQMLE from [69] considers a graph arising from the differential of a given polynomial map – a node **a** is connected to all the nodes that vanish at the differential at **a**. However, it is not entirely clear how the property we choose to construct such graphs impacts the complexity of the algorithm. We revisit the algorithm, and show how it can be generalized, i.e. abstracted from the property used in [69], under certain conditions. In this section we present this generalization: a birthday-based algorithm for finding an isomorphism between two objects when a specific solver exists. In this form, it can be applied to a broader type of equivalence problems, using more general invariants, here implemented as a predicate $\mathbb{P}$.

Let S_1 and S_2 be subsets of a finite universe U of equal size N. Assume there exists a predicate $\mathbb{P} : U \to \{\top, \bot\}$ that can be computed in polynomial time, and denote the (maximal) cost of computing $\mathbb{P}$ for $u \in U$ by $C_{\mathbb{P}}$. Assume $\mathbb{P}$ is invariant under the equivalence φ, i.e. $\mathbb{P}(x) = \top \leftrightarrow \mathbb{P}(\varphi(x)) = \top$. Let $U_\top = \{x \in U \mid \mathbb{P}(x) = \top\}$, and $d = |U_\top|/|U|$. We call d the *density* of the predicate $\mathbb{P}$ and we assume the density on S_1 and S_2 is approximately equal to d. Furthermore, assume the existence of an algorithm FINDFUNCTION, that given $x \in S_1, y \in S_2$ returns φ if $y = \varphi(x)$ and $\bot$ otherwise. We denote the cost

of a query to FindFunction by C_{FF}. Then, Algorithm 28 finds an equivalence function $\varphi : S_1 \to S_2$, if it exists.

Algorithm 28 General Birthday-based Equivalence Finder

```
1: procedure SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if ℙ(a) then
6:             L ← L ∪ {a}
7:     until |L| = ℓ
8:     return L
9: procedure CollisionFind(S1, S2)
10:    L1 ← SampleSet(S1, ℙ, ℓ)
11:    L2 ← SampleSet(S2, ℙ, ℓ)
12:    for all (a, b) ∈ L1 × L2 do
13:        φ ← FindFunction(a, b)
14:        if φ ≠ ⊥ then
15:            return solution φ
16:    return ⊥
```

Lemma 13. Let d be the density of $\mathbb{P}$. To achieve a fixed success probability of $1 - 1/e$, Algorithm 28 performs on average $\mathcal{O}(\sqrt{N/d})$ operations in SampleSet and queries FindFunction at most $d \cdot N$ times.

The optimal density, up to a polynomial factor, is $d = N^{-1/3} \cdot C_{FF}^{-2/3}$, for which the total time complexity of the algorithm is $\mathcal{O}(N^{2/3} \cdot C_{FF}^{1/3})$ and the memory complexity is $\mathcal{O}(N^{1/3} \cdot C_{FF}^{-1/3})$. If FindFunction runs in polynomial time, this reduces to time complexity of $\tilde{\mathcal{O}}(N^{2/3})$ and memory complexity of $\mathcal{O}(N^{1/3})$.

Proof. The expected number of elements in S_1 and S_2 such that $\mathbb{P}(x)$ holds is equal to dN by the definition of the density d. By the birthday paradox, it is enough to take lists of size $\ell = \sqrt{d \cdot N}$, to be sure that with probability of $1 - \frac{1}{e}$ FindFunction returns a solution [372]. With this length of the lists, the number of queries to FindFunction is $\ell^2 = dN$. On the other hand, the number of samples needed to build the list L_1 of elements $a \in S_1$ such that $\mathbb{P}(a)$ holds is $\ell/d = \sqrt{N/d}$, and similarly for L_2. This gives a complexity of $\mathcal{O}(\sqrt{N/d})$ to build these lists L_i.

The total running time is optimal when these two quantities $\sqrt{N/d}$ and $d \cdot N \cdot C_{FF}$ are equal, which holds when $d = N^{-1/3} \cdot C_{FF}^{-2/3}$. Such a density gives complexity of $\mathcal{O}(N^{\frac{2}{3}} \cdot C_{FF}^{\frac{1}{3}})$ for SampleSet and at most $N^{\frac{2}{3}}$ queries to FindFunction. If C_{FF} is polynomial, this gives a total time complexity of $\tilde{\mathcal{O}}(N^{\frac{2}{3}})$. The memory requirements of the algorithm correspond to the size of the lists L_i. This results in a memory complexity of $\mathcal{O}(N^{\frac{1}{3}} C_{FF}^{-\frac{1}{3}})$, or $\mathcal{O}(N^{\frac{1}{3}})$ if C_{FF} is polynomial. □

Remark 3. The success probability in Lemma 13 is chosen somewhat arbitrarily, mostly for practical verification of the algorithm's correctness. It can be increased to any value $1 - 1/c$ for a positive constant c by appropriately building lists that are larger only by a constant factor compared to the case treated in Lemma 13. The overall complexity only differs by a constant factor, i.e., does not change asymptotically.

As mentioned earlier, the algorithm presented in [69] is a special case of Algorithm 28. Their algorithm can be seen as an instantiation of Algorithm 28 by defining $G_{\mathcal{F}}$ (resp. $G_{\mathcal{P}}$) to be the linearity graph of $\mathcal{F}$ (resp. $\mathcal{P}$), where a node $\mathbf{a}$ is connected to all nodes $\mathbf{x}$ such that $D_{\mathbf{a}}\mathcal{F}(\mathbf{x}) = 0$ (resp. $D_{\mathbf{a}}\mathcal{P}(\mathbf{x}) = 0$), taking the predicate

$$\mathbb{P}_\kappa(\mathbf{a}) : \dim \ker D_{\mathbf{a}}\mathcal{F} = \kappa$$

on the universe $\mathbb{F}_q^{k\times N}$, and taking for FindFunction the assumed polynomial-time solver from [173] for inhQMLE. By finding a collision (α, β) such that $\beta = \alpha\mathbf{S}$ we can create an inhomogeneous instance $\mathcal{P}'$, $\mathcal{F}'$ using that $\mathcal{P}(\mathbf{x}+\alpha) = \mathcal{F}(\mathbf{x}\mathbf{S}+\beta)\mathbf{T}$, by defining $\mathcal{P}'(\mathbf{x}) = \mathcal{P}(\mathbf{x}+\alpha)$ and $\mathcal{F}'(\mathbf{x}) = \mathcal{F}(\mathbf{x}+\beta)$. Running FindFunction on $\mathcal{P}'$ and $\mathcal{F}'$ then returns $\mathbf{S}$ and $\mathbf{T}$. In this case, Lemma 13 gives the precise result from [69, Thm. 1], which we present as a corollary to our Lemma 13, for completeness.

Corollary 14. Assuming a polynomial-time solver for inhQMLE, an $\mathsf{hQMLE}(N, N)$ instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$ can be solved with complexity and number of queries equal to $\tilde{\mathcal{O}}(q^{\frac{2}{3}N})$ with a success probability of $1 - 1/c$ for any $c > 0$ and a memory complexity of $\mathcal{O}(q^{\frac{1}{3}N})$.

Proof. Let $G_{\mathcal{F}}$ be the linearity graph of $\mathcal{F}$, where a node $\mathbf{a}$ is connected to all $\mathbf{x}$ such that $D_{\mathbf{a}}\mathcal{F}(\mathbf{x}) = 0$, and similarly define $G_{\mathcal{P}}$. We use the predicate $\mathbb{P}_\kappa(\mathbf{a}) : \dim \ker D_{\mathbf{a}}\mathcal{F} = \kappa$, that is, the predicate holds if $\deg(\mathbf{a}) = q^\kappa$. The density of the predicate d_κ in the universe of $N \times N$ matrices is independent of $\mathcal{F}$ and $\mathcal{P}$, and is therefore the same as the density of linear maps with kernel of dimension κ. Thus, d_κ is approximately a monotonic decreasing function in κ, going from 1 to 0. Hence, by Lemma 13, there exists some optimal κ for which we get that $d_\kappa \approx |G_{\mathcal{P}}|^{-1/3} = q^{-N/3}$, which gives a total time complexity of $q^{\frac{2}{3}N}$ and a memory complexity of $q^{\frac{1}{3}N}$. □

Remark 4. The assumption on a polynomial-time solver in [69] turns out to be too strong: such a solver exists for random instances, however, for inhQMLE instances as obtained in Corollary 14 the running time is probably not polynomial [68]. Nevertheless, the algorithm and result are valid, but require

a different rebalancing depending on C_{FF}. Section 5 analyzes C_{FF} in detail for different instances.

To apply this approach to MCE instances, we need to generalize to the case of N not necessarily equal to k. For an $\mathsf{MCE}(n,m,k)$ instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C},\mathcal{D})$, we get an $\mathsf{hQMLE}(n+m,k)$ instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F},\mathcal{P})$ by Theorem 6. We take again the predicate $\mathbb{P}_\kappa(\mathbf{a}) : \dim\ker D_{\mathbf{a}}\mathcal{F} = \kappa$, but this time on the universe $\mathbb{F}_q^{k\times(n+m)}$, where $D_{\mathbf{a}}\mathcal{F}$ lives. To get a similar result to Corollary 14, we need to show two things. **a)**, that this predicate satisfies the assumptions required for Algorithm 28. **b)**, that there is a κ such that the density d_κ of $\mathbb{P}_\kappa$ is optimal as described in Lemma 13. If both are satisfied, we get a complexity of $\mathcal{O}(q^{\frac{2}{3}(n+m)}C_{\mathrm{FF}}^{\frac{1}{3}})$, hence $\tilde{\mathcal{O}}(q^{\frac{2}{3}(n+m)})$ when the solver is polynomial, with a success probability of $1-1/c$ for any $c>0$ for an $\mathsf{MCE}(n,m,k)$ instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C},\mathcal{D})$. We start with **a)**.

Lemma 15. The predicate $\mathbb{P}_\kappa(D_{\mathbf{a}}\mathcal{F}) : \dim\ker D_{\mathbf{a}}\mathcal{F} = \kappa$ is a suitable predicate for Algorithm 28, as **i)** $\mathbb{P}_\kappa$ can be computed in polynomial time, **ii)** is invariant under equivalence, **iii)** and d_κ does not depend on $\mathcal{F}$.

Proof.

1. The cost $C_{\mathbb{P}_\kappa}$ is the cost of computing $\dim\ker D_{\mathbf{a}}\mathcal{F}$, i.e. computing the kernel of a $k\times(n+m)$ matrix over $\mathbb{F}_q$. This can be done in polynomial time.

2. Let $\mathcal{P}(\mathbf{x}) = \mathcal{F}(\mathbf{xS})\mathbf{T}$ be the equivalence. If $\mathbf{x}\in\ker D_{\mathbf{a}}\mathcal{P}$ then $\mathbf{xS}\in\ker\mathcal{F}_{\mathbf{aS}}$ and vice versa, as $\mathbf{T}$ does not affect the kernel. As $\mathbf{S}$ is invertible, we get a one-to-one correspondence $\mathbf{x}\mapsto\mathbf{xS}$ between the kernels, so $\mathbb{P}_\kappa(D_{\mathbf{aS}}\mathcal{F}) = \mathbb{P}_\kappa(D_{\mathbf{a}}\mathcal{P})$.

3. For $\mathcal{F}$ coming from an MCE instance, we always have $-\mathbf{a}\in\ker D_{\mathbf{a}}\mathcal{F}$. We want to show that the distribution of the rank of $D_{\mathbf{a}}\mathcal{F}$ follows the ranks of linear maps vanishing at $-\mathbf{a}$. This is given by [163, Thm. 2] for even characteristic and easily adapted to odd characteristic, which shows d_κ is independent of $\mathcal{F}$.

□

We now continue with **b)**: we show that there is a κ such that d_κ is optimal. For now, existence of κ is enough to derive a complexity on MCE. We will explicitly compute κ in Section 5 when we give an overview of C_{FF} for specific parameter sets (k,n,m).

The general density d_κ for the predicate $\mathbb{P}_\kappa$ is given by the following lemma, taking $a = k$ and $b = n + m$ to avoid confusion with regards to n, m and $n + m$.

Lemma 16. Let $\mathbb{P}_\kappa : \dim \ker \mathbf{M} = \kappa$ be a predicate on $a \times b$ matrices over $\mathbb{F}_q$ with $a \geqslant b$. Then the density of the predicate $\mathbb{P}_\kappa$ is $d_\kappa = 1/\Theta(q^{(\kappa^2+\kappa\cdot(a-b))})$.

Proof. The number of rank-r matrices of size $a \times b$ over a finite field $\mathbb{F}_q$ is well-known [251] and given by

$$\prod_{i=0}^{r-1} \frac{(q^a - q^i)(q^b - q^i)}{q^r - q^i} = \Theta\left(q^{(a+b-r)r}\right) .$$

We have $\kappa = b - r$ and so $d_\kappa^{-1} = \frac{|U|}{|U_\top|} = \Theta(\frac{q^{ab}}{q^{-(a+b-r)r}}) = \Theta(q^{\kappa^2+\kappa(a-b)})$. Specifically when the matrix is square, $d_\kappa^{-1} = \Theta(q^{\kappa^2})$. □

From Lemma 16 we can conclude that for some κ, the density d_κ is optimal. This means we satisfy both **a)** and **b)** and we can apply Lemma 13.

In conclusion, we get our first result on the hardness of MCE, which significantly improves straightforward enumeration. This requires that such a κ exists, which happens when $k \leqslant 2(n + m)$, by Lemma 16. For now, in contrast to [69, Thm. 1], we do not assume a polynomial-time solver for the inhomogeneous case of QMLE, but instead write the cost as C_{FF}. We explore the precise cost of FindFunction in Section 5.

Theorem 17. An $\mathsf{MCE}(n, m, k)$ instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ over $\mathbb{F}_q$ with $k \leqslant 2(n + m)$ can be solved using Algorithm 28 with time complexity equal to $\mathcal{O}(q^{\frac{2}{3}(n+m)} \cdot C_{\mathrm{FF}}^{\frac{1}{3}} \cdot (C_{\mathbb{P}_\kappa} + 1))$, memory complexity equal to $\mathcal{O}(q^{\frac{1}{3}(m+n)} C_{\mathrm{FF}}^{-\frac{1}{3}})$ and with success probability of $1 - 1/c$ for any $c > 0$, where C_{FF} denotes the cost of a single query to FindFunction.

We will show in Section 5 that, even though C_{FF} is *not* polynomial-time, the complexity of Algorithm 28 is still $\tilde{\mathcal{O}}(q^{\frac{2}{3}(n+m)})$ for some κ.

When $k > 2(n + m)$, practically all differentials $D_{\mathbf{a}}\mathcal{F}$ will have only the trivial kernel spanned by $-\mathbf{a}$. and so we can no longer assume elements with $\dim \ker D_{\mathbf{a}}\mathcal{F} > 1$ exist. In such a scenario, we have two alternatives:

- Take a single element $\mathbf{a}$ and run FindFunction on $(\mathbf{a}, \mathbf{b})$ for all $\mathbf{b} \in \mathbb{F}_q^{n+m}$ until we find the isometry. This deterministic process has a time complexity of $\mathcal{O}(q^{(n+m)} \cdot C_{\mathrm{FF}})$. The memory requirements of this algorithm are negligible, since we do not build lists of elements.

- Alternatively, note that in this case $n \leqslant 2(k+m)$. Thus, we can also use the result of Lemma 12, and instead of an $\mathsf{MCE}(n,m,k)$ instance, we can solve an $\mathsf{MCE}(k,m,n)$ instance using Algorithm 28. In this case we end up with a complexity of $\tilde{\mathcal{O}}(q^{\frac{2}{3}(k+m)})$. However, for the given regime of parameters, this is always larger than $\tilde{\mathcal{O}}(q^{(n+m)})$, so the first (deterministic) approach is better.

4.3 SECOND ALGORITHM

The algorithm that we presented in the previous section does not take advantage of the bilinear structure of an instance of MCE when viewed as hQMLE. In such a case, the differential $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}$ of a k-dimensional bilinear form admits a special structure.

Lemma 18. Let $\mathcal{F}(\mathbf{x},\mathbf{y})$ be a k-dimensional bilinear form with $\mathbf{x} \in \mathbb{F}_q^m$ and $\mathbf{y} \in \mathbb{F}_q^n$. Let $\mathbf{F_a}$ denote the $k \times n$ matrix of the linear map $\mathcal{F}(\mathbf{a},-) : \mathbb{F}_q^n \to \mathbb{F}_q^k$ for a fixed $\mathbf{a} \in \mathbb{F}_q^m$. Similarly let $\mathbf{F_b}$ denote the $k \times m$ matrix of the linear map $\mathcal{F}(-,\mathbf{b}) : \mathbb{F}_q^m \to \mathbb{F}_q^k$ for a fixed $\mathbf{b} \in \mathbb{F}_q^n$. Then

$$D_{(\mathbf{a},\mathbf{b})}\mathcal{F}(\mathbf{x},\mathbf{y}) = \begin{pmatrix} \mathbf{F_b} & \mathbf{F_a} \end{pmatrix} \begin{pmatrix} \mathbf{x}^\top \\ \mathbf{y}^\top \end{pmatrix}.$$

Proof. By bilinearity, $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}(\mathbf{x},\mathbf{y}) := \mathcal{F}(\mathbf{x}+\mathbf{a},\mathbf{y}+\mathbf{b}) - \mathcal{F}(\mathbf{x},\mathbf{y}) - \mathcal{F}(\mathbf{a},\mathbf{b})$ equals $\mathcal{F}(\mathbf{a},\mathbf{y}) + \mathcal{F}(\mathbf{x},\mathbf{b}) = \mathbf{F_a}\mathbf{y}^\top + \mathbf{F_b}\mathbf{x}^\top$. □

Similarly for $\mathcal{P}$, we use the notation $\mathbf{P_a}$ and $\mathbf{P_b}$. The equivalence in such a case becomes $\mathcal{P}(\mathbf{x},\mathbf{y}) = \mathcal{F}(\mathbf{x}\mathbf{A}, \mathbf{y}\mathbf{B}^\top)\mathbf{T}$, with $\mathbf{A},\mathbf{B}$ precisely the matrices from the MCE instance. Then, as $\mathcal{F}$ and $\mathcal{P}$ are bilinear, one can see SAMPLESET in Algorithm 28 as sampling both $\mathbf{a} \in \mathbb{F}_q^n$ and $\mathbf{b} \in \mathbb{F}_q^m$ at the same time as one $(\mathbf{a},\mathbf{b}) \in \mathbb{F}_q^{n+m}$, until $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}$ has a kernel of dimension κ. However in the bilinear case, $\mathbf{a}$ influences only the matrix $\mathbf{F_a}$, and $\mathbf{b}$ influences only $\mathbf{F_b}$. Hence, we can sample $\mathbf{a} \in \mathbb{F}_q^m$ and $\mathbf{b} \in \mathbb{F}_q^n$ separately. Even better, we can apply ideas from Algorithm 28 to the smaller universes $U_\mathbf{a} = \mathbb{F}_q^{k\times n}(q)$ and $U_\mathbf{b} = \mathbb{F}_q^{k\times m}(q)$, where $\mathbf{F_a}$ and $\mathbf{F_b}$ live. By finding well-chosen predicates in these smaller universes, we can hope to find collisions faster.

We first analyse properties of $\mathbf{F_a}$ and $\mathbf{F_b}$. Let $\mathfrak{F}_a$ be the set of elements $\mathbf{a}$ for which $\dim\ker \mathbf{F_a}$ is non-trivial, i.e.

$$\mathfrak{F}_a = \{\mathbf{a} \in \mathbb{F}_q^m \mid \dim\ker \mathcal{F}(\mathbf{a},-) > 0\},$$

and similarly define $\mathfrak{F}_b$

$$\mathfrak{F}_b = \{\mathbf{b} \in \mathbb{F}_q^n \mid \dim \ker \mathcal{F}(-, \mathbf{b}) > 0\}.$$

For $\mathcal{P}$, we define $\mathfrak{P}_a$ and $\mathfrak{P}_b$ similarly. For isomorphic bilinear forms $\mathcal{F}$ and $\mathcal{P}$, these sets have special properties.

Lemma 19. Let $(\mathbf{A}, \mathbf{B}, \mathbf{T}) : \mathcal{F} \to \mathcal{P}$ be an isomorphism between two k-tuples of bilinear homogenous quadratic polynomials $\mathcal{F}$ and $\mathcal{P}$, such that $\mathcal{P}(\mathbf{x}, \mathbf{y}) = \mathcal{F}(\mathbf{xA}, \mathbf{yB}^\top)\mathbf{T}$. We have the following properties:

1. Given $\mathbf{a} \in \mathfrak{F}_a$ and any $\mathbf{b} \in \ker \mathbf{F_a}$, we get $\mathcal{F}(\mathbf{a}, \mathbf{b}) = 0$.
2. $\mathfrak{F}_b$ is completely determined by $\mathfrak{F}_a$, as $\mathfrak{F}_b = \bigcup_{\mathbf{a} \in \mathfrak{F}_a} \ker \mathbf{F_a}$.
3. For $\mathbf{a} \in \mathbb{F}_q^n$ and $\mathbf{y} \in \mathbb{F}_q^m$, we have $\mathbf{P_a}(\mathbf{y}) = \mathbf{F_{aA}}(\mathbf{yB}^\top)\mathbf{T}$.
4. For $\mathbf{a} \in \mathbb{F}_q^n$, we get $\ker \mathbf{P_a} = \ker \mathcal{F}_{\mathbf{aA}} \cdot \mathbf{B}^\top$.
5. The isomorphism $(\mathbf{A}, \mathbf{B}, \mathbf{T})$ induces the bijections

$$\mathfrak{P}_a \to \mathfrak{F}_a : \mathbf{a} \mapsto \mathbf{aA}, \quad \mathfrak{P}_b \to \mathfrak{F}_b : \mathbf{b} \mapsto \mathbf{bB}^\top.$$

Proof.

1. $\mathbf{b} \in \ker \mathbf{F_a}$ is equivalent by definition to $\mathbf{F_a b}^\top = \mathcal{F}(\mathbf{a}, \mathbf{b}) = \mathbf{0}$.
2. This follows directly from 1.: $\mathbf{b} \in \mathfrak{F}_b$ only if there exists an $\mathbf{a} \in \mathfrak{F}_a$ such that $\mathcal{F}(\mathbf{a}, \mathbf{b}) = \mathbf{0}$. But then $\mathbf{b} \in \ker \mathbf{F_a}$ for this specific $\mathbf{a}$.
3. Per definition $\mathbf{P_a}(\mathbf{y}) = \mathcal{P}(\mathbf{a}, \mathbf{y}) = \mathcal{F}(\mathbf{aA}, \mathbf{yB}^\top)\mathbf{T} = \mathbf{F_{aA}}(\mathbf{yB}^\top)\mathbf{T}$.
4. This follows directly from 3.: as $\mathbf{T}$ is invertible, it does not affect the kernels, so $\mathbf{y} \in \ker \mathbf{P_a}$ if and only if $\mathbf{yB}^\top \in \ker \mathbf{F_{aA}}$
5. This follows directly from 4.: Given $\mathbf{a} \in \mathfrak{P}_a$ we get $\mathbf{aA} \in \mathfrak{F}_a$ and vice versa as $\mathbf{A} \in \mathrm{GL}_m(q)$. A similar argument gives $\mathfrak{F}_b \to \mathfrak{P}_b$.

□

Lemma 19 shows that $\mathbf{a} \in \mathfrak{F}_a$ and $\mathbf{b} \in \mathfrak{F}_b$ describe all non-trivial roots $(\mathbf{a}, \mathbf{b})$ of a given $\mathcal{F}$. For an instance $(\mathbf{A}, \mathbf{B}, \mathbf{T}) : \mathcal{F} \to \mathcal{P}$, Item 5 shows that non-trivial roots are mapped bijectively by $(\mathbf{A}, \mathbf{B}, \mathbf{T})$. Such non-trivial roots can be used to find collisions more easily between $\mathcal{F}$ and $\mathcal{P}$. However, this requires that instances $\mathcal{F} \to \mathcal{P}$ *have* non-trivial roots. We can get an estimate on the sizes of $\mathfrak{F}_a$, $\mathfrak{F}_b$, $\mathfrak{P}_a$, and $\mathfrak{P}_b$ for given parameters n, m, and k, in the following way.

Lemma 20. When $k \geqslant n$, $|\mathfrak{F}_a| = |\mathfrak{P}_a| \approx q^{2n-k-1}$ and $|\mathfrak{F}_b| = |\mathfrak{P}_b| \approx q^{2m-k-1}$.

Proof. By Lemma 19, we get $|\mathfrak{F}_a| = |\mathfrak{P}_a|$. Then, using Lemma 16, we see that the size of these sets is dominated by elements $\mathbf{a}$ with $\kappa = \dim\ker \mathbf{F_a} = 1$ (a one-dimensional kernel). From the same lemma, the density of $\kappa = \dim\ker \mathbf{F_a} = 1$ elements is $d_1 = q^{-(1+1\cdot(k-n))}$. Hence we expect $d_1 \cdot q^n = \Theta(q^{2n-k-1})$ such elements. A similar argument gives $|\mathfrak{F}_b| = |\mathfrak{P}_b|$ as $\Theta(q^{2m-k-1})$. □

We can apply Lemma 20 to different parameter regimes to get an understanding of when non-trivial roots exist.

Corollary 21. Assuming $n = m$, an $\mathsf{MCE}(n, m, k)$ instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$ has an expected value $\mathcal{E}_{n,m,k,q}$ of non-trivial roots

- when $k < 2n$, with $\mathcal{E}_{n,m,k,q} = \Theta(q^{2n-k-1})$,
- when $k = 2n$, with $\mathcal{E}_{n,m,k,q} = \Theta(\frac{1}{q})$,
- when $k > 2n$, with $\mathcal{E}_{n,m,k,q} = \Theta(\frac{1}{q^{k-2n+1}})$.

From these results, we can expect non-trivial roots for an $\mathsf{MCE}(n, m, k)$ instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$ whenever $k \leqslant n + m$. These non-trivial roots can be seen as a suitable predicate on the smaller universes $\mathcal{U}_{\mathbf{a}}$ and $\mathcal{U}_{\mathbf{b}}$: we search for collisions $(\mathbf{a}, \mathbf{b}) \times (\mathbf{aA}, \mathbf{bB}^\top)$, where $(\mathbf{a}, \mathbf{b})$ is a non-trivial root of $\mathcal{P}$, and $(\mathbf{aA}, \mathbf{bB}^\top)$ of $\mathcal{F}$. Given such a collision, we proceed as in Section 4.2.

The following result shows that we always find such a collision if $\mathcal{F}$ and $\mathcal{P}$ have non-zero roots with certainty.

Lemma 22. Let $m \leqslant n$ and $k \leqslant n + m$. Let $\mathfrak{F}_a$, $\mathfrak{F}_b$ and $\mathfrak{P}_a$, $\mathfrak{P}_b$ describe the non-trivial roots of an $\mathsf{MCE}(n, m, k)$ instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$. Let $\mathbf{x} = (\mathbf{a}, \mathbf{b}) \in \mathfrak{F}_a \times \mathfrak{F}_b$, then looping over $\mathbf{y} \in \mathfrak{P}_a \times \mathfrak{P}_b$ gives a collision $(\mathbf{x}, \mathbf{y})$ with certainty.

Proof. This follows quickly from Lemma 19. We have $\mathbf{x} \in \mathfrak{F}_a \times \mathfrak{F}_b$ and two bijections $\mathfrak{F}_a \to \mathfrak{P}_a$ and $\mathfrak{F}_b \to \mathfrak{P}_b$, so $\mathbf{x}$ is mapped to some $\mathbf{y} \in \mathfrak{P}_a \times \mathfrak{P}_b$. As this set is finite, we can loop over it in a finite number of steps until we find the collision. □

Therefore, as soon as we have non-trivial roots, we can use a single one of them to find a collision. This leads to the following pseudo-algorithm:

1. Compute $\mathfrak{F}_b$ by computing $\ker \mathbf{F_b}$ for all $\mathbf{b} \in \mathbb{F}_q^m$.

2. If $\mathfrak{F}_b$ is non-empty, compute $\mathfrak{F}_a$ using Lemma 19-2, and repeat for $\mathfrak{P}_a$ and $\mathfrak{P}_a$.
3. Sample a single $\mathbf{x} \in \mathfrak{F}_a \times \mathfrak{F}_b$.
4. Perform FindFunction$(\mathbf{x}, \mathbf{y})$ for all $\mathbf{y} \in \mathfrak{P}_a \times \mathfrak{P}_b$ until the solver finds μ.

Corollary 23. Let $m \leqslant n$ and $k \leqslant n + m$. The above algorithm terminates successfully and has a total complexity of $\mathcal{O}(q^m \cdot C_{\mathbb{P}} + q^{2(n+m-k-1)} \cdot C_{\mathrm{FF}})$, where $C_{\mathbb{P}}$ denotes the cost of computing $\ker \mathbf{F_b}$ and C_{FF} denotes the cost of a single query to FindFunction.

Proof. Building $\mathfrak{F}_b$ and $\mathfrak{P}_b$ has a complexity of $\mathcal{O}(q^m \cdot C_{\mathbb{P}})$, and these directly give us $\mathfrak{F}_a$ and $\mathfrak{P}_a$ by Lemma 19. Then, for every step in the loop we get a query to FindFunction. By Lemma 20, the size of $\mathfrak{P}_a \times \mathfrak{P}_b$ is at most $\mathcal{O}(q^{2(n+m-k-1)})$. □

We will see later in Section 5 that the dominating complexity is $q^m \cdot C_{\mathbb{P}}$ as for specific parameters (k, n, m) the number of queries z can be reduced so that $z \cdot C_{\mathrm{FF}} < q^m$. As $C_{\mathbb{P}}$ is polynomial, we get a complexity of $\tilde{\mathcal{O}}(q^m)$ for such instances.

For efficiency, one can decrease further the number of queries to FindFunction by applying other, secondary predicates. For example, the sets $\mathfrak{F}_a \times \mathfrak{F}_b$ and $\mathfrak{P}_a \times \mathfrak{P}_b$ can be split into zeros $\mathfrak{F}^{\mathbf{0}} = \{\mathbf{x} \in \mathbb{F}_q^{n+m} \mid \mathcal{F}(\mathbf{x}) = \mathbf{0}\}$ and non-zeros $\mathfrak{F} = \mathfrak{F}_a \times \mathfrak{F}_b \setminus \mathfrak{F}^{\mathbf{0}}$, which reduces the collision search to each of these sets. Another secondary predicate is to only use elements $\mathbf{a}$ with $\dim \ker \mathbf{F_a} = \kappa$ for some specific value $\kappa > 0$. Algorithm 29 describes an MCE-solver for instances with non-trivial roots.

Practically, since the algorithm is deterministic, we do not need to build and store the list $\mathfrak{F}$. We only need to find one element from it. However, for iterating through the list $\mathfrak{P}$, S_a and S_b need to be stored. The estimated size of these lists is $q^{n+m-k-1}$.

The next theorem summarises the conditions and cost of Algorithm 29 for solving MCE.

Theorem 24. Assuming a solver for the inhomogenous case of QMLE with cost C_{FF}, an MCE(n, m, k) instance over $\mathbb{F}_q$ with $m \leqslant n$ and $k \leqslant n + m$, in which case roots exist for $\mathcal{F}$ and $\mathcal{P}$ with overwhelming probability, can be solved using Algorithm 29 with $\mathcal{O}\left(q^m \cdot C_{\mathbb{P}}\right)$ operations in SampleZeros and z queries to the solver. This amounts to a total time complexity of $\mathcal{O}\left(q^m \cdot C_{\mathbb{P}} + z \cdot C_{\mathrm{FF}}\right)$. The memory complexity of the algorithm is $\mathcal{O}(q^{n+m-k-1})$.

Algorithm 29 Bilinear MCE-Solver, assuming $n \geqslant m$.

procedure SAMPLEZEROS($\mathcal{F}$)
  $S, S_a, S_b \leftarrow \emptyset$
  for all $\mathbf{b} \in \mathbb{F}_q^m$ **do**
    if $\dim \ker \mathbf{F_b} > 0$ **then**
      $S_b \leftarrow S_b \cup \{\mathbf{b}\}$
  $S_a \leftarrow \cup_{\mathbf{b} \in S_b} \ker \mathbf{F_b} \setminus \{0\}$
  $S \leftarrow S_a \times S_b$
  return S

procedure COLLISIONFIND($\mathcal{F}, \mathcal{P}$)
  $\mathfrak{F} \leftarrow$ SAMPLEZEROS($\mathcal{F}$)
  $\mathfrak{P} \leftarrow$ SAMPLEZEROS($\mathcal{P}$)
  $\mathbf{x} \xleftarrow{\$} \mathfrak{F}$
  for all $\mathbf{y} \in \mathfrak{P}$ **do**
    $\mu \leftarrow$ FINDFUNCTION($\mathbf{x}, \mathbf{y}$)
    if $\mu \neq \perp$ **then**
      return solution μ
  return $\perp$

We will show in Section 5 that, even though C_{FF} is *not* polynomial-time, the dominating factor in this complexity is still $q^m \cdot C_{\mathbb{P}}$, where $C_{\mathbb{P}}$ is the cost to compute the kernel of an $m \times k$ matrix.

The regime of operation of Theorem 24 seems to imply that we can use it only if $k \leqslant n + m$. However, note that if $k > n + m$ then $n \leqslant k + m$. Hence, by Lemma 12, we can turn the given $\mathsf{MCE}(n, m, k)$ instance into an $\mathsf{MCE}(k, m, n)$ instance and solve this instance using Algorithm 29. This results in a complexity of $\tilde{\mathcal{O}}(q^m)$, where we assume $m = \min\{m, n, k\}$. Thus, we obtain the following general theorem which is our main result about the practical complexity of solving MCE.

Theorem 25. An $\mathsf{MCE}(n, m, k)$ instance over $\mathbb{F}_q$ can be solved using Algorithm 29 in time $\tilde{\mathcal{O}}\left(q^{\min\{m,n,k\}}\right)$.

§5 Filling the Gaps in the Complexity Analysis

The cost $C_{\mathbb{P}}$ is polynomial in all of the cases because it either requires computing the rank of a linear map or sampling a random element from a set. The FINDFUNCTION in Algorithms 28 and 29 checks whether a given pair of vectors is a collision, and if so, it returns the solution to the MCE instance. This is done by solving an instance of the inhBMLE that has the same solutions as the input MCE instance. Thus, to estimate the value of C_{FF}, we analyse the complexity of inhBMLE on these instances, by relying on algorithms that have been developed for the inhQMLE case with $N = k$.

5.1 ALGORITHMS FOR INHQMLE

The two algorithms described in this section have been used for tackling the inhQMLE problem within the birthday-based algorithm for hQMLE [68, 69]. Their analysis is thus important to estimate C_{FF}. In Section 5.2, we adapt this analysis for the inhBMLE case with arbitrary k and N and we see how this affects Algorithms 28 and 29 for different parameter sets.

The Gröbner bases attack.

The algebraic attack on the inhQMLE problem first reduces $\mathcal{P}(\mathbf{x})\mathbf{T}^{-1} = \mathcal{F}(\mathbf{x}\mathbf{S})$, with $\mathbf{S}$ and $\mathbf{T}$ unknown, to a system of polynomial equations. By rewriting the problem in matrix form we obtain the constraints

$$\begin{aligned} \sum_{1 \leqslant r \leqslant k} \widetilde{T}_{rs}\mathbf{P}^{(r)} &= \mathbf{S}\mathbf{F}^{(s)}\mathbf{S}^{\top}, \ \forall s, 1 \leqslant s \leqslant k, \\ \mathbf{P}^{[1]}\mathbf{T}^{-1} &= \mathbf{S}\mathbf{F}^{[1]}, \\ \mathbf{P}^{[0]}\mathbf{T}^{-1} &= \mathbf{F}^{[0]}, \end{aligned} \tag{42}$$

where $\mathbf{F}^{[1]} \in \mathbb{F}_q^{N \times k}$ and $\mathbf{P}^{[1]} \in \mathbb{F}_q^{N \times k}$ describe the degree-1 homogeneous part of an inh(Q/B)MLE instance and $\mathbf{F}^{[0]} \in \mathbb{F}_q^{k}$ and $\mathbf{P}^{[0]} \in \mathbb{F}_q^{k}$ describe the degree-0 part. We will denote the subsystem of equations derived from the degree-d homogeneous part as $\mathcal{S}_d$. The resulting system can be solved using Gröbner basis algorithms and this is referred to as the Gröbner attack [173]. This work has two useful observations: first, that $\mathbf{S}$ and $\mathbf{T}$ are common solutions to homogeneous parts of separate degrees of an inhQMLE instance, also proven in [67, Lemma 1], and second, that we get a lower-degree system where we solve for $\mathbf{T}^{-1}$. by moving $\mathbf{T}$ to the other side of the equality.

The complexity of Gröbner basis algorithms depends foremost on the *degree of regularity*, which is usually hard to estimate, but can sometimes be observed through experimental work. Such experiments, applied to inhQMLE instances, imply that the system is solved at degree three. A degree-three linearized system in n variables is represented by a matrix of size roughly n^3 and thus, Gaussian elimination on such a system is performed in $\mathcal{O}(n^{3\omega})$ operations, where ω is the linear algebra constant. This reasoning leads to the assumption that there exists a polynomial-time solver for the inhomogeneous case of QMLE. Another empirical observation made in [173] is that the time to construct the system exceeds the time of the Gröbner basis computation. Since the generation of the system is known to be polynomial, this suggests that the

Gröbner basis computation is performed in polynomial time as well. However, these experiments are performed on *random* inhomogeneous instances of the QMLE problem.

In the birthday-based approach for solving QMLE, $\mathbf{F}^{[1]}$, $\mathbf{P}^{[1]}$, $\mathbf{F}^{[0]}$ and $\mathbf{P}^{[0]}$ are obtained from a collision [69]. Specifically, if we have a collision on $\mathbf{x} \in \mathbb{F}_q^N$ and $\mathbf{y} \in \mathbb{F}_q^N$ such that $\mathbf{y} = \mathbf{x}\mathbf{S}$, they are obtained as

$$\mathbf{F}^{[1]} = D_{\mathbf{y}}\mathcal{F}, \qquad \mathbf{P}^{[1]} = D_{\mathbf{x}}\mathcal{P},$$
$$\mathbf{F}^{[0]} = \mathcal{F}(\mathbf{y}), \qquad \mathbf{P}^{[0]} = \mathcal{P}(\mathbf{x}).$$

Instances of inhQMLE derived from a collision are, on average, harder to solve than random inhQMLE instances. Recall that in Algorithm 29 the instances of inhQMLE are chosen such that $\dim\ker D_{\mathbf{y}}\mathcal{F} = \dim\ker D_{\mathbf{x}}\mathcal{P} = \kappa$. Hence, the number of linearly independent equations in $\mathcal{S}_1$ is exactly $k(N-\kappa)$, instead of the expected kN on average. The size of $\mathcal{S}_0$ can also depend on the predicate that we choose for the birthday-based algorithm. For instance, when we use the predicate of searching for a collision between the non-trivial roots of $\mathcal{P}$ and $\mathcal{F}$, we obtain no equations in $\mathcal{S}_0$. Additionally, since $\mathbf{F}^{[1]}$ (resp. $\mathbf{P}^{[1]}$) and $\mathbf{F}^{[0]}$ (resp. $\mathbf{P}^{[0]}$) are obtained respectively from computing the differential of and evaluating $\mathcal{F}$ (resp. $\mathcal{P}$) at a given point, $\mathcal{S}_1$ and $\mathcal{S}_0$ are not as independent from $\mathcal{S}_2$ as they would be in the random case. It is difficult to estimate the complexity of solving these instances compared to solving random instances with the same structure.

Figure 26 shows experiments confirming our intuition that the complexity of collision-derived instances is worse than that of random ones. This implies that we can not rely on the experimental observations in [173] to estimate the complexity of these specific instances. We conclude that, in contrast with the literature, we cannot assume that C_{FF} is polynomial when the Gröbner attack is used.

The matrix-pencil attack.

The matrix-pencil attack was proposed in Bouillaguet's thesis [68] and used for the implementation of the birthday-based attack [69]. This algorithm has a complexity of $\mathcal{O}(N^6)$ with non-negligible success probability for random inhQMLE instances when $N = k$. Its complexity for inhQMLE instances derived from a collision attack depends strongly on the parameter κ. We give a general description of the approach.

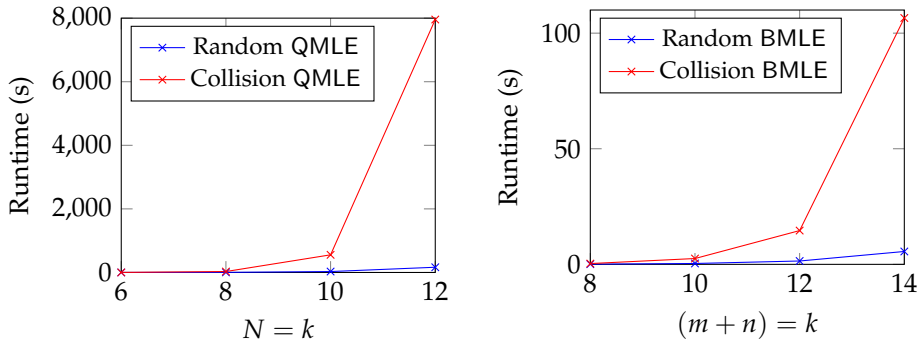


Figure 26: Comparison of runtime for solving random and collision-derived inh(Q/B)MLE instances using the Gröbner attack. Results are averaged over 50 runs.

The first step is to retrieve a basis of the solution space V of the subsystem of linear equations $\mathcal{S}_1$. Let $\ell = \dim V$ and let $(\mathbf{S}^{[1]}, \mathbf{T}^{[1]}), \ldots, (\mathbf{S}^{[\ell]}, \mathbf{T}^{[\ell]})$ be a basis of V. Once the solution space of $\mathcal{S}_1$ is known, in order to find the solution space of the overall system one rewrites $\mathcal{S}_2$ as a system in ℓ variables. Concretely, this is done by replacing $\mathbf{S}$ and $\mathbf{T}$ by $\sum_{i=1}^{\ell} x_i \mathbf{S}^{[i]}$ and $\sum_{i=1}^{\ell} x_i \mathbf{T}^{[i]}$ in Equation (42) and then looking for solutions in variables $x_1, \ldots, x_\ell$. This standard approach is also described in [67]. A key idea in the matrix-pencil attack is to use the knowledge of $\mathbf{F}^{[1]}/\mathbf{P}^{[1]}$ and $\mathbf{F}^{[0]}/\mathbf{P}^{[0]}$ to find a (second) collision and double the number of linear equations in $\mathcal{S}_1$. Supposing that there exists $\mathbf{x}'$ such that $\mathbf{x}'\mathbf{P}^{[1]} = \mathbf{P}^{[0]}$, we infer that there also exists $\mathbf{y}'$ such that $\mathbf{y}'\mathbf{F}^{[1]} = \mathbf{F}^{[0]}$ and that $\mathbf{y}' = \mathbf{x}'\mathbf{S}$. We can thus append the equations obtained from $(D_{\mathbf{x}'}\mathcal{P})\mathbf{T}^{-1} = \mathbf{S}(D_{\mathbf{y}'}\mathcal{F})$ to $\mathcal{S}_1$. After applying this technique, the resulting system is usually highly overdetermined and can be solved through direct linearization. The most favorable case is when $\mathbf{x}'$ and $\mathbf{y}'$ are uniquely identified. However, if $\dim \ker \mathbf{F}^{[1]} = \kappa > 1$, then $\mathbf{x}'$ is chosen arbitrarily and we loop through the q^κ possible values for $\mathbf{y}'$. The complexity of the algorithm is $\mathcal{O}(q^\kappa \ell^2 N^4)$, under the condition that $\ell(\ell+1)/2 \leq |\mathcal{S}_2|$. Another condition for the success of this approach is that $\mathcal{P}(\mathbf{x}) \neq 0$ and there is an $\mathbf{x}$ such that $\mathbf{x}D_{\mathbf{x}}\mathcal{P} = \mathcal{P}(\mathbf{x})$, because this assumption is used to find the second collision. As per the analysis in [68], the probability that the condition for success is met is $1 - 1/q + 1/q^3 + \mathcal{O}(1/q^6)$.

5.2 THE COMPLEXITY OF INHOMOGENEOUS BMLE

In the following analysis, we use the matrix-pencil algorithm as the inhBMLE solver, as it seems to outperform the Gröbner attack and we have a better understanding of its complexity for these specific instances.

The case $k \leq n + m$.

Based on the analysis in Section 4.3 for the purpose of usage in Algorithm 29, and Lemma 12, we assume $k \leq m + n$ and $m \leq \min(m, n, k)$, hence $n + m - k \leq n$.

The complexity of Algorithm 29 is dominated by the SampleZeros function, as long as the complexity of the inhBMLE solver and the size over which it loops does not surpass $\mathcal{O}(q^m)$. In the matrix-pencil algorithm, we cannot use the zero subsets $\mathfrak{F}^{\mathbf{0}}$ and $\mathfrak{P}^{\mathbf{0}}$, as this contradicts its condition for success $\mathcal{P}(\mathbf{x}) \neq 0$. The non-zeros subsets $\mathfrak{F}$ and $\mathfrak{P}$ can be used with a small adjustment to the algorithm: after finding a basis of the solution space of $\mathcal{S}_1$, we rewrite and solve through linearization the system comprised of both $\mathcal{S}_2$ and $\mathcal{S}_0$. Note that $\mathfrak{F}$ and $\mathfrak{P}$ are non-empty only when the instance has at least two roots. We are free to choose κ as long the sets are non-empty. Using Lemma 16, with $0 \leq (n + m) - k \leq n$, this means that there is some (bounded) κ so that we loop over a set smaller than $q^{m-\kappa}$, and so, the total complexity is smaller than $\mathcal{O}(q^m)$.

The case $n + m < k < 2(n + m)$.

This case is not relevant for Algorithm 29, but it is for Algorithm 28. Since the complexity of the inhBMLE solver contains a non-negligible factor of q^κ, the choice of κ needs to be adapted, so that the running times of SampleSet and CollisionFind are equal. Let $N = n + m$ and let $r = N - k$. The optimal κ is chosen such that

$$q^{\frac{N-(\kappa^2+\kappa r)}{2}} \cdot q^{\kappa^2+\kappa r} \approx q^{N-(\kappa^2+\kappa r)} \cdot q^{\kappa}.$$

This gives us $\kappa = \frac{k-(n+m+\sqrt{\delta})}{2} + \frac{1}{3}$, with $\delta = (k - (n + m))^2 + \frac{4}{3}(k + \frac{1}{3})$. The complexity of the overall algorithm with this optimal choice for κ is then

$$q^{\frac{n+m}{2} + \frac{k-\sqrt{\delta}}{6} + \frac{1}{9}}.$$

We get that $\sqrt{\delta} \geqslant |k - (n+m)|$ and so for all values of k between $n+m$ and $2(n+m)$, the term $k - \sqrt{\delta}$ is bounded by $n+m$, and hence this gives a bound on the complexity by $\mathcal{O}(q^{\frac{2}{3}(n+m)+\frac{1}{9}})$. The term $\frac{1}{9}$ adds a few bits at most to this complexity, but is negligible for most cryptographic purposes.

The case $k \geq 2(n+m)$.

When $k \geq 2(n+m)$, as per Lemma 16, the probability that there exist elements with $\dim \ker D_{(\mathbf{a},\mathbf{b})}\mathcal{F} > 1$ is extremely small, which is why we can not define a distinguishing predicate for Algorithm 28 and $\kappa = 1$ with overwhelming probability. In this case, the complexity of the matrix-pencil algorithm is

$$\mathcal{O}(q \cdot (m+n)^6),$$

as with random inhBMLE instances.

§6 Experimental Results

To confirm our theoretical findings, we solved randomly generated positive instances of the MCE problem, using the two approaches presented in this paper. First, we implement the birthday-based Algorithm 28 in three steps. (1) We randomly generate a positive instance $\mathcal{I}_{\mathsf{MCE}}(\mathcal{C}, \mathcal{D})$ of $\mathsf{MCE}(n, m, k)$ and reduce it to an instance $\mathcal{I}_{\mathsf{hQMLE}}(\mathcal{F}, \mathcal{P})$ of $\mathsf{hQMLE}(m+n, k)$. (2) We build the two sample sets for a predefined predicate $\mathbb{P}$ and we combine them to create pairs of potential collisions. (3) For each pair we create an inhQMLE instance and we query an inhQMLE solver until it outputs a solution for the maps $\mathbf{S}$ and $\mathbf{T}$. Our implementation is built on top of the open source birthday-based hQMLE solver from [68], which is implemented in MAGMA [66].

Table 16 shows running times for solving the MCE problem using Algorithm 28. The goal of this first experiments was to confirm that there is a parameter choice where the probability of success of the algorithm surpasses $1 - 1/e$ and that our running times are comparable to the ones given in [69]. These experiments are done with the parameter $q = 2$ and all results are an average of 50 runs.

The second approach, described in Section 4.3, uses the bilinear structure of hQMLE instances derived from MCE instances to have an improved algorithm for building the sample sets and a more precise predicate that results in fewer queries to the inhQMLE solver. The consequence of these two improvements to

Table 16: Experimental results on solving the MCE problem using Algorithm 28.

$m = n$	k	κ	Size Set	Runtime (s) SAMPLESET	Runtime (s) inhQMLE solver	Success probability
10	20	5	2	21	3154	0.70
11	22	5	3	31	2004	0.63
12	24	5	6	76	13873	0.73

the runtime can be observed in Table 17 where we show experimental results of Algorithm 29 using the non-zeros subsets.

Recall that this approach can only be used when there exist at least two roots of $\mathcal{F}$ and $\mathcal{P}$. Otherwise, the sampled sets contain only the trivial root and we use Algorithm 28 to solve the instance. Table 17 shows results of the case when the sets are non-trivial and the probability of this case for the given parameters is shown in the last column. For efficiency, we take the minimal subset with a common dimension of the kernel of $\mathbf{F}_{\mathbf{b}}$, and when looking for collisions, we are careful to skip pairs $(\mathbf{ab}, \mathbf{a'b'})$ where $\dim\ker \mathbf{F}_{\mathbf{b}} = \dim\ker \mathbf{P}_{\mathbf{b'}}$ but $\dim\ker D_{(\mathbf{a},\mathbf{b})}\mathcal{F} \neq \dim\ker D_{(\mathbf{a'},\mathbf{b'})}\mathcal{P}$. In these experiments, $q = 3$ and all results are an average of 50 runs.

Table 17: Experimental results on solving the MCE problem using the non-zeros-subsets variant of Algorithm 29. The last column denotes the percentage of instances with at least two non-trivial roots.

$m = n$	k	Size Set	Runtime (s) SAMPLEZEROS	Runtime (s) inhQMLE solver	% w. roots
8	15	10.40	0.56	175.34	24
	14	35.56	0.60	236.12	68
9	17	12.00	1.74	396.04	22
	16	37.97	1.72	1020.25	70
10	19	25.60	5.13	2822.32	14
	18	36.72	5.05	1809.09	82

Our experiments confirm that Algorithm 29 outperforms Algorithm 28 for solving MCE instances with non-trivial roots.

ACKNOWLEDGEMENTS. We thank Charles Bouillaguet for providing the implementation resulting from [68].

XIII

TAKE YOUR MEDS: DIGITAL SIGNATURES FROM MATRIX CODE EQUIVALENCE

In this chapter, we show how to use the Matrix Code Equivalence (MCE) problem as a new basis to construct signature schemes. This extends previous work on using isomorphism problems for signature schemes, a trend that has recently emerged in post-quantum cryptography. Our new formulation leverages a more general problem and allows for smaller data sizes, achieving competitive performance and great flexibility. Using MCE, we construct a zero-knowledge protocol which we turn into a signature scheme named Matrix Equivalence Digital Signature (MEDS). We provide an initial choice of parameters for MEDS, tailored to NIST's Category 1 security level, yielding public keys as small as 2.8 kilobyte and signatures ranging from 18 kilobyte to just around 6.5 kilobyte, along with a reference implementation in C.

§1 Introduction

Post-Quantum Cryptography (PQC) comprises all the primitives that are believed to be resistant against attackers equipped with a considerable quantum

This chapter is based on the paper

> Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. "Take your MEDS: digital signatures from matrix code equivalence". In: *International conference on cryptology in Africa*. Springer. 2023, pp. 28–52.

Some content has been removed such as the overlap with Chapter XII and a section on ring signatures. For the rest, changes are editorial in nature. At the time of writing this thesis, cryptanalysis of MCE has significantly advanced [109, 289, 318] and current parameters for MEDS are wildly different from the ones presented here. I have left the technical content of this chapter 'as is' to reflect our understanding at the time of publishing of the paper.

computing power. Several such schemes have been around for a long time [305, 320], some being in fact almost as old as RSA [269]; however, the area itself was not formalized as a whole until the early 2000s, for instance with the first edition of the PQCrypto conference [294]. The area has seen a dramatic increase in importance and volume of research over the past few years, partially thanks to NIST's interest and the launch of the PQC Standardization process in 2017 [296]. After 4 years and 3 rounds of evaluation, the process has crystallized certain mathematical tools as standard building blocks (e.g. lattices, linear codes, multivariate equations, isogenies etc.). Some algorithms [164, 180, 211] have now been selected for standardization, with an additional one or two to be selected among a restricted set of alternates [4, 8, 11] after another round of evaluation. While having a range of candidates ready for standardization may seem satisfactory, research is still active in designing PQC primitives. In particular, NIST has expressed the desire for a greater diversity among the hardness assumptions behind signature schemes, and announced a partial re-opening of the standardization process for precisely the purpose of collecting non-lattice-based protocols.

Cryptographic group actions are a popular and powerful instrument for constructing secure and efficient cryptographic protocols. The most well-known is, without a doubt, the action of finite groups on the integers modulo a prime, or the set of points on an elliptic curve, which give rise to the *Discrete Logarithm Problem (DLP)*, i.e. the backbone of public-key cryptography. Recently, proposals for post-quantum cryptographic group actions started to emerge, based on the tools identified above: for instance, isogenies [96], linear codes [56], trilinear forms [362] and even lattices [210]. All of these group actions provide very promising solutions for cryptographic schemes, for example signatures [30, 144, 362], ring signatures [29, 53] and many others; at the same time, they are very different in nature, with unique positive and negative aspects.

Contributions

In this work, we formalize a new cryptographic group action based on the notion of *Matrix Code Equivalence*. This is similar in nature to the *code equivalence* notion at the basis of LESS [30, 56], and in fact belongs to a larger class of isomorphism problems that include, for example, the lattice isomorphism problem, and the well-known isomorphism of polynomials [305]. The hardness of MCE was studied in [138] and Chapter XII, from which it is possible to conclude that this is a suitable problem for post-quantum cryptography. Indeed, we show that it is possible to use MCE to build a zero-knowledge protocol, and

hence a signature scheme, which we name *Matrix Equivalence Digital Signature*, or simply MEDS. For our security analysis, we first study in detail the collision attacks from Chapter XII and then we develop two new attacks. The first attack that we propose uses a nontrivial algebraic modeling inspired from the minors modellings of MinRank in [27, 170]. The second one is an adaptation of Leon's algorithm [256] for matrix codes. Based on this analysis, we provide an initial parameter choice, together with several computational optimizations, resulting in a scheme with great flexibility and competitive data sizes.

A reference implementation for the full scheme is available at

meds-pqc.org

§2 Preliminaries

Chapter I gives a general introduction to signature schemes and how to design these from cryptographic group actions. Chapter III gives an introduction to codes and their isometries. Here, we only introduce a few variants of MCE, and the concept of *ring signatures*.

To simplify notation, we use the following operator in this chapter:

$$\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}) := \mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B}).$$

Multiple-Instance Matrix Code Equivalence.

We introduce a *multiple-instance* version of MCE, which is at the base of one of the optimizations, using *multiple public keys*, which we will describe in Section 4. It is easy to see that this new problem reduces to MCE, as done for instance in [30] for the Hamming case.

Problem 1 (Multiple Matrix Code Equivalence)**.** MIMCE$(k, n, m, r, \mathcal{C}, \mathcal{D}_i)$:
Given: $(r+1)$ equivalent $[m \times n, k]$ matrix codes $\mathcal{C}, \mathcal{D}_1, \ldots, \mathcal{D}_r$.
Goal: Find – if any – $\mathbf{A} \in \mathrm{GL}_m(q)\mathbf{B} \in \mathrm{GL}_n(q)$ such that $\mathcal{D}_i = \mathbf{A}\mathcal{C}\mathbf{B}$ for some $i \in \{1, \ldots, r\}$.

§3 Protocols from Matrix Code Equivalence

MCE yields a promising building block for post-quantum cryptographic schemes. In this section, we obtain a digital signature scheme by instantiating Protocol I.3 with MCE. and then applying the Fiat-Shamir transformation [178].

3.1 THE MCE SIGMA PROTOCOL

The first building block in our work is the Sigma protocol in Protocol 4, in which a Prover proves the knowledge of an isometry $(\mathbf{A}, \mathbf{B})$ between two equivalent matrix codes. The security result is given in Theorem 1. We assume a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$.

Protocol 4. MCE Sigma protocol.

$\mathcal{A}$, knows $(\mathbf{A}, \mathbf{B})$, $(\mathbf{G}_0, \mathbf{G}_1)$		$\mathcal{B}$, knows $(\mathbf{G}_0, \mathbf{G}_1)$
$\tilde{\mathbf{A}}, \tilde{\mathbf{B}} \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$ $\tilde{\mathbf{G}} \leftarrow \mathsf{SF}(\pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}(\mathbf{G}_0))$ $\mathsf{cmt} \leftarrow \mathsf{H}(\tilde{\mathbf{G}})$		
	$\xrightarrow{\mathsf{cmt}}$	
		$\mathsf{ch} \xleftarrow{\$} \{0,1\}$
	$\xleftarrow{\mathsf{ch}}$	
If $\mathsf{ch} = 0$: $\quad \mathsf{resp} \leftarrow (\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$ If $\mathsf{ch} = 1$: $\quad \mathsf{resp} \leftarrow (\tilde{\mathbf{A}}\mathbf{A}^{-1}, \mathbf{B}^{-1}\tilde{\mathbf{B}})$		
	$\xrightarrow{\mathsf{resp}}$	
		If $\mathsf{ch} = 0 : \mathbf{G} \leftarrow \mathbf{G}_0$ If $\mathsf{ch} = 1 : \mathbf{G} \leftarrow \mathbf{G}_1$ $h' = \mathsf{H}(\mathsf{SF}(\pi_{\mathsf{resp}}(\mathbf{G})))$ $\mathsf{cmt} \stackrel{?}{=} h'$

Theorem 1. Protocol 4 is complete, 2-special sound and honest-verifier zero-knowledge assuming the hardness of the MCE problem.

We discuss this result in a more general form in Chapter XIV. By repeating the Sigma protocol for $t = \lambda$ rounds and applying the Fiat-Shamir transform [178], we get a λ-bit secure signature scheme, given in three algorithms Algorithm 30, Algorithm 31, and Algorithm 32.

Public key and signature size.

We begin by calculating the communication costs for Protocol 4. Whenever $c = 0$, the response consists entirely of randomly-generated objects. Thus, it

Algorithm 30 Keygen for the basic MCE signature scheme

Input: A security parameter λ.
Output: A keypair $(\mathsf{sk}, \mathsf{pk})$ with $\mathsf{sk} = (\mathbf{A}, \mathbf{B}) \in \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$ and pk a pair of codes $\mathbf{G}_0, \mathbf{G}_1 \subseteq \mathbb{F}_q^{m \times n}$

1: $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{m \times n \times k}$ until $\mathbf{G}_0$ is k-dimensional
2: $\mathbf{A} \xleftarrow{\$} \mathrm{GL}_m(q)$, $\mathbf{B} \xleftarrow{\$} \mathrm{GL}_n(q)$
3: $\mathbf{G}_1 \leftarrow \mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}_0))$
4: $\mathsf{sk} \leftarrow (\mathbf{A}, \mathbf{B})$, $\mathsf{pk} \leftarrow (\mathbf{G}_0, \mathbf{G}_1)$
5: **return** $(\mathsf{sk}, \mathsf{pk})$

Algorithm 31 Sign for the basic MCE signature scheme

Input: A message msg, a secret key $\mathsf{sk} = (\mathbf{A}, \mathbf{B})$ with $\mathsf{pk} = (\mathbf{G}_0, \mathbf{G}_1)$.
Output: A signature σ for the message msg

1: **for** $i = 0, \ldots, t-1$ **do**
2: $\quad \tilde{\mathbf{A}}_i \xleftarrow{\$} \mathrm{GL}_m(q)$, $\tilde{\mathbf{B}}_i \xleftarrow{\$} \mathrm{GL}_n(q)$
3: $\quad \tilde{\mathbf{G}}_i \leftarrow \mathsf{SF}(\pi_{\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i}(\mathbf{G}_0))$
4: $h \leftarrow \mathsf{H}(\tilde{\mathbf{G}}_0, \ldots, \tilde{G}_{t-1}, \mathsf{msg})$
5: Parse h as $h_0 | \ldots | h_{t-1}$ with $h_i \in \{0,1\}$
6: **for** $i = 0, \ldots, t-1$ **do**
7: $\quad \mu_i \leftarrow \tilde{\mathbf{A}}_i \cdot \mathbf{A}_{h_i}^{-1}$, $\nu_i \leftarrow \mathbf{B}_{h_i}^{-1} \cdot \tilde{\mathbf{B}}_i$
8: $\sigma \leftarrow (h, \mu_0, \ldots, \mu_{t-1}, \nu_0, \ldots, \nu_{t-1})$
9: **return** σ

Algorithm 32 Verif for the basic MCE signature scheme

Input: A signature σ, a message msg and a public-key $\mathsf{pk} = (\mathbf{G}_0, \mathbf{G}_1)$.
Output: Either true (valid signature) or false (invalid signature)

1: Parse h as $h_0 | \ldots | h_{t-1}$ with $h_i \in \{0,1\}$
2: **for** $i = 0, \ldots, t-1$ **do**
3: $\quad \hat{\mathbf{G}}_i \leftarrow \mathsf{SF}(\pi_{\mu_i, \nu_i} \mathbf{G}_{h_i})$
4: $h' = \mathsf{H}(\hat{\mathbf{G}}_0, \ldots, \hat{\mathbf{G}}_{t-1}, \mathsf{msg})$
5: **return** $h \stackrel{?}{=} h'$

can be represented efficiently by a single seed of length λ that generates two random (invertible) matrices, reducing the communication cost. We also add a hash digest of length 2λ to avoid collision attacks. This yields the following cost per round, in bits:

$$\begin{cases} 3\lambda + 1 & \text{if } c = 0 \\ 2\lambda + 1 + (m^2 + n^2)\lceil \log_2(q) \rceil & \text{if } c = 1 \end{cases}$$

For the basic MCE signature scheme, we calculate the sizes as follows. First, since the matrix $\mathbf{G}_0$ is random, it can also be represented via a short seed, and therefore can be included in the public key at negligible cost. As the number of rounds t is equal to desired security level λ, the protocol yields the following sizes (in bits):

- Public key size: $\lambda + k(mn - k)\lceil \log_2(q) \rceil$
- Average signature size: $t \cdot \left(1 + \dfrac{\lambda + (m^2 + n^2)\lceil \log_2(q) \rceil}{2}\right)$.
- Maximum signature size: $t \cdot \left(1 + 2\lambda + (m^2 + n^2)\lceil \log_2(q) \rceil\right)$.

§4 MEDS: Matrix Equivalence Digital Signature

In this section, we apply known optimizations from the literature and specific optimizations for MCE to the basic MCE signature scheme to obtain the full scheme: Matrix Equivalence Digital Signature (MEDS).

4.1 Improving the performance of the basic protocol

Multiple keys.

The first optimization is a popular one in literature [30, 54, 144], and it consists of utilizing multiple public keys, i.e. multiple equivalent codes $\mathbf{G}_0, \ldots, \mathbf{G}_{s-1}$, each defined as $\mathbf{G}_i = \mathsf{SF}(\pi_{\mathbf{A}_i, \mathbf{B}_i}(\mathbf{G}_0))$ for uniformly chosen secret keys[1] $(\mathbf{A}_i, \mathbf{B}_i)$. This allows to reduce the soundness error from $1/2$ to $1/s$. The optimization works by grouping the challenge bits into strings of $\ell = \lceil \log_2 s \rceil$ bits, which can then be interpreted as binary representations of the indices $\{0, \ldots, s-1\}$, thus

1 For convenience, we write $\mathbf{A}_0 = \mathbf{I}_m$, $\mathbf{B}_0 = \mathbf{I}_n$.

dictating which public key will be used in the protocol per round. Security is preserved since the proof of unforgeability can easily be modified to rely on a multi-instance version of the underlying problem, in our case MIMCE (Problem 1), which reduces to the original MCE. Although in the literature s is often chosen as a power of 2, this is not always optimal. In this chapter, we will therefore select the value of s such that we optimize scheme performance and signature size.

Remark 1. This optimization comes at the cost of an s-fold increase in public-key size. As shown for instance in [144], it is possible to reduce this impact by using Merkle trees to a hash of the tree commitment of all the public keys. However, this would add some significant overhead to the signature size, because it would be necessary to include the paths for all openings. Considering the sizes of the objects involved, such an optimization is not advantageous in our case.

Fixed-weight challenges.

Another common optimization is the use of fixed-weight challenges [53], which exploits the fact that response isometries for challenges $h_i = 0$ can be communicated by a single seed of length λ instead of two matrices of size $(m^2 + n^2) \cdot \lceil \log_2 q \rceil$. The idea is to generate the challenge string h with a fixed number zeros, i.e. the Hamming weight of h is fixed, rather than uniformly random. Then, when $h_i = 0$, the response (μ_i, ν_i) consists entirely of randomly-generated objects from a seed σ_i, and so we only transmit the seed σ_i in the signature.

This creates a noticeable imbalance between the two types of responses, and hence it makes sense to minimize the number of non-zero values. To this end, one can utilize a so-called *weight-restricted hash function*, that outputs values in $\mathbb{Z}^t_{s,w}$, by which we denote the set of vectors with elements in $\{0, \ldots, s-1\}$ of length t and weight w, where s denotes the number of public keys. In this way, although the length of the challenge strings increases, the overall communication cost scales down proportionally to the value of w. In terms of security, this optimization only entails a small modification in the statement of the Forking Lemma, and it is enough to choose parameters such that $\log_2 \binom{t}{w} \geq \lambda$. In practice, we achieve a weight-restricted hash functionby using a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$, an expanding the output to a t-tuple $(h_0, \ldots, h_{t-1}) \in \mathbb{Z}^t_{s,w}$ of weight w.

Seed tree.

The signature size can be further improved using a *puncturable PRF* [59] instantiated as a *seed tree* [53, § 2.7]. The idea is to more efficiently communicate the seeds that generate the random matrices in those rounds i with $h_i = 0$ not as $t - w$ individual seeds, but as derived seeds from a single master seed. First, we generate the many seeds used throughout the protocol in a recursive way, starting from a master seed mseed and building a binary tree, via repeated PRNG applications, having t seeds as leaves. When the required $t - w$ values need to be retrieved, it is then enough to reveal the appropriate sequence of nodes. This reduces the space required for the seeds from $\lambda(t - w)$ to λN_{seeds}, where N_{seeds} can be upper bounded by $2^{\lceil \log_2(w) \rceil} + w(\lceil \log_2(t) \rceil - \lceil \log_2(w) \rceil - 1)$, as shown in [204]. As suggested in [53], we are including a 256-bit salt to ward off multi-target collision attacks and the leaf address as identifier for domain separation in the inputs of the seed-tree hash functions.

Partially seeding the public key.

Finally, using multiple public keys comes at a significant size increase to the full public key, so we propose a new optimization that trades public-key size for private-key size. This optimization is not generic for all such Sigma-protocols based on group actions: it requires the specific struture of the underlying MCE group action. This optimization is inspired by the trade-off in the key generation of Rainbow [159] and UOV [52]. It has not been previously used in Fiat-Shamir signatures, but we conjecture similar optimizations may be possible for other cryptographic group actions based on equivalence probems, such as [30, 54, 144]. Instead of generating the secret $(\mathbf{A}_i, \mathbf{B}_i)$ from a secret seed and then deriving the public key $\mathbf{G}_i$, we instead *partially* generate $\mathbf{G}_i^0$ from a public seed and use $\mathbf{G}_i^0$ to find a (secret) isometry $(\mathbf{A}_i, \mathbf{B}_i)$. We then apply $(\mathbf{A}_i, \mathbf{B}_i)$ to $\mathbf{G}_0$ to obtain the full $\mathbf{G}_i$ which becomes the public key. More precisely, we do the following for each public key $\mathbf{G}_i$:

- Select two random codewords $\mathbf{C}_0$ and $\mathbf{C}_1$ in $\mathbf{G}_0$.
- Generate, from a public seed, a complete $m \times n$ codeword $\mathbf{P}_1$ and the top $m - 1$ rows of codeword $\mathbf{P}_2$ (depending on the parameters m, n one can get slightly more rows when $m \neq n$).

- Find $\mathbf{A}$ and $\mathbf{B}$ by solving the linear system

$$\mathbf{P}_1\mathbf{B}^{-1} = \mathbf{A}\mathbf{C}_0,$$
$$\mathbf{P}_2\mathbf{B}^{-1} = \mathbf{A}\mathbf{C}_1,$$

  by fixing the first (top left) value of $\mathbf{A}$ and set $\mathbf{A}_i, \mathbf{B}_i \leftarrow \mathbf{A}, \mathbf{B}$ as the i-th psecret key.

- Apply $(\mathbf{A}_i, \mathbf{B}_i)$ to $\mathbf{G}_0$ to get the public key $\mathbf{G}_i$.

Therefore, to communicate the i-th public key, we use the public seed to generate $\mathbf{P}_1, \mathbf{P}_2$ and only require a full matrix description for the remianing $k-2$ basis vectors of $\mathbf{G}_i$. For verification, the complete $\mathbf{G}_i$ are reconstructed using the seed.

4.2 THE FULL SCHEME: MEDS

To give a complete picture, we present the MEDS signature scheme, including the optimisations from Section 4.1[2], by its three core algorithms: KeyGen, Sign, and Verif. The various parameters control different optimizations: s refers to the number of public keys used, w refers to the fixed weight of the challenge hash string, and t refers to the numbers of rounds required for λ-bit security. Concrete parameter choices will be discussed in Section 6.2. We assume fixed matrices $\mathbf{P}_0, \mathbf{P}_1 \in \mathbb{F}_q^{m\times n}$. For simplicity, we write $\mathbf{A}_0 := \mathbf{I}_m$ and $\mathbf{B}_0 := \mathbf{I}_n$.

Public key and signature size.

With these various optimizations, we obtain the following public key and signature size for MEDS:

- MEDS public-key size: $\lambda + (s-1)((k-2)(mn-k)+n)\lceil \log_2(q)\rceil$
- MEDS signature size:

$$\underbrace{\lambda}_{h} + w \cdot \underbrace{(m^2+n^2)\lceil \log_2(q)\rceil}_{\{\mu_i,\nu_i\} \text{ when } h_i=1} + \underbrace{\lambda N_{\text{seeds}}}_{\{\mu_i,\nu_i\} \text{ when } h_i=0} + \underbrace{2\lambda}_{\text{salt}}$$

2 For readability, we do not add the full details of the seed tree optimisation in the Sign and Verif algorithms.

Algorithm 33 KeyGen for the MEDS signature scheme

Input: A security parameter λ.
Output: A keypair $(\mathsf{sk}, \mathsf{pk})$ with $\mathsf{sk} = (\mathbf{A}_i, \mathbf{B}_i)_{i=0}^{s-1}$ and pk an s-tuple of codes $(\mathbf{G}_0, \ldots, \mathbf{G}_{s-1})$

$\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{m \times n \times k}$ until $\mathbf{G}_0$ is k-dimensional
for $i = 1, \ldots, s-1$ **do**
  $\mathbf{C}_0, \mathbf{C}_1 \xleftarrow{\$} \mathbf{G}_0$
  Solve $(\mathbf{A}_i, \mathbf{B}_i) \in \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$ from
    $\mathbf{P}_0 \mathbf{B}_i^{-1} = \mathbf{A}_i \mathbf{C}_0$ and $\mathbf{P}_1 \mathbf{B}_i^{-1} = \mathbf{A}_i \mathbf{C}_1$
  $\mathbf{G}_i \leftarrow \mathsf{SF}(\pi_{\mathbf{A}_i, \mathbf{B}_i}(\mathbf{G}_0))$
$\mathsf{sk} \leftarrow (\mathbf{A}_1, \mathbf{B}_1, \ldots, \mathbf{A}_{s-1}, \mathbf{B}_{s-1})$, $\mathsf{pk} \leftarrow (\mathbf{G}_0, \mathbf{G}_1, \ldots, \mathbf{G}_{s-1})$
return $(\mathsf{sk}, \mathsf{pk})$

Algorithm 34 Sign for the MEDS signature scheme

Input: A message msg, a secret key $\mathsf{sk} = (\mathbf{A}_i, \mathbf{B}_i)_{i=1}^{s-1}$, and $\mathsf{pk} = (\mathbf{G}_i)_{i=0}^{s-1}$.
Output: A signature σ for the message msg

for $i = 0, \ldots, t-1$ **do**
  $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \leftarrow \mathsf{ExpandSeed}(\mathsf{seed}_i)$
  $\tilde{\mathbf{G}}_i \leftarrow \mathsf{SF}(\pi_{\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i}(\mathbf{G}_0))$
$h \leftarrow \mathsf{H}(\tilde{\mathbf{G}}_0, \ldots, \tilde{G}_{t-1}, \mathsf{msg})$
Expand h to $h_0 | \ldots | h_{t-1}$ with $h_i \in \{0, 1, \ldots, s-1\}$
for $i = 0, \ldots, t-1$ **do**
  if $h_i = 0$ **then** $\mu_i, \nu_i \leftarrow \mathsf{seed}_i$
  if $h_i > 0$ **then** $\mu_i \leftarrow \tilde{\mathbf{A}}_i \cdot \mathbf{A}_{h_i}^{-1}$, $\nu_i \leftarrow \mathbf{B}_{h_i}^{-1} \cdot \tilde{\mathbf{B}}_i$
$\sigma \leftarrow (h, \mu_0, \ldots, \mu_{t-1}, \nu_0, \ldots, \nu_{t-1})$
return σ

Algorithm 35 Verif for the MEDS signature scheme

Input: A signature σ, a message msg and $\mathsf{pk} = (\mathbf{G}_0, \ldots, \mathbf{G}_{s-1})$.
Output: Either true (valid signature) or false (invalid signature)

Expand h to $h_0 | \ldots | h_{t-1}$ with $h_i \in \{0, 1, \ldots, s-1\}$
for $i = 0, \ldots, t-1$ **do**
  $\hat{\mathbf{G}}_i \leftarrow \mathsf{SF}(\pi_{\mu_i, \nu_i}(\mathbf{G}_{h_i}))$ ▷ Expand seed_i if $h_i = 0$.
$h' = \mathsf{H}(\hat{\mathbf{G}}_0, \ldots, \hat{\mathbf{G}}_{t-1}, \mathsf{msg})$
return $h \stackrel{?}{=} h'$

§5 Concrete Security Analysis

In this section, we expand the security analysis on MCE from Chapter XII. Recall that, from that chapter, we have a birthday-based algorithm for solving MCE with complexity $\tilde{O}(q^m)$, with $m = \min\{m, n, k\}$. Here, we furthermore give two algebraic attacks and an attack based on Leon's algorithm [256].

5.1 An algebraic attack using direct modelling

We have shown in Theorem XII.2 and Theorem XII.3 that MCE is equivalent to BMLE, which means that any solver for BMLE can be easily adapted to solve MCE. One of the natural attack avenues is thus to model the problem as an algebraic system of polynomial equations over a finite field. This approach was taken in [174], where the general Isomorphism of Polynomials (IP) problem was investigated. Here, we focus specifically on BMLE and perform a detailed complexity analysis.

First, fix arbitrary bases $(\mathbf{C}^{(1)}, \dots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \dots, \mathbf{D}^{(k)})$ of the codes $\mathcal{C}$ and $\mathcal{D}$ respectively. In terms of the bases, the MCE problem can be rephrased as finding $\mathbf{A} \in \mathrm{GL}_m(q)$, $\mathbf{B} \in \mathrm{GL}_n(q)$ and $\mathbf{T} = (t_{ij}) \in \mathrm{GL}_k(q)$ such that, for all $1 \leq r \leq k$ we have

$$\sum_{1 \leqslant s \leqslant k} t_{rs} \mathbf{D}^{(s)} = \mathbf{A} \mathbf{C}^{(r)} \mathbf{B} \,. \tag{43}$$

The system (43) consists of knm equations in the $m^2 + n^2 + k^2$ unknown coefficients of the matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{T}$. The quadratic terms of the equations are always of the form $\gamma a_{ij} b_{i'j'}$ for some coefficients a_{ij} and $b_{i'j'}$ of $\mathbf{A}$ and $\mathbf{B}$ respectively which means the system (43) is bilinear. Note that the coefficients of $\mathbf{T}$ appear only linearly. As previously, we can guess the m^2 variables from $\mathbf{A}$, which will lead us to a linear system that can be easily solved. However, we can do better by exploiting the structure of the equations.

For ease of readability, we denote the i-th row of a matrix $\mathbf{M}$ by $\mathbf{M}_{i,}$, and we denote the j-th column by $\mathbf{M}_{,j}$, for the remainder of this section. Note that, in (43), for $i \neq j$, the unknown coefficients from two rows $\mathbf{A}_{i,}$ and $\mathbf{A}_{j,}$ don't appear in the same equation and, symmetrically, the same holds for $\mathbf{B}_{,i}$ and $\mathbf{B}_{,j}$. We will make use of this only for the matrix $\mathbf{A}$, that is, we consider only part of the system, and control the number of variables from $\mathbf{A}$. The goal is to reduce the number of variables that we need to guess before obtaining an overdetermined linear system, and we want to do this in an optimal way. Consider the first α

rows from $\mathbf{A}$. Extracting the equations that correspond to these rows in (43) leads us to the system:

$$\sum_{1 \leqslant s \leqslant k} t_{rs} \mathbf{D}_{i\_}^{(s)} = \mathbf{A}_{i\_} \mathbf{C}^{(r)} \mathbf{B}, \quad \text{for all } 1 \leqslant r \leqslant k,\ 1 \leqslant i \leqslant \alpha. \tag{44}$$

Guessing the αm coefficients from $\mathbf{A}_{i\_}$ for all $1 \leqslant i \leqslant \alpha$ leads to a linear system of αkn equations in $n^2 + k^2$ variables. Choosing $\alpha = \lceil \frac{n^2+k^2}{kn} \rceil$, the complexity of the approach becomes $\mathcal{O}(q^{m\lceil \frac{n^2+k^2}{kn} \rceil}(n^2+k^2)^3)$. For the choice of $m = n = k$, a reasonable assumption, this reduces to at least $\alpha = 2$ and a complexity of $\mathcal{O}(q^{2n} n^6)$.

Note that, one can solve the bilinear system (44) directly using for example XL [135] and the analysis for bilinear systems from [308], and similar results can be obtained from [169]. We have verified, however, that due to the large number of variables compared to the available equations, the complexity greatly surpasses our other attacks.

5.2 AN ALGEBRAIC ATTACK USING IMPROVED MODELLING

In order to improve upon the algebraic attack using direct modelling, we improve the model and completely avoid the t_{rs} variables. This modelling is in the spirit of the minors modellings of MinRank [27, 170].

As previously, let $\mathbf{G}$ and $\mathbf{G}'$ be the $k \times mn$ generator matrices of the equivalent codes $\mathcal{C}$ and $\mathcal{D}$ respectively. Then, for the unknown invertible matrices $\mathbf{A}$ and $\mathbf{B}$, using Lemma III.6, $\tilde{\mathbf{G}} = \mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B})$ is also a generator matrix of $\mathcal{D}$. We take the coefficients of $\mathbf{A}$ and $\mathbf{B}$ to be our unknowns. A crucial observation for this attack is that each row $\tilde{\mathbf{G}}_{i\_}$ of $\tilde{\mathbf{G}}$ is in the span of the rows of $\mathbf{G}'$, since $\mathbf{G}'$ and $\tilde{\mathbf{G}}$ define the same code. This means that adding $\tilde{\mathbf{G}}_{i\_}$ to $\mathbf{G}'$ does not change the code, i.e.,

$${}^{(i)}\mathbf{G}' = \begin{pmatrix} \mathbf{G}' \\ \tilde{\mathbf{G}}_{i\_} \end{pmatrix}$$

is not of full rank. From here, all maximal minors $\left| \left({}^{(i)}\mathbf{G}'_{\_j_1}\ {}^{(i)}\mathbf{G}'_{\_j_2} \dots {}^{(i)}\mathbf{G}'_{\_j_{k+1}} \right) \right|$ of ${}^{(i)}\mathbf{G}'$, for every $\{j_1, j_2, \dots, j_{k+1}\} \subset \{1, 2, \dots, mn\}$, are zero.

Now, as in a minors modeling of MinRank, we can form equations in the unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$ by equating all maximal minors to zero, which amounts to a total of $\binom{mn}{k+1}$ equations. Since the unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$ appear only in the last row of the minors, and only bilinearly, the

whole system is also bilinear. Thus we have reduced the problem to solving the bilinear system

$$| \left({}^{(i)}\mathbf{G}'_{\cdot j_1} \; {}^{(i)}\mathbf{G}'_{\cdot j_2} \ldots {}^{(i)}\mathbf{G}'_{\cdot j_{k+1}} \right) | = 0, \quad \begin{array}{l} \text{for all } i \in \{1, 2, \ldots, k\} \text{ and all} \\ \{j_1, j_2, \ldots, j_{k+1}\} \subset \{1, 2, \ldots, mn\} \end{array} \tag{45}$$

in the $m^2 + n^2$ unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$.

At first sight, (45) seems to have more than enough equations to fully linearize the system. However, the majority of these equations are linearly dependent. In fact, there are only $(mn - k)k$ linearly independent equations. To see this, fix some i and consider a minor $| \left({}^{(i)}\mathbf{G}'_{\cdot j_1} \; {}^{(i)}\mathbf{G}'_{\cdot j_2} \ldots {}^{(i)}\mathbf{G}'_{\cdot j_{k+1}} \right) |$ of ${}^{(i)}\mathbf{G}'$. Since all rows except the first do not contain any variables, the equation

$$| \left({}^{(i)}\mathbf{G}'_{\cdot j_1} \; {}^{(i)}\mathbf{G}'_{\cdot j_2} \ldots {}^{(i)}\mathbf{G}'_{\cdot j_{k+1}} \right) | = 0$$

defines the linear dependence between the columns ${}^{(i)}\mathbf{G}'_{\cdot j_1}, \ldots {}^{(i)}\mathbf{G}'_{\cdot j_{k+1}}$. But the rank of the matrix is k, so all columns can be expressed through some set of k independent columns. Thus, in total, for a fixed i we have $mn - k$ independent equations and in total $(mn - k)k$ equations for all i.

Alternatively, we can obtain the same amount of equations from $\tilde{\mathbf{G}}$ and the generator matrix $\mathbf{G}'^{\perp}$ of the dual code of $\mathcal{D}$. Since $\tilde{\mathbf{G}}$ should also be a generator matrix of $\mathcal{D}$, we construct the system

$$\mathbf{G}'^{\perp} \cdot \tilde{\mathbf{G}}^{\top} = \mathbf{0},$$

which again has $(mn - k)k$ bilinear equations in $n^2 + m^2$ variables.

The complexity of solving the obtained system using either of the modellings strongly depends on the dimension of the code – it is the smallest for $k = mn/2$, and grows as k reduces (dually, as k grows towards mn). In Section 6, we give the concrete complexity estimate for solving the system for the chosen parameters using bilinear XL and the analysis from [308].

The attack does not seem to benefit a lot from being run on a quantum computer, as the costly part comes from solving a large linear system for which there are no useful quantum algorithms available. The only viable approach seems to be to 'Groverize' an enumeration part of the algorithm: One could enumerate over one set of the variables, either of $\mathbf{A}$ or $\mathbf{B}$, typically the smaller one, and solve a biliner system of less variables. Grover's algorithm could then speed up this enumeration quadratically. However, in the classical case the best approach is *not* to use enumeration, so this approach only makes sense for quite

small values of the field size i.e. only when $q < 4$. In this parameter regime, however, combinatorial attacks perform significantly better, so this approach becomes irrelevant.

5.3 LEON-LIKE ALGORITHM ADAPTED TO THE RANK METRIC

Leon [256] proposed an algorithm to solve the permutation code equivalence problem in the Hamming metric, which relies on the basic property that the weight distribution of two equivalent codes is the same as isometries preserve the weight of codewords, by definition. Thus, finding the set of codewords of minimal weight in both codes reveals enough information to find a permutation that maps one set to the other. With high probability, this permutation is the unknown isometry between the codes. This algorithm is quite unbalanced and heavy on the 'codewords finding' side, since it requires finding all codewords of minimal weight. Beullens' algorithm [51] relaxes this constraint by performing a collision-based algorithm instead, much in the spirit of Algorithm 28. First, the algorithm builds, per code, a list of codewords of a particular weight, including their multiset of entries. Then, it tries to find a collision between these lists using the multisets. Given a collision, it employs an efficient subroutine for reconstructing the isometry.

The approach from the Hamming metric can be translated to matrix codes and can be used to solve MCE, with some necessary adjustments. First of all, finding codewords of a given rank r is equivalent to an instance of MinRank [136, 170] for k matrices of size $m \times n$ over $\mathbb{F}_q$. Depending on the parameters, we have noticed that the Kipnis-Shamir modelling [237] and Bardet's modelling [27] perform the best, so we use both in our complexity estimates.

For the collision part, we require a collision between *pairs* of codewords $(\mathbf{C}_1, \mathbf{C}_2)$ and $(\mathbf{D}_1, \mathbf{D}_2)$: given only a collision $\mathbf{C}_1$ from $\mathcal{C}$ and $\mathbf{D}_1$ from $\mathcal{D}$, it is not possible to determine the isometry $(\mathbf{A}, \mathbf{B})$, as there are many isometries possible between single codewords. Thus, there is no efficient way of checking that these codewords collide nor finding the correct isometry. On the other hand, a pair of codewords is typically enough. For the pairs $(\mathbf{C}_1, \mathbf{C}_2)$ and $(\mathbf{D}_1, \mathbf{D}_2)$ we can form the system of $2mn$ linear equations

$$\begin{cases} \mathbf{A}^{-1}\mathbf{D}_1 = \mathbf{C}_1\mathbf{B} \\ \mathbf{A}^{-1}\mathbf{D}_2 = \mathbf{C}_2\mathbf{B} \end{cases} \tag{46}$$

in the $m^2 + n^2$ unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$. When $m = n$, which is a typical choice, the system is expected to be overdetermined, and thus solved[3] in $\mathcal{O}(n^6)$. In practice, and since $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{D}_1$ and $\mathbf{D}_2$ are low-rank codewords, there are fewer than $2n^2$ linearly independent equations, so instead of a unique solution, we can obtain a basis of the solution space. However, the dimension of the solution space is small enough so that coupling this technique with one of the algebraic modelings results in a system that can be solved through direct linearization. It is then easy to check whether the obtained isometry maps $\mathcal{C}$ to $\mathcal{D}$. We will thus assume, as a bound for the complexity of this attack, that we need to find collisions between pairs of codewords.

We let $C(r)$ denote the number of codewords of rank r in a k-dimensional $m \times n$ matrix code. Using a birthday argument, two lists of size $L = \sqrt{2C(r)}$ of rank r codewords of $\mathcal{C}$ and $\mathcal{D}$ are enough to find two collisions. To detect the two collisions, we need to generate and solve systems as in Equation (46) for all possible pairs of elements from the respective lists, so $\binom{L}{2}^2$ systems in total. We have $C(r) \approx q^{r(n+m-r)-nm+k}$ [251], so the total complexity amounts to

$$\mathcal{O}(q^{2(r(n+m-r)-nm+k)}(m^2+n^2)^\omega).$$

A deterministic variant of this approach has the same asymptotic complexity: we choose two rank-r codewords of $\mathcal{C}$ and check these for a 2-collision against all pairs of rank-r codewords of $\mathcal{D}$, which requires solving $\binom{C(r)}{2}$ systems.

Finally, we choose r so that both parts, the MinRank and the collision part, are as closely balanced as possible. Section 6 discusses the complexity of this approach for the chosen parameters of our scheme.

When considering the quantum version of the algorithm, we obtain a quadratic speedup in the collision part using Grover's algorithm [203]. Because hybridization is also possible for the MinRank part, it can also benefit from using Grover, especially for larger fields.

§6 Implementation and Evaluation

In this section, we give an assessment of the performance of MEDS. We begin with important considerations about the implementation of the signature scheme. Then, we provide concrete parameter choices for MEDS and a first pre-

3 The optimization to reduce the public-key size in Section 4.1 works in roughly the same way.

liminary evaluation of its performance based on a C reference implementation. The source code of our implementation is available at

https://github.com/MEDSpqc/meds.

6.1 IMPLEMENTATION

Besides performance, the security of a cryptographic implementation is of crucial importance. By "implementation security" here we mean the resilience of an implementation against physical attacks such as timing attacks, power-analysis attacks, and fault-injection attacks. While the requirement for side-channel and fault-injection attacks heavily depends on whether physical access is possible for an attacker, it is widely considered best practice to provide protection against timing attacks as baseline for all cryptographic implementations. In order to do this, the implementation must be *constant time*, i.e., timing variations based on secret data must be prevented. This typically means that branching and memory access based on secret data must be avoided.

There are generic constructions to achieve timing security for basically any given cryptographic scheme. However, if the design of a cryptographic primitive does not take constant-time requirements into consideration, such generic constructions can be computationally expensive. Therefore, designing a cryptographic scheme such that it supports an efficient protection against timing attacks can make it easier to implement the scheme securely. In turn, this makes a scheme more secure and efficient in practice.

In the case of MEDS, we need to consider the implementation security of key generation and signing. Verification only operates on public data and hence does not require further protection. The basic operations of MEDS during key generation and signing are:

- field arithmetic,
- matrix multiplication,
- generating random invertible matrices, and
- computing a canonical form of a matrix.

All these operations must be implemented securely, and thus, we discuss these one by one.

Field arithmetic.

We are using a relatively small prime field in MEDS. Hence, for finite field addition and multiplication, we can simply perform integer arithmetic followed by a reduction modulo the prime. On most architectures, constant-time operations for adding and multiplying small operands are available. The reduction modulo the prime can be implemented using standard approaches in literature, e.g., by Granlund and Montgomery [201], Barrett [33], or Montgomery [281]. Inversion of field elements can efficiently be implemented in constant time using Fermat's little theorem and optimal addition chains.

In the C reference implementation, we simply use the modulo operation for reduction and Fermat's little theorem for inversion to get a first impression on the performance of the scheme. For using MEDS in practice, the timing side-channel security of the modulo operation in particular needs to be verified.

Matrix multiplication.

The basic schoolbook algorithm for multiplying matrices is not data dependent and hence constant time. More sophisticated approaches like the method by Arlazarov, Dinic, Kronrod, and Faradžev [14] may be used on secret data, but most likely do not significantly improve the performance for the relatively small finite field and the matrices used in MEDS, and neither are other asymptotically more efficient matrix-multiplication algorithms more efficient for the given matrix dimensions. Hence, for the C reference implementation, we are simply using schoolbook multiplication. If a more efficient algorithm is used for optimized implementations, it must be ensured that a constant-time algorithm is used.

Generating random invertible matrices.

In MEDS, we need to generate random invertible matrices from seeds to generate potentially secret matrices $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \in \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$, which we need to recompute in verification whenever $h_i = 0$. We list two approaches for generating random invertible matrices:

1. *Trial-and-error approach:* Generate a random matrix $\mathbf{M}$ and attempt to compute its inverse. If $\mathbf{M}$ does not have an inverse, try again with a new random matrix. This approach requires constant-time Gaussian

elimination as $\mathbf{M}$ needs to be kept secret and might require several attempts[4] before a random invertible matrix has been found.

2. *Constructive approach:* Construct a matrix $\mathbf{M}$ that is ensured to be inveritble. One approach for this is described by Randall [319]: Generate a random lower-left triangular matrix $\mathbf{L}$ with 1s on the diagonal and an upper-right triangular matrix $\mathbf{U}$ with no zeroes on the diagonal, as well as a permutation matrix $\mathbf{P}$, similar to the result of an LUP decomposition. Then compute the random invertible matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{P}^{-1}\mathbf{L}\mathbf{U}$.

   Both generation of and multiplication with $\mathbf{P}$ are expensive to implement in constant time. Therefore, we can follow the approach of [362]: we leave out $\mathbf{P}$, and compute $\mathbf{M}$ as $\mathbf{M} = \mathbf{L}\mathbf{U}$ directly. However, this covers only a subset of size $((q-1)/q)^n$ of all invertibe matrices in $\mathbb{F}_q^{n\times n}$.

The fastest approach in our experiments is the constructive approach $\mathbf{M} = \mathbf{L}\mathbf{U}$ following [362], which we thus use in all cases.

Computing a canonical form for a matrix.

MEDS requires to compute a canonical form of matrices before hashing. However, during signing, this must be computed in constant time. Computing the reduced row-echelon form of a matrix in constant time is expensive, but other canonical matrix forms can be computed more efficiently in constant time, such as the *systematic form* and the *semi-systematic* form of a matrix, used, e.g., in Classic McEliece [8]. Both these forms are special cases of the reduced row-echelon form.

For the systematic form, all pivoting elements are required to reside on the left diagonal of the matrix, i.e., a matrix $\mathbf{G} \in \mathbb{F}_q^{k\times mn}$ in systematic form has the shape $(\mathbf{I}_k|\mathbf{G}')$ with $\mathbf{G}' \in \mathbb{F}_q^{k\times(mn-k)}$ and where $\mathbf{I}_k$ denotes the $k \times k$ identity matrix. The requirements for the semi-systematic form are more relaxed: Following [8, Sec. 2.2.1], we say that a matrix $\mathbf{G}$ is in (μ, ν)-semi-systematic form if $\mathbf{G}$ has r rows (i.e., no zero rows), the pivot element in row $i \leq r - \mu$ also is in column i and the pivot element in row $i > r - \mu$ is in a column $c \leq i - \mu + \nu$.

However, not all matrices admit a systematic or semi-systematic form. In such cases, we need to restart the computation with new random data. The probability that a matrix $\mathbf{G} \in \mathbb{F}_q^{k\times mn}$, $k \leq mn$ is full rank is $\prod_{i=1}^{k}(q^{mn} - q^{i-1})/q^{kmn}$. Therefore, the probability that $\mathbf{G}$ has a systematic form is $\prod_{i=1}^{k}(q^k -$

4 Although over fields of larger characteristic, almost any matrix is invertible.

$q^{i-1})/q^{k^2}$ and the probability that it has a semi-systematic form is $\prod_{i=1}^{k-\mu}(q^k - q^{i-1})/q^{(k-\mu)k} \cdot \prod_{i=1}^{\mu}(q^\nu - q^{i-1})/q^{\mu\nu}$. The probability and the cost of constant-time implementation for the semi-systematic form depend on μ and ν.

In total, we have the following three options to avoid computing a reduced row-echelon form in constant time for $\tilde{\mathbf{G}}_i$ during signing:

1. *Basis change:* After computing $\tilde{\mathbf{G}}'_i = \pi_{\tilde{\mathbf{A}}_i,\tilde{\mathbf{B}}_i}(\mathbf{G}_0)$, perform a basis change $\tilde{\mathbf{G}}''_i = \mathbf{M}_i\tilde{\mathbf{G}}'_i$ with a secret random invertible matrix $\mathbf{M}_i \in \mathrm{GL}_k(q)$. Then, compute a canonical form $\tilde{\mathbf{G}}_i = \mathsf{SF}(\tilde{\mathbf{G}}''_i)$ on a public $\tilde{\mathbf{G}}''_i$ without the need for a constant-time implementation. This removes the requirement of a constant-time computation of the canonical form but introduces extra cost for the generation of and multiplication with a random invertible matrix $\mathbf{M}_i$. Instead of an invertible matrix $\mathbf{M}_i$, we may use just a random matrix as the probability that this matrix is not invertible is low. Still, if this happens, the computation of the canonical form fails. In that case, the process needs to be restarted with a different $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$.

2. *Semi-systematic form:* Compute the semi-systematic form. This requires a less expensive constant-time implementation than for the reduced-row-echelon form. However, computing the canonical form might fail if no semi-systematic form exists, in which case the computation needs to be restarted.

3. *Systematic form:* Compute the systematic form. This can be implemented even more easily and cheaper in constant time than computing the semi-systematic form. However, systemization failure is more frequent and it is more likely that computations need to be restarted.

We performed generic experiments to investigate the performance impact of these variants. Our experiments indicate that the implementation cost for variant 1 is higher than that of the other two. For the specific parameter sets we propose in Section 6.2, the probability that $\tilde{\mathbf{G}}_i$ does not have a systematic form is about 0.0015%. Therefore, even though the failure probability can be reduced by computing the semi-systematic form compared to the systematic form with well-chosen μ and ν, the overall overhead of computing the semi-systematic form is likely higher than the overall overhead of computing the systematic form. Hence, we use the systematic form throughout as canonical form.

$\lceil \log_2 q \rceil$	$m = n = k$	Birthday	Algebraic	Leon	Sig.
9	16	235.29	181.55	131.20	13 296
9	17	249.04	194.55	149.65	16 237
10	15	244.62	174.75	130.50	12 428
11	14	250.79	160.24	131.21	12 519
12	14	272.40	160.24	141.17	13 548
13	**13**	**274.10**	**146.76**	**130.41**	**11586**
14	13	294.10	146.76	134.41	13 632
20	12	383.75	138.46	135.40	16 320

Table 18: Cost of the investigated attacks in log scale, and 'Sig.' for 'signature size in bytes. Preferred choice in bold.

6.2 PARAMETER CHOICE AND EVALUATION

A summary of the cost of the three different attacks described in Section 5 is given in Table 18. First, we decide to set $m = n = k$, as this seems to be the Goldilocks zone for our scheme. For k larger, the algebraic attack becomes significantly better, and the same is true for Leon's attack when k is smaller. Then, for finite fields of different sizes, we find the smallest value of n that achieves the required security level of 128 bits[5]. We see that Leon's algorithm performs the best in most cases, although the algebraic approach is almost as good. Finally, to determine the optimal value for q, we choose the optimization parameters (s, t, and w) such that the sizes of the public key and the signature are comparable, and we report the signature size in the last column of Table 18. We conclude that the sweet spot for 128-bit security is given for the 13-bit prime $q = 8191$ and $m = n = k = 13$.

Remark 2. Given these parameters, we heuristically assume that the automorphism group of the codes is trivial with overwhelming probability. It is computationally infeasible to compute the automorphism group of codes of this size; however, data on smaller-sized codes shows that the probability of a random code having a trivial automorphism group grows rapidly as q, n, and m increase.

5 At the moment of writing this thesis, cryptanalysis on MCE has significantly improved. I have left these numbers 'as is' to reflect our understanding at the time this chapter was published [110].

6 https://bench.cr.yp.to/supercop.html

In this setting, we can vary s, t, and w for different trade-offs of public-key and signature sizes, as well as performance. We also check the impact of q if we aim for small public keys or small signatures, instead if balancing these two as in Table 18. In such cases, both 11-bit and 13-bit primes for q seem to perform similarly well. Hence, we stick to the 13-bit prime $q = 8191$ in our discussion.

Table 19 provides an overview of 128-bit security parameters for MEDS, highlighting different performance and key/signature size trade-offs. The best attack for all parameter sets based on $q = 8191$, $m = n = 13$, and $k = 13$ is the Leon-like attack as shown in Table 18 with an expected cost of slightly over 2^{130} operations. The best quantum attack is obtained by Groverizing Leon's algorithm and has a cost of around 2^{88} operations. We select s, t, and w such that the probability of an attack on the Fiat-Shamir construction is around 2^{-128}. To improve the efficiency of vectorized implementations using SIMD instructions in the future, we select t as a multiple of 16. In general, we are using all optimizations discussed in Section 4. However, we provide one parameter set without using the seed tree (without '-st' in the name of the parameter set).

Table 20 shows the resulting performance of these parameter sets from our constant-time C reference implementation on an AMD Ryzen 7 PRO 5850U CPU. The C reference implementation follows the implementation discussion above but does not apply any further algorithmic or platform-specific optimizations. We expect that optimized and vectorized implementations can significantly increase the performance.

The parameter set MEDS-2826-st with $s = 2$ provides the smallest public key of about 2.8 kilobytes and a signature of about 18 kilobytes. MEDS-8445-st increases the public key size with $s = 4$ to slightly over 8 kilobytes while reducing the signature size to about 10.5 kilobytes. MEDS-8445-st-f is a 'fast' variant of this parameter set with a smaller $t = 160$ but a larger $w = 23$, resulting in a larger signature size of about 14 kilobytes. MEDS-8445-st-s is 'small' and goes the opposite direction, providing a smaller signature size of about 8.5 kilobytes due to a smaller $w = 13$ at a larger computational cost due to $t = 1760$. These three sibling parameter sets illustrate the impact of t and w on performance and signature size.

MEDS-11255-st provides balanced public key and signature sizes, with both around 11 kilobytes, and a small sum of signature and public-key size at moderate computational cost for signing and verification due to $t = 224$. Removing the seed tree optimization comes with an increase in signature size of about 2 kilobytes, which illustrates the impact of the seed tree.

Finally, sets MEDS-42161-st, MEDS-356839-st, and MEDS-716471-st push the public key size to an extreme at the expense of key generation time in the pursue of reducing signature size and computational cost for signing and verification. However, we expect that at least the key generation time can significantly be improved by optimizing the computation of solving the medium-size sparse linear system used for partially seeding the public key.

Overall, Table 19 and Table 20 highlight the large degree of flexibility offered by the MEDS scheme. All parameter sets are competitive with respect to existing literature, such as the LESS-FM scheme.

6.3 COMPARISON TO RELATED SIGNATURE SCHEMES

Table 21 shows a comparison of public key and signature sizes as well as computational performance of our new MEDS scheme with some established schemes and related recent proposals. While the comparison of public key and signature sizes is accurate, the performance comparison needs to be taken with a large grain of salt: While we provide numbers in the same metric (mega cycles – mcyc.), a direct comparison is difficult as not all schemes have received the same degree of optimization and not all data has been obtained on the same CPU architecture.

The performance data from the 'classical' scheme ed25519 as well as from the NIST PQC schemes CRYSTALS-Dilithium [164], Falcon [180], and SPHINCS+ [211] has been obtained directly from the SUPERCOP website[7]. We selected the performance data from the AMD64 Zen CPU, which is an AMD Ryzen 7 1700 from 2017, i.e., the same microarchitecture (but a different CPU) as we used for our measurements of MEDS.

For UOV [52], LESS [30] and Wavelet [21] we list the performance data as reported in the respective papers unless such data was unavailable. In the case of SDitH [5], only reports of performance data in milliseconds on a 3.1 GHz Intel Core i9-9990K are available. We computed the corresponding number of cycles from this to enable a rough comparison to the other schemes, but note that this data is therefore not entirely accurate.

Table 21 shows that, although code-based schemes do not compete well with pre-quantum or lattice-based PQC schemes, MEDS fills a gap for multivariate or code-based schemes, with a relatively small combined size of public key and signature. Furthermore, its versatility in parameter selection allows for

7 https://bench.cr.yp.to/results-sign.html – amd64; Zen (80of11); 2017 AMD Ryzen 7 1700; 8 x 3000 MHz; rumba7, supercop-20220506

great flexibility for specific applications. In terms of performance, the current implementation of MEDS is still unoptimized. We expect speed-ups of at least one order of magnitude from SIMD parallelization on AVX256 and AVX512 CPUs, since both the data-independent loop of the Fiat-Shamir construction and the matrix arithmetic lend themselves to efficient parallelization. An optimized implementation of MEDS for modern SIMD architectures or embedded systems is left as future work.

Parameter Set	q	n	m	k	s	t	w	ST	PK	Sig.	FS
MEDS-2826-st	8191	13	13	13	2	256	30	✓	2826	18 020	−129.74
MEDS-8445-st-f	8191	13	13	13	4	160	23	✓	8445	13 946	−128.01
MEDS-8445-st	8191	13	13	13	4	464	17	✓	8445	10 726	−128.76
MEDS-8445-st-s	8191	13	13	13	4	1760	13	✓	8445	8702	−128.16
MEDS-11255-st	8191	13	13	13	5	224	19	✓	11 255	11 618	−128.45
MEDS-11255	8191	13	13	13	5	224	19	–	11 255	13 778	−128.45
MEDS-42161-st	8191	13	13	13	16	128	16	✓	42 161	9616	−128.85
MEDS-356839-st	8191	13	13	13	128	80	12	✓	356 839	7288	−129.64
MEDS-716471-st	8191	13	13	13	256	64	11	✓	716 471	6530	−127.37

Table 19: Parameters for MEDS, for $\lambda = 128$ bits of classical security. 'ST' for seed tree. 'PK' for 'public-key size' and 'Sig.' for 'signature size in bytes, 'FS' for 'Fiat-Shamir' probability logarithmic to base 2.

Parameter Set	Key Generation		Signing		Verification	
	(ms)	(mcyc.)	(ms)	(mcyc.)	(ms)	(mcyc.)
MEDS-2826-st	71.13	135.14	102.79	195.30	98.00	186.21
MEDS-8445-st-f	211.45	401.75	63.21	120.09	60.14	114.27
MEDS-8445-st	211.35	401.57	185.68	352.79	178.42	339.01
MEDS-8445-st-s	211.77	402.36	697.00	1324.31	673.19	1279.05
MEDS-11255-st	258.18	490.54	88.12	167.44	84.47	160.48
MEDS-11255	258.99	492.08	88.19	167.56	84.50	160.56
MEDS-42161-st	969.97	1842.95	50.54	96.03	48.42	92.00
MEDS-356839-st	8200.83	15 581.58	31.63	60.10	32.38	61.52
MEDS-716471-st	18 003.07	34 205.83	25.57	48.58	28.94	54.98

Table 20: Performance of MEDS in time (ms) and mega cycles (mcyc.) at 1900 MHz on an AMD Ryzen 7 PRO 5850U CPU following the SUPERCOP setup[6] computed as median of 16 randomly seeded runs each.

Scheme	pk size (byte)	sig size (byte)	key gen (mcyc.)	sign (mcyc.)	verify (mcyc.)
ed25519 (scop)	32	64	0.0484	0.0513	0.182
[164] dilithium2 (scop)	1 312	2 420	0.151	0.363	0.163
[180] falcon512dyn (scop)	897	666	19.5	0.880	0.0856
[211] sphincsf128shake256 (scop)	32	16 976	6.86	220	9.91
[211] sphincss128shake256 (scop)	32	8 080	218	3 500	4.04
[52] UOV ov-Ip	278 432	128	2.90	0.105	0.0903
[30] LESS-I	8 748	12 728	—	—	—
[21] Wavelet	3 236 327	930	7 400	1 640	1.09
[5] SDitH Var3f	144	12 115	—	4.03	3.04
[5] SDitH Var3sss	144	5 689	—	994	969
MEDS-8445-st-f	8 445	13 914	402	120	114
MEDS-11255-st	11 255	11 586	491	168	160
MEDS-42161-st	42 161	9 584	1 840	96.0	92.0
MEDS-716471-st	716 471	6 498	34 200	48.6	55.0

Table 21: Performance comparison to other relevant schemes (mcyc. rounded to three significant figures). Data marked with '(scop)' is from the SUPERCOP website. For SPHINCS+ we list results for the 'simple' variant.

XIV

GUIDE TO THE DESIGN OF SIGNATURE SCHEMES

Cryptography based on group actions has been studied since 1990. In recent years, however, the area has seen a revival, partially due to its role in post-quantum cryptography. MEDS, introduced in Chapter XIII, is an example of a signature scheme based on a group actions, and recently several other works have proposed signature schemes based on group actions, as well as a variety of techniques aimed at improving their performance and efficiency. Most of these techniques can be explained as transforming one Sigma protocol into another, while essentially preserving security.

In this chapter, we present a unified taxonomy of such techniques. In particular, we describe all techniques in a single fashion, show how they impact the performance of the resulting protocols and analyse in detail how different techniques can be combined for optimal performance. Furthermore, to provide a tangible perspective, we apply the results of our analysis to the (group action-based) candidates in the current NIST Call for Additional Signatures. This gives a full overview of the state of the art of signatures based on group actions, as well as a flexible tool which is easy to adapt and employ in the design of future schemes.

This chapter is an abridged version of the paper

> Giacomo Borin, Edoardo Persichetti, Federico Pintore, Krijn Reijnders, and Paolo Santini. "A Guide to the Design of Digital Signatures based on Cryptographic Group Actions". In: *Journal of Cryptology* **TODO:** fix.**TODO:** fix (2024), **TODO:** fix.

I focus on the generic scheme optimization techniques, and their combinations, and the analysis of signature schemes based on group actions. To be conscise and clear, the description of the optimizations is slightly less formal. For an extensive and formal analysis of each transformation, I refer to the full paper.

§1 Introduction

Formally introduced in 1990 by Brassard and Yung [71], cryptographic group actions have seen a renewed interest in recent years, thanks to their flexibility in constructing post-quantum signature schemes and other primitives based on various notions of isomorphism, be that isogenies between elliptic curves [54, 96, 104, 143, 144, 155], equivalence of linear or matrix codes [30, 56], including MEDS as described in Chapter XIII, isomorphism of lattices [165], alternating trilinear forms [362] and others.

Nowadays, the landscape of digital signatures based on group actions is broad. The different signature schemes that have been proposed so far are all built on a Sigma protocol Π, as defined in Definition I.6, which allow a prover to prove knowledge of a group element whose action sends a public element of the set on which the group acts into another public set element, with the simplest example given in Protocol I.3. The protocol Π has been instantiated from a variety of group actions, leading to the current wide range of signature schemes based on group actions.

Furthermore, several generic and scheme-specific techniques to increase the performance or achieve trade-offs between scheme parameters have been presented [30, 53, 56, 144, 362], part of which we have seen already in Chapter XIII. with recent proposals focusing on adaptating the MPC-in-the-head paradigm [234], which can be applied to a variety of mathematical assumptions [175, 176, 204, 230], to the protocols based on group actions [227].

In practice, we find that while designing cryptographic primitives based on group actions, one has to consider a significant amount of literature and a wide variety of variables to achieve optimal performance. What is missing is a unified description of these techniques and clear insight into the effect of different combinations of these techniques on the performance of signature schemes based on group actions.

Contributions

This work aims to fill this gap in the literature, by presenting a complete taxonomy of the current techniques used in the design of signature schemes based on group actions. We do this in all generality, using a unified description framework, and providing a detailed analysis of each technique's impact on performance and security. Our main contribution is Table 22, which summarises the effectiveness of all current techniques and fruitful combinations in terms of

(maximum) signature size, public key size, signing time and verification time. To derive Table 22, we discuss each technique individually in Section 3 and combinations of these techniques in Section 4. More precisely, we discuss

- Multiple Keys, labelled by **MK**,
- Fixed-Weight Challenges, labelled by **FW**,
- Seed Tree, labelled by **ST**, and
- MPC-in-the-Head, labelled by **MPC**, and two transformations specific for the MPC-in-the-head setting. To the best of our knowledge, they have not appeared in the literature before.
  - Skipping Edges, labelled by **Skip**, and
  - Hypercube, labelled by **Cube**.

The rows of Table 22 report the results obtained for digital signatures constructed by applying the Fiat-Shamir transform [178] to the Sigma protocol Π, combined with various different choices of optimisation techniques. The efficiency of signature schemes is assessed by a few different metrics, such as size or speed, each of them corresponding to a column in the table. More precisely,

- the first column provides the reference to the subsection of the paper where the efficiency of the corresponding signature scheme is analysed in full details,
- the column **Max Sig. Size** reports the maximum signature size where, to simplify reading, we have removed the "static" part consisting of salt and σ_2 (the challenge string) which is identical for all cases,
- the column **Pk Size** reports the size of the public key,
- the columns **Sign. Time** and **Ver. Time** report, respectively, the time required to perform signing and verification, expressed in numbers of required group action evaluations, as a function of the system parameters. An asterisk indicates a possibly expensive Merkle tree computation must be accounted for.

- the column **Condition**, describes the necessary condition on parameters to be satisfied for each particular combination.

The Hypercube technique requires the group action to be commutative, reflected by the $\checkmark_C$ marks in the **Cube** column. The Skipping Edges technique enjoys two variants (edges skipped on the left or on the right), which we mark by $\checkmark_L$, resp. $\checkmark_R$, in the **Skip** column to denote which variant is considered.

Based on this systematic and unified approach, we are able to effectively summarise the state of the current digital signatures schemes based on group actions in Section 5. We propose several new parameters for some of these schemes, derived from Table 22, which are effectively an improvement over the original ones in specific metrics. Thus, we show how the comprehensive analysis provided in this work becomes an effective tool to design and improve the efficiency of digital signatures based on group actions.

	Transformations						Bandwidth Cost (in ℓ_G, ℓ_X and λ)		Comp. Cost (in gae)		
Section	**MK**	**FW**	**ST**	**MPC**	**Skip**	**Cube**	**Max Sig. Size**	**Pk Size**	**Sign. Time**	**Ver. Time**	**Condition**
Sec. 2	-	-	-	-	-	-	$t\ell_G$	ℓ_X	t	t	$t = \lambda$
Sec. 3.1	✓	-	-	-	-	-	$t\ell_G$	$s\ell_X$	t	t	$(s+1)^t \geq 2^\lambda$
Sec. 3.2	-	✓	-	-	-	-	$(t-w)\lambda + w\ell_G$	ℓ_X	t	t	$\binom{t}{w} \geq \lambda$
Sec. 3.3	-	-	✓	-	-	-	$N_{\mathsf{seed}}\lambda + w\ell_G$	ℓ_X	t	t	$(s+1)^t \geq 2^\lambda$
Sec. 3.4	-	-	-	✓	-	-	$t((m-1)\lambda + \ell_G)$	ℓ_X	tm	tm	$(m+1)^t \geq 2^\lambda$
Sec. 3.5	-	-	-	✓	$✓_L$	-	$t\big((m-1)\lambda + \ell_G + 2\lambda\big)$	ℓ_X	tm	$t\left(1+\frac{m}{2}\right)$	$(m+1)^t \geq 2^\lambda$
Sec. 3.5	-	-	-	✓	$✓_R$	-	$t((m-1)\lambda + \ell_G)$	ℓ_X	tm	$t\left(1+\frac{m}{2}\right)$	$(m+1)^t \geq 2^\lambda$
Sec. 3.6	-	-	-	✓	-	$✓_C$	$t((m-1)\lambda + \ell_G)$	ℓ_X	$t\log(m)$*	$t\log(m)$	$(m+1)^t \geq 2^\lambda$
Sec. 4.1	-	✓	✓	-	-	-	$N_{\mathsf{seed}}\lambda + w\ell_G$	ℓ_X	t	t	$\binom{t}{w} \geq \lambda$
	✓	✓	✓	-	-	-	$N_{\mathsf{seed}}\lambda + w\ell_G$	$s\ell_X$	t	t	$\binom{t}{w}s^w \geq \lambda$
Sec. 4.2	-	-	✓	✓	-	-	$t(\lceil\log(m)\rceil\lambda + \ell_G)$	ℓ_X	tm	tm	$(m+1)^t \geq 2^\lambda$
	✓	-	✓	✓	-	-	$t(\lceil\log(m)\rceil\lambda + \ell_G)$	$s\ell_X$	tm	tm	$(sm+1)^t \geq 2^\lambda$
	-	✓	✓	✓	-	-	$N_{\mathsf{seed}}\lambda + w(\lceil\log(m)\rceil\lambda + \ell_G)$	ℓ_X	tm	tm	$\binom{t}{w}m^w \geq 2^\lambda$
	✓	✓	✓	✓	-	-	$N_{\mathsf{seed}}\lambda + w(\lceil\log(m)\rceil\lambda + \ell_G)$	$s\ell_X$	tm	tm	$\binom{t}{w}(sm)^w \geq 2^\lambda$
	-	-	✓	✓	-	$✓_C$	$t(\log(m)\lambda + \ell_G)$	ℓ_X	$t\log(m)$*	$t\log(m)$	$(m+1)^t \geq 2^\lambda$
	✓	-	✓	✓	-	$✓_C$	$t(\log(m)\lambda + \ell_G)$	$s\ell_X$	$t\log(m)$*	$t\log(m)$	$(sm+1)^t \geq 2^\lambda$
Sec. 4.3	-	-	✓	✓	$✓_L$	-	$t(\lceil\log(m)\rceil\lambda + \ell_G + 2\lambda)$	ℓ_X	tm	$t(1+\frac{m}{2})$	$(m+1)^t \geq 2^\lambda$
	✓	-	✓	✓	$✓_L$	-	$t(\lceil\log(m)\rceil\lambda + \ell_G + 2\lambda)$	$s\ell_X$	tm	$t(1+\frac{m}{2})$	$(sm+1)^t \geq 2^\lambda$
	-	-	✓	✓	$✓_R$	-	$t(\ell_G + 3\lambda)$	ℓ_X	tm	$t(1+\frac{m}{2})$	$(m+1)^t \geq 2^\lambda$
	✓	-	✓	✓	$✓_R$	-	$t(\ell_G + 3\lambda)$	$s\ell_X$	tm	$t(1+\frac{m}{2})$	$(sm+1)^t \geq 2^\lambda$
	-	✓	✓	✓	$✓_L$	-	$N_{\mathsf{seed}}3\lambda + w(\lceil\log(m)\rceil\lambda + \ell_G + 2\lambda)$	ℓ_X	tm	$t + w\frac{m+1}{2}$	$\binom{t}{w}m^w \geq 2^\lambda$
	✓	✓	✓	✓	$✓_L$	-	$N_{\mathsf{seed}}3\lambda + w(\lceil\log(m)\rceil\lambda + \ell_G + 2\lambda)$	$s\ell_X$	tm	$t + w\frac{m+1}{2}$	$\binom{t}{w}(sm)^w \geq 2^\lambda$

Table 22: Summary of all meaningful combinations of techniques, with resulting maximum signature size, public-key size, sign time and verification time measured in group action evaluations (gae).

§2 Starting Point

Chapter I gives a general introduction to signature schemes, with a focus on Sigma protocols. In particular, Section I.2.2 derives the initial Sigma protocol from a group action, which is the starting point for this work. Thus, we assume the reader is familiar with the t-round Sigma protocol based on group actions, which we denote Π_{basic}. Recall that in Π_{basic}, we commit to t values $\tilde{x}_i = \tilde{g}_i \star x$, sample a challenge ch from $\{0,1\}^t$, and respond with a vector resp where $\mathsf{resp}_i = \tilde{g}_i \cdot g^{-\mathsf{ch}_i}$. We proceed by discussing transformations, and their combinations, of this protocol.

We will in general denote commitments to set elements by $\tilde{x} \in X$, and the associated group elements by $\tilde{g} \in G$, so that $\tilde{x} = \tilde{g} \star x$ for a known value $x \in X$. In visualisations, red dotted arrows represent secret information, e.g. the secret key(s), solid black arrows represent commitments, and dashed gray arrows for a map known only to those that know the associated red secret information.

To unify notation, we emphasize that for Π_{basic} with $t = \lambda$ the soundness error is reduced to $2^{-\lambda}$, and is turned into a EUF-CMA-secure signature via the Fiat-Shamir transform. The **average signature size** is given by

$$\mathsf{SigSize}(\mathsf{FS}(\Pi_{\text{basic}})) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\Big(\underbrace{\frac{1}{2}\lambda}_{\mathsf{resp\ for\ ch}_i=0} + \underbrace{\frac{1}{2}\ell_G}_{\mathsf{resp\ for\ ch}_i=1} \Big) . \tag{47}$$

In practice, the **maximum signature size** is more relevant, as an implementation needs to allocate a certain amount of bits for a signature. In this instance, this is given by

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi_{\text{basic}})) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\ell_G \tag{48}$$

which corresponds to row 1 of Table 22. Whenever the average and maximum values differ, we list both the average and maximum signature size. For all known practical instantiations the signing and verification costs are dominated by the computations of group actions evaluation (gae), so we approximate the signing time as λ gae. We do that also for the techniques presented later. As this work is practice-focused, we list the maximum signature size and the timings in Table 22 to ease comparison between different techniques for real-world applications.

REPRESENTATIONS OF ELEMENTS. The way that elements are represented is of significant importance to the bandwidth efficiency of signature schemes based on group actions.

- *Group elements* that are used to obtain commitments are always sampled uniformly at random from G, thus can be generated and represented by a seed of length λ.
- *Challenges* are usually simply represented as integers[1].
- *Responses* consist of various non-random group elements, corresponding to the matching path required. Efficient representations of group elements are an active area of research.

Precisely in the case where the challenge $\mathsf{ch}_i = 0$, we are asked to respond with the randomly generated group element $\tilde{g}_i$ such that $\tilde{x}_i = \tilde{g}_i \star x$, which can be represented by the seed. Thus, for those rounds with $\mathsf{ch}_i = 0$, the response is only λ bits instead of ℓ_G bits.

§3 Transformations of Sigma Protocols

We present for each technique a description, a visualisation, it effect on the metrics for the resulting signature scheme, and a brief discussion on security. Thus, each subsection describes precisely the information summarized in Table 22.

3.1 MULTIPLE KEYS

De Feo and Galbraith [144] first presented the idea of using multiple public keys for group actions to decrease the signature size by reducing the soundness error. This transformation is meant to be applied *before* any increase in the number of rounds.

TRANSFORMATION. By using multiple public keys $y_1, \ldots, y_s$ associated to different secret keys $g_1, \ldots, g_s$ such that $y_i = g_i \star x$, the challenge spaces increases from $\{0, 1\}$ to $\{0, \ldots, s\}$ as we now reveal a path from y_{ch} to the commitment $\tilde{x}$, where $y_0 = x$.

1 The integer 0 represents the challenge that asks to link each commitment $\tilde{x}$ to the origin x. This particular challenge is referred to as the *consistency check*, as it asks for the information to repeat the commitment generation.

This reduces the soundness error and thus the resulting signature size, at the cost of increasing the public key size. The protocol does not change much: instead of sampling a challenge ch uniform random from $\{0,1\}$, we sample ch from $\{0,1,\dots,s\}$. The response consists of $\mathsf{resp} = \tilde{g}g_{\mathsf{ch}}^{-1}$, with the only difference the larger range for ch. The verification step takes into account that the response resp is applied to y_{ch}, thus verifies that $\tilde{x} \stackrel{?}{=} \mathsf{resp} \star y_{\mathsf{ch}}$. For the remainder of this section, we denote this protocol Π'.

VISUALISATION. A visualisation of the protocol is shown in Figure 27. By committing to a single $\tilde{x} \in X$ we get $s+1$ possible paths to reveal by composing the path $x \to \tilde{x}$ with the inverses of secret paths $x \to y_j$.

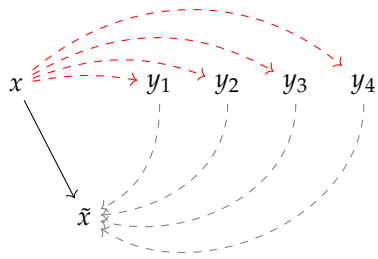


Figure 27: Multiple Keys, initiated with $s = 4$ public keys. The commitment $\tilde{x}$ can be linked to any of the public keys y_i.

RESULTING METRICS. The transformation increases the size of the public key to $s \cdot \ell_X$ where we assume that $x = y_0$ is part of the public parameters. We decrease the **soundness error** to

$$\varepsilon' = \frac{1}{|\{0,\dots,s\}|} = \frac{1}{s+1}\,. \tag{49}$$

This means that, in principle, this optimisation could be used alone, without any iterations, if s was chosen large enough (e.g. $s+1 = 2^\lambda$). However, due to the exponential increase in public key size, this approach is de facto not applicable in practice. This optimisation is therefore more directly combined with Π_{basic}, setting $t = \lceil \lambda / \log(s+1) \rceil$. Then, when transformed into a signature scheme via the Fiat-Shamir transform, the resulting **average signature size** is given by

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\Big(\underbrace{\frac{1}{s+1}\lambda}_{\mathsf{resp\ for\ ch}=0} + \underbrace{\frac{s}{s+1}\ell_G}_{\mathsf{resp\ for\ ch}\neq 0} \Big) \tag{50}$$

and the **maximum signature size** is therefore

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\ell_G \tag{51}$$

which corresponds to row 2 of Table 22. Hence, this optimisation yields a tradeoff between signature size and public key size. On the one hand, we require fewer rounds t thanks to the reduced soundness error, which improves speed. On the other hand, the size of the public key grows linearly in s.

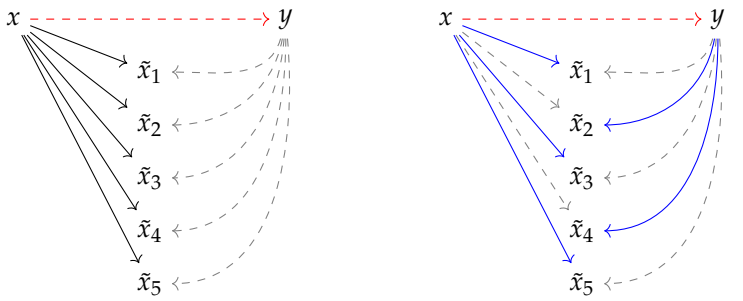
To decrease the combined size of public key and signature, the optimal value of s depends on the specific group action used, in particular on the balance between ℓ_X and ℓ_G. For schemes where set elements are relatively large, such as LESS or MEDS, the optimal value s is small (e.g. 4 or 5). Schemes with small set elements, such as SeaSign or CSI-FiSh, use a much larger number of public keys, compressed in a Merkle tree structure. For the response, only the set elements required for the challenges are transmitted (and verified with a Merkle proof). It should be noted that this strategy makes sense for the situations in which set elements and group elements are of comparable size. Moreover, this transformation still requires that all the public keys are computed during the key generation phase, so this should be considered as an additional limitation.

SECURITY. The obtained protocol Π' is complete, honest-verifier zero-knowledge and special sound. The proof is easily reduced to a variant of vectorisation given a tuple $(x, y_1, \ldots, y_s)$ instead of (x, y), with all y_i and x in the same orbit. This problem can be reduced to the vectorisation problem with a single pair (x, y), as shown in [30].

3.2 FIXED-WEIGHT CHALLENGES

Seeds generating randomly-sampled objects take up only λ bits, whereas a full group element takes up ℓ_G bits. Thus, it can be beneficial to increase the number of rounds where $\mathsf{ch}_i = 0$, so that we can reply wth the seed seed_i that generates the i-th $\tilde{g}_i$ corresponding to $\tilde{x}_i = \tilde{g}_i \star x$.

In practice, we accomplish this by sampling challenges from a *skewed* distribution, so that only a few challenges, say w, are non-zero, and the rest are zero.



(a) Example with $t = 5$ rounds.

(b) Response to $\mathsf{ch} = 01010$, with $w = 2$.

Figure 28: Example with $\mathsf{ch} \in \{0,1\}^t$ for $t = 5$ and $w = 2$. Responses in blue, starting from x when $\mathsf{ch}_i = 0$, and from y when $\mathsf{ch}_i = 1$.

In other words, we sample the string of challenges $\mathsf{ch} \in \{0,1\}^t$ as a vector of a pre-determined, fixed Hamming weight w (i.e. number of non-zero positions). Hence, this optimisation is commonly referred to as *Fixed-Weight Challenges* [53].

TRANSFORMATION. By fixing the Hamming weight of ch, that is, the commitment is sampled uniformly random from $\{v \in \{0,1\}^t \mid \mathsf{wt}(v) = w\}$ for a fixed weight w, we increase the number of rounds and decrease the communication cost. The rest of the protocol remains the same. For the remainder of this section, we denote this protocol Π'.

VISUALISATION. An example of a 5-round protocol with fixed weight $w = 2$ can be seen in Figure 28, where $t - w = 3$ challenges must be zero.

RESULTING METRICS. The transformation results in a **soundness error** of

$$\varepsilon' = \frac{1}{|\{v \in \{0,1\}^t \mid \mathsf{wt}(v) = w\}|} = 1/\binom{t}{w}.$$

To reach λ-bit security, we must therefore increase the number of rounds t beyond $t = \lambda$ until $\binom{t}{w} \geq \lambda$.

For each response, we send t group elements. Out of these, each of the $(t - w)$ group elements $\tilde{g}_i$ for $\mathsf{ch}_i = 0$ can be represented by a seed of size λ, whereas the remaining w group elements $\tilde{g}_i g^{-1}$ for $\mathsf{ch}_i \neq 0$ are communicated

as a full group element of size ℓ_G. After applying the Fiat-Shamir transform, the **exact signature size** is given by

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + \underbrace{(t-w)\cdot\lambda}_{\text{resp for } \mathsf{ch}_i=0} + \underbrace{w\cdot\ell_G}_{\text{resp for } \mathsf{ch}_i\neq 0} \tag{52}$$

which corresponds to row 3 of Table 22. Thus, Fixed-Weight Challenges meaningfully decreases signature size whenever there is a noticeable asymmetry between λ, for $\mathsf{ch}_i = 0$, and ℓ_G, for $\mathsf{ch}_i \neq 0$, at the cost of slower signing and verification, as the number of group action evaluations increases.

SECURITY. The protocol Π' is complete, honest-verifier zero-knowledge, and special sound [53].

3.3 SEED TREE

In t-round Sigma protocols, we generate the random group elements $\tilde{g}_i \in G$ by seeds seed_i, and communicate seed_i whenever $\mathsf{ch}_i = 0$. This can be improved by using a *puncturable pseudorandom function* (PRF) [59], which allows us to *generate* t seeds from a single root $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^\lambda$ and to *reveal* the subset of all seed_i with $i \in S$ for a given set $S \subseteq \{0,\ldots,t\}$.

An instantiation of such a puncturable PRF is as a *seed tree* [53]: from the root $\mathsf{seed}_{\mathsf{root}}$, we derive a tree of seeds, by hashing, until we have more than t leaves. We can disclose a subset of intermediate nodes in such a way that anyone can recompute the final leaves seed_i for $i \in S$ and therefore derive $\tilde{g}_i \in G$, with $S = \{i \mid \mathsf{ch}_i = 0\}$, without learning anything on the other leaves seed_i for $i \notin S$. The number of intermediate nodes we need to reveal depends on the precise subset $S \subseteq \{0,\ldots,t\}$ but can be upperbounded to some value $N_{\mathsf{seed}} \leq |S|$, effectively decreasing the communication cost for the $\mathsf{ch}_i = 0$ part of the signature to $N_{\mathsf{seed}} \cdot \lambda$. More precisely, to use a seed tree, we require three functions.

1. SeedTree: a function that on input of the root $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^\lambda$ constructs a binary tree with at least t leaves seed_i, by recursively expanding each seed using a PRF.

2. GetNodes: a function that, on input a root $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^\lambda$ and a subset $S \subseteq \{0,\ldots,t\}$ derives a list L of intermediate nodes $\mathsf{intseed}_j$ that cover

the leaves seed_i for $i \in S$. In other words[2], the subtrees derived from each $\mathsf{intseed}_j$ cover precisely the seed_i for $i \in S$ and reveal no other seed_i for $i \notin S$.

3. GetLeaves: a function that, given the list L of intermediate nodes $\mathsf{intseed}_j$ and the subset $S \subseteq \{0, \dots t\}$ derives the required leaves seed_i for $i \in S$.

TRANSFORMATION. From the three functions above, the use of a seed tree in a Sigma protocol of t-rounds follows naturally.

During the commitment phase, we first sample a root $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^\lambda$ and derive t seeds seed_i using SeedTree. We use these seeds seed_i to derive pseudo-random $\tilde{g}_i \in G$, and commit to $\tilde{x}_i = \tilde{g}_i \star x$ for $i \in \{1, \dots, t\}$. The challenge remains the same.

The response for any round with $\mathsf{ch}_i \neq 0$ remains the same, however, for the subset $S := \{i \in \{0, \dots, t\} \mid \mathsf{ch}_i = 0\}$, we use GetNodes with input $\mathsf{seed}_{\mathsf{root}}$ and S to obtain a list L of intermediate nodes $\mathsf{intseed}_j$. The list L is included in the response.

For verification, the verifier applies GetLeaves on L and S to retrieve the required nodes seed_i for $i \in S$ and derives $\tilde{g}_i$ for every $\mathsf{ch}_i = 0$, which allow the verifier to recompute $\tilde{x}_i = \tilde{g}_i \star x$ in those rounds. For the other rounds i with $\mathsf{ch}_i \neq 0$, the response contains the required element $\tilde{g}_i \cdot g^{-1}$, which allows the verifier to recompute $\tilde{x}_i = \tilde{g}_i \cdot g^{-1} \star y$. For the remainder of this section, we denote this protocol Π'.

VISUALISATION. Figure 29 visualises the seed tree for $t = 2^2$ rounds, where two internal nodes are revealed to derive three seeds.[3]

RESULTING METRICS. First of all, the **soundness error** is unchanged with this transformation. The signature size is instead affected considerably, depending on the number of internal seeds N_{seed} we need to communicate. In the full paper on which this chapter is based [61], we prove the following upper bound on N_{seed}.

2 The precise technical definition is much more easily understood after a quick glance at the visualisation, in Figure 29.

3 We assume for visualisation purposes that t is a power of 2. Whenever this is not the case, it is easy to derive similar tree structures with precisely t leaves, or to use a tree with $2^k > t$ leaves and ignoring the last $2^k - t$ leaves.

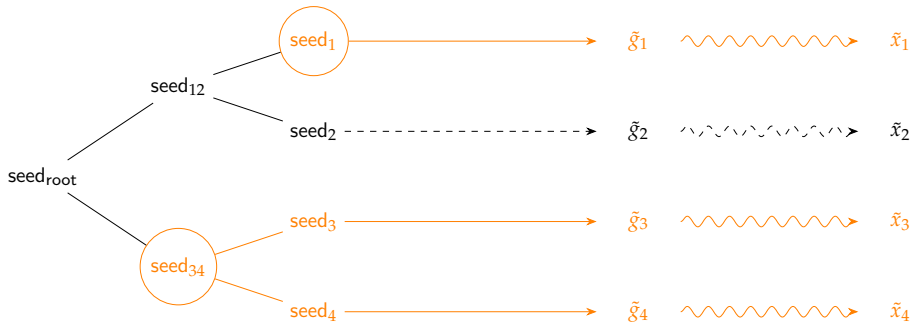


Figure 29: Example of seed tree structure, where the root $\mathsf{seed}_{\mathsf{root}}$ generates 4 leaves seed_i during commitment. On a challenge ch with $\mathsf{ch}_1 = 0$, $\mathsf{ch}_3 = 0$ and $\mathsf{ch}_4 = 0$, revealing the encircled nodes $\mathsf{intseed}_1 = \mathsf{seed}_1$ and $\mathsf{intseed}_2 = \mathsf{seed}_{34}$ is enough to disclose $\tilde{g}_1, \tilde{g}_3$ and $\tilde{g}_4$, whereas the dashed information on $\tilde{g}_2$ stays secret.

Proposition 1. Let t be the number of rounds, and let u be the Hamming weight of t, with binary expansion of t written as $t = \sum_{i=1}^{u} t_i$ with t_i the appropriate power of 2.

Then, for any set $S \subseteq \{1, \ldots, t\}$ of size $w := |S|$, the number of internal nodes we need to reveal to cover all leaves seed_i with $i \in S$ is no greater than

$$N_{\mathsf{seed}} = w \log(t/w) + u - 1 \,. \tag{53}$$

With Proposition 1 that the Hamming weight $w := \mathsf{wt}(\mathsf{ch})$ gives us an upperbound on N_{seed} and the communication cost in the signature for all $\mathsf{ch}_i = 0$ decreases from $(t - w) \cdot \lambda$ to $N_{\mathsf{seed}} \cdot \lambda$. Thus, when transformed into a signature scheme via the Fiat-Shamir transform, the resulting **signature size** is given by

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + N_{\mathsf{seed}}\lambda + w\ell_G \tag{54}$$

which corresponds to row 4 of Table 22.

For large sizes of t, computing the full seed tree or the required leaves from the list of internal nodes may take considerable computational effort, which may noticeably affect signing and verification times.

SECURITY. Beullens, Katsumata, and Pintore [53, Lemma 1] show that, when modelling the PRF as a random oracle, the internal nodes obtained by GetNodes

using $\mathsf{seed}_{\mathsf{root}}$ are indistinguishable from the one generated via a simulator without access to $\mathsf{seed}_{\mathsf{root}}$ for any computationally unbounded adversary making up to a polynomial number of calls to the random oracle.

3.4 MPC-IN-THE-HEAD

Previous optimizations all computed every commitment $\tilde{x}_i$ by a single group action $\tilde{x}_i = \tilde{g}_i \star x$. The following optimizations, we explore optimizations that arrive by longer walks, i.e., multiple group actions applied sequentially to x. In particular, such protocols are deeply linked with the MPC-in-the-head technique [217, 227].

The core idea[4] is the following: consider m randomly generated group elements $\tilde{g}_i$ applied iteratively to x to obtain m commitments $\tilde{x}_i$, as

$$x \xrightarrow{\tilde{g}_1} \tilde{x}_1 \xrightarrow{\tilde{g}_2} \ldots \xrightarrow{\tilde{g}_m} \tilde{x}_m.$$

As long as we know the secret g for the public key (x, y) with $y = g \star x$, we can compute both paths $x \to \tilde{x}_i$ from $y \to \tilde{x}_i$ as

$$x \xrightarrow{\prod_{j=1}^{i} \tilde{g}_j} \tilde{x}_i \quad \text{and} \quad y \xrightarrow{\prod_{j=1}^{i} \tilde{g}_j \cdot g^{-1}} \tilde{x}_i.$$

As long as we do not reveal a path to-and-fro $\tilde{x}_i$ from *both* x and y, we ensure g remains secret. To prove knowledge of g, a challenger may pick some $\tilde{x}_i$ and challenge us to reveal the paths $x \to \tilde{x}_1 \to \ldots \to \tilde{x}_i$, and the remaining paths $y \to \tilde{x}_{i+1} \to \ldots \to \tilde{x}_m$, keeping only the path $\tilde{x}_i \to \tilde{x}_{i+1}$ a secret. This increases the challenge space to $\mathsf{ch} \in \{0, \ldots, m\}$ per round, with $\mathsf{ch} = 0$ corresponding to revealing only the path $y \to \tilde{x}_1 \to \ldots \to \tilde{x}_m$.

Remark 1. This technique is associated with *MPC-in-the-Head* because the protocol can be seen as the simulation of $m + 1$ users doing an MPC evaluation of the group action $g \star x$ by sequentially applying random group elements $\tilde{g}_i$ per user i.

4 Again, we strongly recommend a quick glance at the visualisation in Figure 30 before reading the details.

TRANSFORMATION. We explain the transformation for a single round, which is then easily generalized to t rounds. The prover starts by generating m random group elements $\tilde{g}_i$, computes

$$x \xrightarrow{\tilde{g}_1} \tilde{x}_1 \xrightarrow{\tilde{g}_2} \dots \xrightarrow{\tilde{g}_m} \tilde{x}_m,$$

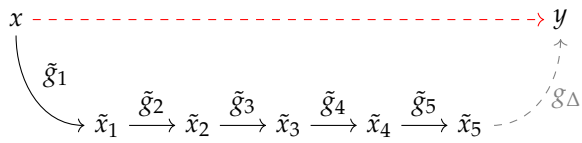
and commits to all $\tilde{x}_i$. The challenge ch is sampled uniformly random from $\{0, \dots, m\}$. When $\mathsf{ch} \neq m$, the prover replies with

- all $\tilde{g}_i$ with $i \leq \mathsf{ch}$ to connect x to all $\tilde{x}_i$ for $i \leq \mathsf{ch}$,
- the element $\tilde{g}_y := \tilde{g}_{i+1} \cdot \dots \tilde{g}_1 \cdot g^1$ to reveal the path $y \to \tilde{x}_{\mathsf{ch}+1}$, and
- all $\tilde{g}_i$ with $i \geq \mathsf{ch}+1$ to connect $\tilde{x}_{\mathsf{ch}+1}$ to $\tilde{x}_i$ for all $i \geq \mathsf{ch}+1$.

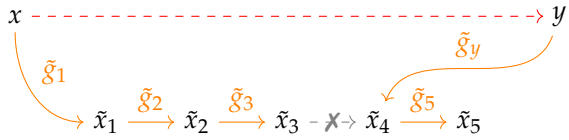
When $\mathsf{ch} = m$, the prover simply replies with all $\tilde{g}_i$ to reveal the path $x \to \tilde{x}_1 \to \dots \to \tilde{x}_m$.

With the response, the verifier recomputes the paths $x \to \tilde{x}_i$ for all $i \leq \mathsf{ch}$, the path $y \to \tilde{x}_{\mathsf{ch}+1}$, and the paths $\tilde{x}_{\mathsf{ch}+1} \to \tilde{x}_i$ for all $i \geq \mathsf{ch}+1$. Whenever $\mathsf{ch} = m$, the verifier simply only computes the paths $x \to \tilde{x}_i$. For the remainder of this section, we denote this protocol Π'.

VISUALISATION. Figure 30 shows an example of the paths in the commitment, with g_Δ computed as $g_\Delta = g \cdot \tilde{g}_1^{-1} \cdot \dots \cdot \tilde{g}_m^{-1}$.



(a) The commitment to five set values $\tilde{x}_i \in X$.



(b) For $\mathsf{ch} = 3$, we reveal the paths $x \to \tilde{x}_1 \to \tilde{x}_2 \to \tilde{x}_3$ and $y \to \tilde{x}_4 \to \tilde{x}_5$.

Figure 30: An example of the MPC-in-the-head protocol with $m = 5$ parties.

RESULTING METRICS. One can easily show that, given the same commitments $\tilde{x}_i$, two different challenges $\mathsf{ch} \neq \mathsf{ch}'$, and two valid responses, one can quickly recover the secret g: assuming $\mathsf{ch} < \mathsf{ch}'$, the second response must contain the path $x \to \tilde{x}_{\mathsf{ch}+1}$ whereas the first response must contain the path $\tilde{x}_{\mathsf{ch}+1} \to y$. Therefore, the transformed protocol has a **soundness error** per round of

$$\varepsilon' = \frac{1}{m+1}. \tag{55}$$

Per round, each response is composed of m group elements, out of which $m-1$ group elements can be described by their seeds except for the group element $\tilde{g}_y$ describing the path $y \to \tilde{x}_{\mathsf{ch}+1}$, which is communicated as a full group element in ℓ_G bits.

To achieve a λ-bit secure signature scheme, we repeat the protocol for $t = \lceil \lambda / \log(m+1) \rceil$ rounds and apply the Fiat-Shamir transform. The final **average signature size** is given by

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\Big(\underbrace{\frac{1}{m+1}(m\lambda)}_{\mathsf{resp\ for\ ch}=m} + \underbrace{\frac{m}{m+1}((m-1)\lambda + \ell_G)}_{\mathsf{resp\ for\ ch}\neq m} \Big), \tag{56}$$

thus, the **maximal signature size** becomes

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t(\underbrace{(m-1)\lambda}_{\text{seeded } \tilde{g}_j} + \underbrace{\ell_G}_{\tilde{g}_y}), \tag{57}$$

which corresponds to row 5 of Table 22. We require $t \cdot m = \left\lceil \frac{\lambda}{\log(m+1)} \right\rceil \cdot m$ group action evaluations for both signing and verifying. For large m, this quickly becomes costly: for most signature schemes based on group actions, the computational cost of generation and verification is dominated by these group action evaluations. The transformation therefore allows us to decrease the signature size, but performance takes a hit.

SECURITY. Joux [227, Thm. 2] shows that the resulting Sigma-protocol Π' is complete, honest-verifier zero-knowledge, and special sound, with soundness error $\varepsilon' = 1/(m+1)$.

Remark 2. Remarkably, a single round of the MPC-in-the-head protocol is equivalent to an m-round Fixed-Weight Challenges protocol, with weight $w = 1$. Namely, MPC-in-the-head generates the commitment path sequentially as

$$x \xrightarrow{\tilde{g}_1} \tilde{x}_1 \xrightarrow{\tilde{g}_2} \dots \xrightarrow{\tilde{g}_m} \tilde{x}_m,$$

which is equivalent to the parallel generation in Fixed-Weight Challenges

$$x \xrightarrow{\tilde{g}_1} \tilde{x}_1, \quad x \xrightarrow{\tilde{g}_2} \tilde{x}_2, \quad \dots \quad x \xrightarrow{\tilde{g}_m} \tilde{x}_m.$$

MPC-in-the-Head has challenges $\mathsf{ch} \in \{0, \dots, m\}$, whereas Fixed-Weight Challenges, viewed as m rounds with fixed weight $w = 1$, has challenges $\mathsf{ch} = (0, \dots, 0, 1, 0 \dots, 0)$ with only $\mathsf{ch}_i = 1$. Thus we can easily map one to the other. For the response, both reply with the necessary seeded group elements to reach all-but-one commitment, and require a full group element to describe the path $y \to \tilde{x}_{\mathsf{ch}+1}$, in the case of MPC-in-the-head, or $y \to \tilde{x}_i$, in the case of Fixed-Weight Challenges. A full rigorous analysis is available in the full paper [61].

3.5 SKIPPING EDGES

The previous section describes a technique to reduce the soundness error per round to $\frac{1}{m+1}$ at the cost of computing m group action evaluations per round, for both prover and verifier. By *skipping edges*, it is possible to shift part of these m group action evaluations from the verifier to the prover: with a mild increase in signature size, we can reduce the average number of group action evaluations for the verifier by a factor two on average, yielding a significant speed-up in verification.

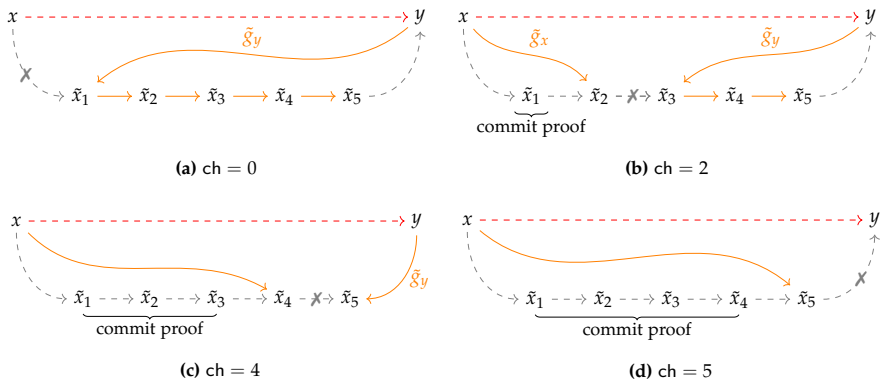
TRANSFORMATION. We assume essentially the same set-up as in MPC-in-the-Head, with a small twist. The prover similarly generate m values $\tilde{g}_i$ and the path $x \to \tilde{x}_1 \to \dots \to \tilde{x}_m$, and furthermore defines $\mathsf{cmt}_1 := \mathsf{H}(\tilde{x}_1)$ and $\mathsf{cmt}_{j+1} := \mathsf{H}(\tilde{x}_j || \mathsf{cmt}_j)$. The prover commits to all $\tilde{x}_i$ by communicating cmt_m. A challenge ch is sampled uniformly random from $\{0, \dots, m\}$. Again, the prover includes all $\tilde{g}_i$ with $i \leq \mathsf{ch}$ in the response, the element $\tilde{g}_y := \tilde{g}_{i+1} \cdot \dots \tilde{g}_1 \cdot g^{-1}$ to reveal the path $y \to \tilde{x}_{\mathsf{ch}+1}$, and all $\tilde{g}_i$ with $i \geq \mathsf{ch}+1$ to connect $\tilde{x}_{\mathsf{ch}+1}$ to $\tilde{x}_i$ for all $i \geq \mathsf{ch}+1$, but *also* includes the $(\mathsf{ch}-1)$-th commitent $\mathsf{cmt}_{\mathsf{ch}-1}$.

The verifier then computes $\tilde{g}_x := \tilde{g}_{\mathsf{ch}} \cdot \dots \cdot \tilde{g}_1$, which allows it to compute $x \to \tilde{x}_{\mathsf{ch}}$ *skipping* the individual computations of $\tilde{x}_i$ for $i < \mathsf{ch}$. The verifier

furthermore computes $y \to \tilde{x}_{\mathsf{ch}+1}$ and from there the paths $\tilde{x}_{\mathsf{ch}+1} \to \ldots \to \tilde{x}_{\mathsf{ch}}$. Given $\mathsf{cmt}_{\mathsf{ch}-1}$ and the recomputed $\tilde{x}_{\mathsf{ch}}, \tilde{x}_{\mathsf{ch}+1}, \ldots, \tilde{x}_m$, the verifier can recompute cmt_m to verify the response. For the remainder of this section, we denote this protocol Π'.

Remark 3. The above description essentially allows the verifier to skip the individual computations $x \to \tilde{x}_1 \to \ldots \tilde{x}_{\mathsf{ch}}$ by a single computation $x \to \tilde{x}_{\mathsf{ch}}$. We refer to this as skipping the *left* edges. Similarly, one can consider a variant of this protocol that skips the *right* edges, by computing the sequential commitment starting from $\mathsf{cmt}_m := \mathsf{H}(\tilde{x}_m)$ and working leftwards to cmt_1. The response contains all $\tilde{g}_i$ for $i \leq \mathsf{ch}$, the element $\tilde{g}_y$, and $\mathsf{cmt}_{\mathsf{ch}+1}$, with verification adapted similarly.

VISUALISATION. We depict the left skip variant in Figure 31, and the right skip variant in Figure 32. The orange edges correspond to the group actions being evaluated by the verifier, whereas the gray edges are the ones that are skipped. Note how the number of group action evaluations decreases for the verifier: in the *left skip* case, when $\mathsf{ch} = 0$, the verifier computes $m = 5$ group actions to verify $y \to \tilde{x}_1 \to \ldots \to \tilde{x}_5$, with no cmt_i used. However, for $\mathsf{ch} > 0$, the verifier computes only $m - \mathsf{ch} + 1$ group actions, since the commitment verification can be executed with only the recomputed values $\tilde{x}_{\mathsf{ch}}, \ldots, \tilde{x}_m$ and $\mathsf{com}_{\mathsf{ch}-1}$. A similar effect, but reversed, can be seen for the *right skip* case in Figure 32.



(a) $\mathsf{ch} = 0$ **(b)** $\mathsf{ch} = 2$

(c) $\mathsf{ch} = 4$ **(d)** $\mathsf{ch} = 5$

Figure 31: Examples for the *left* Skipping Edges variant with $m = 5$ parties.

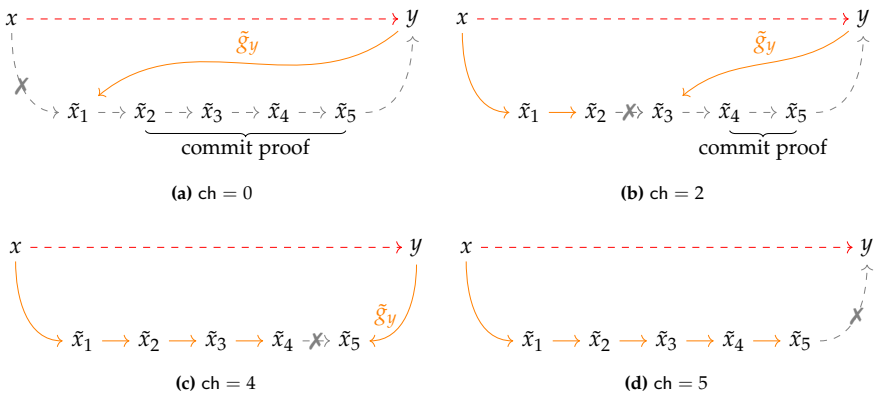


Figure 32: Examples for the *right* Skipping Edges variant with $m = 5$ parties.

RESULTING METRICS. As expected, the **soundness error** remains the same as in MPC-in-the-Head, namely $\varepsilon' = \frac{1}{m+1}$. The signing time increases slightly, as computing the commitments cmt_i requires $m-1$ additional calls to a hash function H. Verification, on the other hand, decreases from m group action evaluations to $m - \mathsf{ch} + 1$ group action evaluations.

Thus, on average we require only $1 + m/2$ group action evaluations in verification. However, the verifier must instead compute $\tilde{g}_x = \tilde{g}_{\mathsf{ch}} \cdot \ldots \cdot \tilde{g}_1$ which requires $\mathsf{ch} - 1$ group operations. We thus find an improvement in verification speed only if group operations are faster than group action evaluations. For cryptographic group actions used in practice, group operations are much faster than evaluations.

To achieve a λ-bit secure signature scheme, the protocol Π' needs to be repeated for $t = \lceil \lambda / \log(m+1) \rceil$ round before applying the Fiat-Shamir transform. The final **average signature size** for the *left skip* variant is then

$$\begin{aligned}\mathsf{SigSize}(\mathsf{FS}(\Pi')) &= \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + \\ &+ t\Big(\underbrace{\frac{1}{m+1}(m\lambda)}_{\text{resp for } \mathsf{ch}=m} + \underbrace{\frac{m}{m+1}((m-1)\lambda + \ell_G)}_{\text{resp for } \mathsf{ch}\neq m} + \underbrace{\frac{m-1}{m+1}2\lambda}_{\mathsf{com}_{\mathsf{ch}-1} \text{ for } \mathsf{ch}>1} \Big) = \\ &= \underbrace{\mathsf{SigSize}(\mathsf{FS}(\Pi))}_{\text{from (56)}} + t \cdot \frac{m-1}{m+1} \cdot 2\lambda. \quad (58)\end{aligned}$$

The signature size is maximised when $\mathsf{ch} = m-1$: we send all $\tilde{g}_i$ for $1 \le i < m$ using their seeds[5], the full group element $\tilde{g}_y$, and the commitment value com_{m-2}. Thus, the **maximum signature size** is

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t \cdot \Big(\underbrace{(m-1)\lambda}_{\text{seeded } \tilde{g}_j} + \underbrace{\ell_G}_{\tilde{g}_y} + \underbrace{2\lambda}_{\mathsf{com}_{m-1}} \Big) , \tag{59}$$

which corresponds to row 6 of Table 22.

For the *right skip* variant, we can avoid sending all $\tilde{g}_i$ for $i > \mathsf{ch}+1$, since $\tilde{g}_y$ is enough to compute the path $y \to \tilde{x}_{\mathsf{ch}+1}$. Thus, the communication cost of the responses for $\mathsf{ch} \neq m$ is, on average:

$$\frac{1}{m+1} \sum_{\mathsf{ch}=0}^{m-1} (\mathsf{ch}\,\lambda + \ell_G) . \tag{60}$$

With some algebraic manipulations, we get an **average signature size** of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + tm\Big(\frac{\ell_G}{m+1} + \frac{\lambda}{2}\Big) + t \cdot 2\lambda. \tag{61}$$

The worst case for the communications cost is $\mathsf{ch} = m-1$, where we send all $\tilde{g}_i$ for $1 \le i < m$ using a seed, and $\tilde{g}_y$ as a full group element. Thus the **maximum signature size** is

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{\mathsf{MaxSigSize}(\mathsf{FS}(\Pi))}_{\text{from (57)}} = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t(\underbrace{(m-1)\lambda}_{\text{seeded } \tilde{g}_j} + \underbrace{\ell_G}_{\tilde{g}_y}) \tag{62}$$

which corresponds to row 7 of Table 22.

SECURITY. The Skipping Edges protocol is complete, honest-verifier zero-knowledge and special sound, with soundness error $\frac{1}{m+1}$. All three properties are easily proven. For special soundness, the proof again relies on finding the right path $x \to y$ from the possible paths you can compute given the responses for two different challenges $\mathsf{ch} \neq \mathsf{ch}'$.

5 It is possible to replace the ch seeds for $\tilde{g}_1, \ldots, \tilde{g}_{\mathsf{ch}}$ by the single full group element $\tilde{g}_x = \tilde{g}_{\mathsf{ch}} \cdot \ldots \cdot \tilde{g}_1$. This decreases the signature size whenever $\ell_g < \mathsf{ch} \cdot \lambda$.

3.6 HYPERCUBE

The final transformation we present is again an improvement on MPC-in-the-Head but requires the group action to be *commutative*. As this is rare for post-quantum cryptographic group actions[6], we switch to additive notation to avoid any confusion. Commutativity of the group action allows us to rearrange the order of steps in an MPC-in-the-Head commitment, which allows us to save both signing and verification time, if computing the group operation is much faster than group action evaluation. We present a generalisation of the *Hypercube* technique [5], applied to group actions.

The scenario is similar to MPC-in-the-Head, that is, we have m randomly generated elements $\tilde{g}_i$, where we change the indexing to $i = 0$ to $m-1$ to simplify notation. We then define $\tilde{g}_m$ by $\tilde{g}_m := g - \sum \tilde{g}_i$ so that the full path from $x \to y$ is given by $\tilde{g}_0 + \ldots + \tilde{g}_{m-1} + \tilde{g}_m$. We assume that m is very large and furthermore that $m + 1 = 2^n$ for some positive integer n so that we work with exactly 2^n group elements when including $\tilde{g}_m$. Then, for any integer $0 \leq j < n$, we compute a set element $\tilde{x}_j$ by

$$\tilde{x}_j := \left(\sum_{i \in S_0(j)} \tilde{g}_i \right) \star x, \quad 0 \leq j < n$$

where $S_0(j)$ denotes the integers from 0 to $m-1$ with $j+1$-th bit equal to 0. Similarly, we define $S_1(j)$ as the integers with $j+1$-th bit equal to 1. Then, for any n-bit string $\mathsf{ch} \in \{0,1\}^n$, each bit j simulates a round of a Sigma protocol with commitment $\tilde{x}_j$: when $\mathsf{ch}_j = 0$, we reveal a path $x \to \tilde{x}_j$, whereas when $\mathsf{ch}_j = 1$, we reveal a path $\tilde{x}_j \to y$, without ever revealing the full path $x \to y$.

TRANSFORMATION. First, the prover randomly generates m elements $\tilde{g}_i$ for $i = 0$ to $m-1$ and computes $\tilde{g}_m$ as $\tilde{g}_m := g - \sum \tilde{g}_i$. The prover then commits *only* to the n values

$$\tilde{x}_j := \left(\sum_{i \in S_0(j)} \tilde{g}_i \right) \star x, \quad 0 \leq j < n.$$

The challenge ch is sampled uniformly random from $\{0,1\}^n$. The prover then includes in their response every element $\tilde{g}_i$ for $i = 0$ to m, except for the element

6 We effectively only know (variants of) CSIDH, as studied in Part 1.

$\tilde{g}_k$, where k is the integer obtained by flipping all bits of ch. With this response, the verifier recomputes the paths to each $\tilde{x}_j$ for $0 \leq j < n$, either, when $\mathsf{ch}_j = 0$, as

$$x \xrightarrow{\sum_{i \in S_0(j)} \tilde{g}_i} \tilde{x}'_j \,,$$

or, when $\mathsf{ch}_j = 1$, as

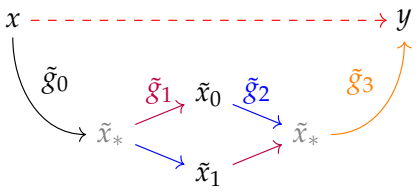
$$y \xrightarrow{-\tilde{g}_m - \sum_{i \in S_1(j)} \tilde{g}_i} \tilde{x}'_j \,.$$

However, the verifier still lacks $\tilde{g}_k$ and can thus not compute the path $x \to y$. For the remainder of this section, we denote this protocol Π'.

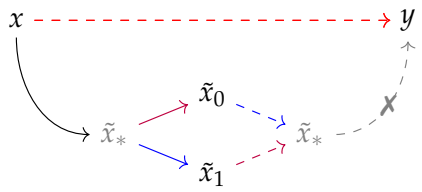
VISUALISATION. Figure 33 visualises the protocol for $m = 3$ parties, with three randomly generated elements $\tilde{g}_0$, $\tilde{g}_1$ and $\tilde{g}_2$, and $\tilde{g}_3$ computed as $g - \sum \tilde{g}_i$. As $m + 1 = 2^2$, then $n = 2$, so the prover computes and commits to

$$\tilde{x}_0 = (\tilde{g}_0 + \tilde{g}_1) \star x, \quad \tilde{x}_1 = (\tilde{g}_0 + \tilde{g}_2) \star x,$$

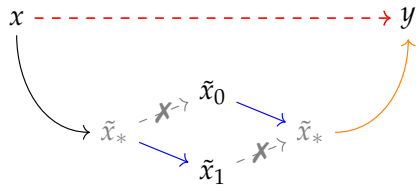
with the other nodes $\tilde{x}_*$ in gray not actually computed. In Figures 33b to 33d we see how, by sending all $\tilde{g}_i$ except $\tilde{g}_k$, the verifier can always recompute the required paths $x \to \tilde{x}_j$ or $\tilde{x}_j \to y$ using a combination of all $\tilde{g}_i$ but $\tilde{g}_k$ (dashed coloured lines are known but not used).



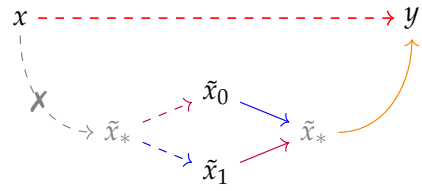
(a) Commitment generation.



(b) ch $= 00$ so $k = 11$, we reveal all but $\tilde{g}_3$.



(c) ch $= 10$ so $k = 01$, we reveal all but $\tilde{g}_1$.



(d) ch $= 11$ so $k = 00$, we reveal all but $\tilde{g}_0$.

Figure 33: The Hypercube with $m = 3$ and thus $n = 2$, with some examples of challenges, using binary representations for ch and k.

RESULTING METRICS. The **soundness error** of the transformed protocol is the same as for MPC-in-the-Head with m parties,

$$\varepsilon = \frac{1}{2^n} = \frac{1}{m+1}.$$

Thus, we need the same amount of rounds, $t = \lceil \lambda/n \rceil$, to reach λ-bit security. The response is composed of all group elements $\tilde{g}_j$ except for a single group element $\tilde{g}_k$ (the "gap"), where k is the integer represented by flipping each bit of ch, and so, the response contains $m-1$ randomly generated group elements $\tilde{g}_j$ and additionally the full group element $\tilde{g}_m$ required to compute paths $y \to \tilde{x}_j$. When $\mathsf{ch} = 0...0$, i.e. only 0s, the response contains just the random elements $\tilde{g}_j$, and $\tilde{g}_{m-1}$ is the "gap" that is left out. Thus, we get an **average signature size**, equal to m-party MPC-in-the-Head, of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\Big(\underbrace{\frac{1}{m+1}(m\lambda)}_{\mathsf{resp\ for\ ch}=0...0} + \underbrace{\frac{m}{m+1}((m-1)\lambda + \ell_G)}_{\mathsf{resp\ for\ ch}\neq 0...0} \Big) \quad (63)$$

and similarly a **maximal signature size** of

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t(\underbrace{(m-1)\lambda}_{\tilde{g}_j, \text{with } j\neq k} + \underbrace{\ell_G}_{\tilde{g}_m}) \quad (64)$$

which corresponds to row 8 of Table 22.

The major difference between MPC-in-the-Head and Hypercube is in the signing and verification cost. Whereas MPC-in-the-Head needs m group action evaluations for commitment and verification, this reduces to $n = \log(m+1)$ using Hypercube, as we only need one group action evaluation per bit of the challenge. Hence, the signing and verification costs drop down to $t \cdot \log(m+1)$ group action evaluations. However, both signer and verifier are required to compute $nm/2$ additions in the group, hence Hypercube mostly makes sense where group operations are very fast compared to group action evaluations, such as in CSI-FiSh [54]. Furthermore, as we need to generate $m = 2^n - 1$ random group elements, extreme parameters such as $n = \lambda$ are not feasible.

SECURITY. Given the similarity with the repeated version of MPC-in-the-Head, soundness can be shown in the same way. The only remaining difficulty is zero-knowledge: to simulate a transcript, take a challenge ch and derive k by flipping all bits. Then sample random elements $\tilde{g}_i$ for all $i \neq k$ and compute $\tilde{x}_j$

as the verifier would in the verification. Using these $\tilde{x}_j$ as the commitment then leads to a valid transcript.

§4 Combinations of Sigma Protocols

This section describes the most common and efficient combinations of the transformations in Section 3. As in the previous section, each subsection describes rows of Table 22

- Section 4.1 analyses combinations of Fixed-Weight Challenges with other transformations, giving rows 9 and 10.
- Section 4.2 analyses combinations of MPC-in-the-Head with other transformations, giving rows 11 to 16.
- Section 4.3 analyses the performance improvement gained from using Skipping Edges, whenever possible, giving rows 17 to 20.

4.1 Combinations with Fixed-Weight Challenges

We analyse combinations of Fixed-Weight Challenges with Seed Tree and Multiple Keys.

Seed trees. Fixed-Weight Challenges naturally combines with Seed Tree to generate the t random group elements $\tilde{g}_i$ used in the commitment. This allows us to represent the $t - w$ elements for $\mathsf{ch}_i = 0$ in the response using the (small) list of internal seeds to generate precisely those leaves $\tilde{g}_i$. This combination does not affect the **soundness error** $\varepsilon' = 1/\binom{t}{w}$. Given t and w such that $\varepsilon' \leq 2^{-\lambda}$ and applying the Fiat-Shamir transform, we get a **signature size** of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\text{salt}} + \underbrace{\lambda}_{\sigma_2} + \underbrace{N_{\text{seed}}\lambda}_{\text{resp for } \mathsf{ch}_i=0} + \underbrace{w \cdot \ell_G}_{\text{resp for } \mathsf{ch}_i\neq 0} \tag{65}$$

where N_{seed} is the number of nodes we need to release in the seed tree, from Equation (53). This equation corresponds to row 9 of Table 22.

Multiple keys and seed trees. Additionally, one can also combine Fixed-Weight Challenges with Multiple Keys using s public keys, and exploiting Seed Tree to generate the t random group elements $\tilde{g}_i$ used in the commitment. We

again represent the $t - w$ elements for $\mathsf{ch}_i = 0$ in the response using the (small) list of internal seeds. This combination leads to a soundness error of

$$\varepsilon' = \binom{t}{w}^{-1} s^{-w} . \tag{66}$$

Given t, s and w such that $\varepsilon' \leq 2^{-\lambda}$ and applying the Fiat-Shamir transform, we get a **signature size** of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + \underbrace{N_{\mathsf{seed}}\lambda}_{\text{resp for } \mathsf{ch}_i=0} + \underbrace{w \cdot \ell_G}_{\text{resp for } \mathsf{ch}_i\neq 0} \tag{67}$$

where N_{seed} is the number of nodes we need to release in the seed tree from Equation (53). This equation corresponds to row 10 of Table 22. The combination of these three transformations is the most common in the current literature. For example, LESS [19, 30], MEDS (Chapter XIII), and ALTEQ [58, 362] all use this combination, as described in more detail in Section 5.

4.2 COMBINATIONS WITH MPC-IN-THE-HEAD

We analyse combinations using MPC-in-the-Head.

SEED TREES. A Seed Tree with m leaves requires sending $\lceil \log(m) \rceil$ seeds to reveal all-but-one leaf. This is precisely the situation encountered in MPC-in-the-Head, with or without Skipping Edges. More precisely, when $\mathsf{ch} \neq m$, we can represent the $m - 1$ random group elements $\tilde{g}_i$ using the $\lceil \log(m) \rceil$ seeds of size λ, and when $\mathsf{ch} = m$ we simply send the root of the tree. Thus, at the cost of a negligible overhead computation, we get a protocol Π' with $t = \lceil \lambda / \log(m+1) \rceil$ rounds to reach λ-bit security. After applying the Fiat-Shamir transform, we get an **average signature size** of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\Big(\underbrace{\frac{1}{m+1}\lambda}_{\text{resp for } \mathsf{ch}=m} + \underbrace{\frac{m}{m+1}(\lceil \log(m) \rceil \lambda + \ell_G)}_{\text{resp for } \mathsf{ch}\neq m} \Big) \tag{68}$$

and therefore a **maximum signature size of**

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\big(\underbrace{\lceil \log(m) \rceil \lambda}_{\text{seeded } \tilde{g}_j} + \underbrace{\ell_G}_{\tilde{g}_{\mathsf{ch}+1}} \big) \tag{69}$$

which corresponds to row 11 of Table 22.

SEED TREES AND MULTIPLE KEYS. If, next to Seed Tree, we use Multiple Keys with s public keys (again with or without Skipping Edges), the challenge space per round becomes $\{0,...,m-1\} \times \{1,...,s\} \cup \{m\}$, and so, the soundness error is reduced from $\frac{1}{m+1}$ to $\frac{1}{sm+1}$. This reduces the number of rounds to t such that $(sm+1)^t \geq 2^\lambda$ and after applying the Fiat-Shamir transform, the resulting **average signature size** becomes

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\Big(\underbrace{\frac{1}{sm+1}\lambda}_{\text{resp for } \mathsf{ch}=m} + \underbrace{\frac{sm}{sm+1}(\lceil \log(m) \rceil \lambda + \ell_G)}_{\text{resp for } \mathsf{ch}\neq m} \Big) \quad (70)$$

and therefore, a **maximum signature size** of

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t(\underbrace{\lceil \log(m) \rceil \lambda}_{\text{seeded } \tilde{g}_j} + \underbrace{\ell_G}_{\tilde{g}_{\mathsf{ch}+1}}) \quad (71)$$

which corresponds to row 12 of Table 22.

FIXED-WEIGHT AND SEED TREES. By a similar logic as before, the large difference in communication cost between $\mathsf{ch} = m$ and $\mathsf{ch} \neq m$ allows us to apply Fixed-Weight Challenges to set the number of times $\mathsf{ch} = m$ occurs. Let Π' be the resulting protocol, where t denotes the number of rounds, and w the number of rounds with $\mathsf{ch}_i = m$; then the corresponding challenge set becomes

$$\{ \mathsf{ch} = (\mathsf{ch}_1, \ldots, \mathsf{ch}_t) \in \{0,...,m\}^t \mid \mathsf{wt}'(\mathsf{ch}) = w \}$$

where wt' is essentially the same as the Hamming weight, but counting the number of entries that are different from m, rather than 0. It follows that the soundness error reduces to $\binom{t}{w}^{-1} m^{-w}$. We use Seed Tree to generate the $\tilde{g}_j$ per round, and another meta-Seed Tree to generate the roots $\mathsf{seed}_{\mathsf{root}\,i}$ of each seed tree per round. Whenever $\mathsf{ch}_i = m$, we only need to release the root $\mathsf{seed}_{\mathsf{root}\,i}$, hence we can disclose the $t - w$ leaves $\mathsf{seed}_{\mathsf{root}\,i}$ of the meta-seed tree using N_{seed} nodes, following (53), to release all the required $\tilde{g}_j$ for rounds i with $\mathsf{ch}_i = m$. Hence, given t, w, m such that $\varepsilon' \leq 2^{-\lambda}$ we get a **signature size** of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + \underbrace{N_{\mathsf{seed}}\lambda}_{\text{resp for } \mathsf{ch}_i=m} + \underbrace{w(\lceil \log(m) \rceil \lambda + \ell_G)}_{\text{resp for } \mathsf{ch}_i\neq m} . \quad (72)$$

which corresponds to row 13 of Table 22. In both generation and verification, the number of group action evaluations increases to tm. By setting $m = 1$, we recover the combination from Section 4.1.

FIXED-WEIGHT, SEED TREES, AND MULTIPLE KEYS. If, next to Fixed-Weight Challenges and Seed Tree, we use Multiple Keys with s public keys, the challenge space per round is

$$\{0, ..., m-1\} \times \{1, ..., s\} \cup \{m\}$$

where the full challenge space requires w challenges with $\mathsf{ch} = m$. Since the protocol is still special sound by counting all the admissible sequence of challenges we see that the soundness error is reduced to $\binom{t}{w}^{-1}(sm)^{-w}$. Hence, given t, w, m, s such that $\varepsilon' \leq 2^{-\lambda}$, after applying the Fiat-Shamir transform, we get a **signature size** of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + \underbrace{N_{\mathsf{seed}}\lambda}_{\text{resp for } \mathsf{ch}_i = m} + \underbrace{w(\lceil \log(m) \rceil \lambda + \ell_G)}_{\text{resp for } \mathsf{ch}_i \neq m} . \tag{73}$$

which corresponds to row 14 of Table 22. The difference between (72) and (73) lies in the improved soundness error obtained via multiple keys.

HYPERCUBE AND SEED TREES. Using Hypercube with t such that $(m+1)^t = 2^{nt} \geq 2^\lambda$, we can further rewrite Equation (68) to get an **average signature size** of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\Big(\underbrace{\frac{1}{m+1}\lambda}_{\text{resp for } \mathsf{ch}=0...0} + \underbrace{\frac{m}{m+1}(n\lambda + \ell_G)}_{\text{resp for } \mathsf{ch}\neq 0..0} \Big) \tag{74}$$

and thus a **maximum signature size** of

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t(\underbrace{n\lambda}_{\tilde{g}_j, \text{with } j \neq k} + \underbrace{\ell_G}_{\tilde{g}_m}) \tag{75}$$

which corresponds to row 15 of Table 22, using $n = \lceil \log(m) \rceil$ as the number of internal nodes we communicate to reveal all-but-one leaf.

HYPERCUBE, SEED TREES AND MULTIPLE KEYS. Applying Multiple Keys with s public keys to the previous combination, the challenge space per round changes from $\{0,1\}^n$ to

$$\underbrace{\{\mathsf{ch} \in \{0,1\}^n \mid \mathsf{ch} \neq (0,\ldots,0)\}}_{m=2^n-1 \text{ elements}} \times \{1,\ldots,s\} \cup \{(0,\ldots,0)\}.$$

Hence, the soundness error is reduced from $\frac{1}{m+1}$ to $\frac{1}{sm+1}$. It follows that, given t such that $(sm+1)^t \geq 2^\lambda$ and applying the Fiat-Shamir transform, we obtain a signature scheme that performs a total of tn group action evaluations and yields an **average signature size**

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\Big(\underbrace{\frac{1}{sm+1}\lambda}_{\text{resp for } \mathsf{ch}=(1,\ldots,1)} + \underbrace{\frac{sm}{sm+1}(n\lambda + \ell_G)}_{\text{resp for } \mathsf{ch}\neq(1,\ldots,1)} \Big) \tag{76}$$

and consequently a **maximal signature size** of

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\mathsf{salt}} + \underbrace{\lambda}_{\sigma_2} + t\Big(\underbrace{n\lambda}_{\tilde{g}_j, \text{with } j\neq \mathsf{ch}} + \underbrace{\ell_G}_{\tilde{g}_{\mathsf{ch}}} \Big) \tag{77}$$

which corresponds to row 16 of Table 22.

4.3 COMBINATIONS WITH SKIPPING EDGES

We analyse the performance improvement when combining Skipping Edges with the previous techniques. We describe the *left skip* combinations first, and shortly discuss the *right skip* variants.

SEED TREES AND MULTIPLE KEYS (LEFT SKIP). We already pointed out that we can use both Seed Tree and Multiple Keys together with Skipping Edges, in the same way as usual. Given s public keys and m users for the *left skip* case, we obtain, as before, a soundness of $\frac{1}{1+sm}$ and an **average signature size** of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{\mathsf{SigSize}(\mathsf{FS}(\Pi'))}_{\text{from (70)}} + t \cdot 2\lambda \tag{78}$$

that is the same term as (70), plus the additional term due to the Merkle proof. By combining (71), we get a **maximum signature size**, independent of the number of public keys s, of

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = \underbrace{\mathsf{MaxSigSize}(\mathsf{FS}(\Pi))}_{\text{from (71)}} + t \cdot \underbrace{2\lambda}_{\mathrm{com}_*} \tag{79}$$

which corresponds to rows 17 and 18 of Table 22. The difference between using Multiple Keys or not is represented again by the trade off in the number of rounds t, and the size of the public key: in fact, for s public keys, the number of rounds is such that $(sm+1)^t \geq 2^\lambda$, thus decreases, but the public key size becomes $s\ell_X + \lambda$, hence increases.

FIXED-WEIGHT, SEED TREES AND MULTIPLE KEYS (LEFT SKIP).
When additionally applying the Fixed-Weight Challenges optimisation to the above, we get again the same improvements in soundness and signature size as in (73), with the additional cost of the Merkle proof. Using left skip, the number of group action evaluations during signing remains tm; however, the average number of group action evaluations during verification is reduced, via some algebraic manipulation, to

$$\underbrace{(t-w)}_{\mathsf{ch}=m} + \underbrace{w\left(\frac{1}{m}\sum_{\mathsf{ch}=0}^{m-1}(m+1-\mathsf{ch})\right)}_{\mathsf{ch}=0,\ldots,m-1} = t + w\cdot\frac{m+1}{2}\,. \tag{80}$$

A direct comparison between the number of node commitments during signing and verification gives

$$\eta = \frac{t + w\frac{m+1}{2}}{tm} \approx \frac{1}{m} + \frac{w}{2t} < 1\,. \tag{81}$$

This leads to a reduction in the computational complexity for verification when compared not only to the combinations involving MPC-in-the-Head (Section 4.2), as seen in (81), but also to the combinations involving Fixed-Weight Challenges (Section 4.1).

To further reduce the signature size, we adapt the strategy used for sequential committing to the intermediate nodes. For each round we commit to the first $m-1$ nodes $\tilde{x}_1, \ldots, \tilde{x}_{m-1}$ and obtain com_{m-1}, while we commit separately to the last element $\tilde{x}_m$ by $\mathrm{com}_m = \mathsf{H}(\tilde{x}_m)$. We commit to the values com_{m-1} obtained

in the t rounds via *another* Merkle Tree with t leaves. The final commitment $\mathsf{com}_{\mathsf{final}}$ is derived from the root of this tree and the other t round values com_m.

Recall that the weight w in this case is defined as the number of rounds with $\mathsf{ch}_j \neq m$, i.e. the rounds in which we recompute com_{m-1}. For the remaining $t - w$ rounds, we require N_{seed} nodes to verify the m rounds. In the end, we get a **signature size** of

$$\begin{aligned}\mathsf{SigSize}(\mathsf{FS}(\Pi')) = \underbrace{2\lambda}_{\text{salt}} + \underbrace{\lambda}_{\sigma_2} + \underbrace{N_{\mathsf{seeds}}\lambda}_{\text{resp for } \mathsf{ch}_i=m} + \underbrace{N_{\mathsf{seeds}}2\lambda}_{\text{Merkle proof for } \mathsf{ch}_i=m} + \\ + w(\underbrace{\lceil \log(m) \rceil \cdot \lambda + \ell_G}_{\text{resp for } \mathsf{ch}_i \neq m} + \underbrace{2\lambda}_{\text{com}_* \text{ for } \mathsf{ch}_i \neq m}) = \\ = \underbrace{\mathsf{SigSize}(\mathsf{FS}(\Pi))}_{\text{from (73)}} + (N_{\mathsf{seed}} + w) \cdot 2\lambda \,. \quad (82)\end{aligned}$$

which corresponds to rows 19 and 20 of Table 22.

SEED TREES AND MULTIPLE KEYS (RIGHT SKIP). For the *right skip* variant with m users we can clearly use Multiple Keys with s public keys, using a sequential PRF. This way we obtain, as before, a soundness of $\frac{1}{1+sm}$ and an **average signature size** of

$$\mathsf{SigSize}(\mathsf{FS}(\Pi')) = t \left\{ \underbrace{\frac{sm}{sm+1}}_{\mathsf{ch}\neq m} \left[\underbrace{\lambda}_{\text{seq. PRF}} + \underbrace{\ell_G}_{g'_{\mathsf{ch}+1}} + \underbrace{2\lambda}_{\text{com}_*} \right] + \underbrace{\frac{1}{sm+1}}_{\mathsf{ch}=m} \lambda \right\} . \quad (83)$$

By considering only the most expensive challenges, we get a **maximum signature size** of

$$\mathsf{MaxSigSize}(\mathsf{FS}(\Pi')) = t \cdot (\ell_G + 3\lambda) \quad (84)$$

which corresponds to rows 21 and 22 of Table 22. The number of round t is again such that $(1 + sm)^{-t} \leq 2^{-\lambda}$.

FIXED-WEIGHT (RIGHT SKIP). When applying Fixed-Weight Challenges to the right skip variant we encounter a problematic trade-off between signature size and verification time.

- when $\mathsf{ch}_i = m$, we send only the root of the seed tree, but we need to perform m group action evaluations in verification;

- when $\mathsf{ch}_i = 0$, we send at least one full group element, and perform only 1 group action evaluations in verification;
- when $0 < \mathsf{ch}_i < m$, we send one full group element, plus 3λ bits of additional information, and perform $\mathsf{ch}_i + 1 \leq m$ group action evaluations in verification.

This trade off makes the use of Fixed-Weight Challenges counterproductive with this variant. In fact, the main advantage would consist of decreasing as much as possible the ratio w/t of rounds with lightweight representations, i.e. the ones with $\mathsf{ch}_i = m$, since all the others requires at least one non-seeded group element. However, this would actually imply an increase in the number of group action evaluations, since now at least $m(t-w)$ of them are required. It follows that one cannot get optimal signature size while simultaneously reducing verification time, as in the *left skip* case. Due to this stalemate, this combination is best avoided.

§5 Signature Schemes from Group Actions

This section analyses some instantiations of the protocols based on group actions, using the language and results laid out in the previous sections. Specifically, we give a high-level overview of the candidates of the additional call for signatures issued by NIST based on group actions, namely LESS [19], MEDS [109] and ALTEQ [58], using the unified terminology developed in this work. We also do a similar analysis for the group actions derived from isogenies of elliptic curves [54, 143].

So far, we have only considered group actions as a black box. In this section, we dive into specific group actions and explain how they gain efficiency on two levels: first, using the generic techniques for group-action based Σ-protocols as described in Sections 3 and 4, and second, using techniques which are specific to each group action, such as more compact representations of group elements or set elements.

5.1 LESS: Linear Equivalence Signature Scheme

LESS [19, 30] is a signature scheme based on the hardness of the *Linear (code) Equivalence Problem* (LEP). At its core, linear code equivalence can be seen as a group action: the group G of *monomial maps* acts on the set of k-dimensional linear codes of length n over the finite field $\mathbb{F}_q$, with $k \geq n$ positive integers.

These monomial maps are isometries for the Hamming metric and consist of a permutation combined with non-zero scaling factors. Thus, G can be seen as the composition of a permutation matrix with a diagonal matrix of non-zero entries, that is, $G = S_n \times (\mathbb{F}_q^*)^n$.

HARDNESS ASSUMPTION. The linear code equivalence problem asks, given two $[n,k]_q$ linear codes $\mathcal{C}, \mathcal{C}'$, to find a monomial map μ such that $\mathcal{C}' = \mu(\mathcal{C})$. Let $\mathbf{G}$ denote a generator matrix for $\mathcal{C}$, and similarly $\mathbf{G}'$ for $\mathcal{C}'$, then the problem can be re-framed as finding the monomial matrix $\mathbf{Q}$ such that $\mathbf{G}'$ and $\mathbf{GQ}$ have the same systematic form, i.e. generate the same linear subspace. For NIST Security Category I, the LESS specification document uses the parameters

$$q = 127,\ n = 252,\ k = 126.$$

SIZE OF ELEMENTS. A random monomial map in $S_n \times (\mathbb{F}_q^*)^n$ can be represented in $\ell_G = n \cdot (\log n + \lceil \log q \rceil)$ bits. A random $[n,k]$ linear code over $\mathbb{F}_q$ in systematic form can be represented by a $k \times n$ matrix in $\mathbb{F}_q$ containing the identity matrix of dimension k as a submatrix, so $k(n-k)\lceil \log q \rceil$ bits are enough to represent it.

SCHEME OPTIMISATION. The LESS scheme instantiates Π_{basic} with the group action described above, and applies the transformations Fixed-Weight Challenges with Seed Tree, as well as Multiple Keys. The current NIST Security Category I specification proposes three sets of values:

- *balanced* has $t = 247$ rounds, weight $w = 30$ and one public key,
- *intermediate* has $t = 244$ rounds, weight $w = 20$, $s = 3$ public keys,
- *short* has $t = 198$ rounds, weight $w = 17$, $s = 7$ public keys.

Note that in [19], differently from this work, the authors use s to indicate the total number of linear codes used as public keys, included the randomly-generated one. The same remark applies to Section 5.2.

The MPC-in-the-Head and Skipping Edges techniques are not used as they are outperformed by Fixed-Weight Challenges. As the group action is not commutative, Hypercube does not apply.

SPECIFIC LESS IMPROVEMENTS. To improve the performance of LESS beyond the above description, Chou, Persichetti, and Santini [112] introduce the concept of *canonical forms*, which allow to compress the information contained

in the monomial map $\mathbf{Q}$. While more information on this can be found in [19, 112], here we only briefly point out the most relevant results:

- if two linear codes can be mapped to equivalent codes with the same canonical forms, then the equivalence between the two codes can be found in polynomial time,
- the equivalence can be verified up to canonical form by using much less information than the one contained in $\mathbf{Q}$ (the exact compression depends on the choice of the canonical form).

In this way, the communication cost required to verify the equivalence is substantially reduced, for example from 473 bytes to 237 bytes using a weaker form of canonical forms (see [19, 310]) to just 32 bytes in [112], using the most well-known choice for a canonical form. However, this also implies that the compressed monomial maps cannot be composed anymore, as $\ell_G \approx 2\lambda$. Thus, using this optimisation limits us to paths of length 1 during verification, making it impossible to combine this optimisation with MPC-in-the-Head and in particular with Skipping Edges.

CONCRETE VALUES. For NIST Security Category I, the specification submitted to round 1 [19] yields

$$\ell_G = 237 \text{ bytes}, \quad \ell_X = 13,892 \text{ bytes}, \quad \lambda = 128.$$

Table 23 then gives the performance for the different parameter sets, with sizes measured in KBytes and speed in group action evaluations (gae).

	Sizes (Kbytes)		Times (gae)	
	Pk	**Sig.**	**Sign.**	**Verif.**
Balanced	13.7	8.4	247	247
Intermediate	41.1	6.1	244	244
Short	95.9	5.2	198	198

Table 23: Performance of LESS for three different parameter sets.

5.2 MEDS: MATRIX EQUIVALENCE DIGITAL SIGNATURES

Chapter XIII introduces MEDS in great detail, as a signature scheme based on *Matrix Code Equivalence* (MCE). Here, we only go over the details reuired

for this work, and additional improvements not yet known at the writing of Chapter XIII.

For NIST Security Category I, the current MEDS specifications [109] use the parameters

$$q = 4093,\ n = 14,\ m = 14,\ k = 14.$$

SIZE OF ELEMENTS. A random isometry $(\mathbf{A}, \mathbf{B}) \in \mathrm{GL}_n(q) \times \mathrm{GL}_m(q)$ can be represented in $\ell_G = (n^2 + m^2) \cdot \lceil \log q \rceil$ bits. A random k-dimensional matrix code $\mathcal{C} \subseteq \mathbb{F}_q^{n \times m}$ can be represented by k matrices $\mathbf{C}_i \in \mathbb{F}_q^{n \times m}$. By vectorising these matrices, one gets a generator matrix $\mathbf{G}$ of size $k \times mn$, which can be represented in standard form using $\ell_X = k \cdot (n \cdot m - k) \lceil \log q \rceil$ bits. For the NIST Security Category I parameters above we would get $\ell_G = 588$ bytes and $\ell_X = 3822$ bytes.

SCHEME OPTIMISATION. The MEDS scheme instantiates Π_{basic} with the group action described above, and applies the transformations Fixed-Weight Challenges with Seed Tree, and Multiple Keys. The current NIST Security Category I specification proposes two sets of values:

- set 1 has $t = 1152$ rounds, weight $w = 14$, and $s = 3$ public keys,
- set 2 has $t = 192$ rounds, weight $w = 20$, and $s = 4$ public keys.

For both sets MPC-in-the-Head is not used, as it is outperformed by Fixed-Weight Challenges. Furthermore, the group action is not commutative, hence Hypercube does not apply. However, as ℓ_G is quite a bit larger than λ, Skipping Edges combined with Fixed-Weight Challenges, Multiple Keys and Seed Tree can improve on both parameter sets for some aspects of the scheme.

- set 3 with $t = 129$ rounds, fixed-weight $w = 10$, $m = 87$ parties and $s = 3$ public keys.

SPECIFIC MEDS IMPROVEMENTS. Two techniques improve the performance of MEDS beyond the description above, using techniques specific for MCE.

1. [109, § 3] shows how to reduce the public key size, i.e. set elements, to $k - 2$ matrices instead of k, reducing ℓ_X to $(k-2) \cdot (nm - k) \cdot \lceil \log q \rceil$ bits.
2. [111] shows how to reduce the isometry representation size, e.g. group elements, to two matrices $\mathbf{C}, \mathbf{C}' \in \mathbb{F}_q^{n \times m}$. Both can be defined in terms of the (known) basis $(\mathbf{C}_i, \ldots, \mathbf{C}_k)$ as $\mathbf{C} = \sum \lambda_i \mathbf{C}_i$ with $\lambda_i \in \mathbb{F}_q$, hence, both $\mathbf{C}$ and $\mathbf{C}'$ can be represented in $k \cdot \lceil \log q \rceil$ bits. This reduces ℓ_G to $2k \cdot \lceil \log q \rceil$.

CONCRETE VALUES. For NIST Security Category I, the MEDS specifications do not apply the second optimalization yet. For fair comparison, we keep the numbers as given in the specification. This gives us

$$\ell_G = 588 \text{ bytes}, \quad \ell_X = 3,297 \text{ bytes}, \quad \lambda = 128.$$

Table 24 then gives the performance for the different parameter sets, with sizes measured in KBytes and speed in group action evaluations (gae).

	Sizes (Kbytes)		Times (gae)	
	Pk	**Sig.**	**Sign.**	**Verif.**
Set 1	9.9	9.9	1152	1152
Set 2	13.2	12.9	192	192
Set 3 (new)	9.9	9.1	11223	569

Table 24: Performance of MEDS for three different parameter sets.

We see that the third set of parameters decreases the signature size and requires roughly only half the average group action evaluations in verification (569 against 1152). Signing, however, is negatively impacted, and becomes about 10 times slower. If we include the above optimisation, ℓ_G decreases substantially to 42 bytes and the ratio with λ drops to only 2.6×. This makes Skipping Edges much less effective, and we find that set 3 does not improve anymore.

5.3 ALTEQ: ALTERNATING TRILINEAR FORM EQUIVALENCE

ALTEQ [58, 362] is a signature scheme based on the hardness of the *Alternate Trilinear Form Equivalence* (ATFE) problem. At its core, the alternate trilinear form equivalence can be seen as a group action, where the *group* $G = \mathrm{GL}_n(q)$ acts on the *set* of alternating trilinear forms

$$\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q.$$

Group elements $\mathbf{A} \in \mathrm{GL}_n(q)$ map ϕ to $\psi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{Az})$.

HARDNESS ASSUMPTION. The alternate trilinear form equivalence problem asks to find $\mathbf{A} \in \mathrm{GL}_n(q)$ given two equivalent trilinear forms ϕ and ψ. For NIST Security Category I, the ALTEQ specification document uses the parameters

$$q = 2^{32} - 5, \; n = 13.$$

SIZE OF ELEMENTS. A random $\mathbf{A} \in \mathrm{GL}_n(q)$ can be represented in $\ell_G = n^2 \cdot \lceil \log q \rceil$ bits. A random alternating trilinear form ϕ can be described as a linear combination of $e_i^* \wedge e_j^* \wedge e_k^*$, where $1 \leq i < j < k \leq n$, which form a (known) basis of the space of alternating trilinear forms. The form ϕ can thus be represented using $\binom{n}{3}$ coefficients in $\mathbb{F}_q$, giving $\ell_X = \binom{n}{3} \cdot \lceil \log q \rceil$.

SCHEME OPTIMISATION. The ALTEQ scheme instantiates Π_{basic} with the group action described above, and applies the transformations Fixed-Weight Challenges and Multiple Keys. The current NIST Security Category I specification proposes two sets of values:

- set 1 has $t = 84$ rounds, weight $w = 22$, and $s = \quad 7$ public keys,
- set 2 has $t = 16$ rounds, weight $w = 14$, and $s = 458$ public keys.

The technique MPC-in-the-Head is not applied in the specification. Altough in [58, Remark 1.4], the authors propose to analyse MPC-in-the-Head to improve the scheme, this will improve the performance of ALTEQ as it is outperformed by Fixed-Weight Challenges. Note that the group action is not commutative, hence Hypercube does not apply.

The current submission of ALTEQ does not use Seed Tree, although in [58, Remark 1.4] the authors write that more understanding is required. Using Proposition 1, we can quantify the gain of using Seed Tree as 272 bytes for set 1 parameters while for set 2 there is no improvement.

Currently, the representation of group elements in ALTEQ is a major limitation. For example, the current NIST Security Category I parameters have $\ell_G \approx 40 \cdot \lambda$. Hence, we can use the combination of Skipping Edges and Fixed-Weight Challenges to improve the signature size while controlling the verification time. We readily compute

- set 3 with $t = 164$ rounds, fixed-weight $w = 10$, $m = 29$ parties and $s = 7$ public keys.

To find this set, we capped the average number of group action evaluations during verification to 4×84 (for this set we need 314), so that we expect

verification to be 4 times slower for set 3 parameters than for set 1 parameters. Signature generation requires 4756 group action evaluations, making it roughly 60 times slower than set 1. However, the parameter set allows us to reach a signature size below 10,000 Kbytes while still keeping the size of the public key reasonable. We stress that, when using only Fixed-Weight Challenges and not Skipping Edges, we require at least 1234 rounds to get such small signatures.

CONCRETE VALUES. For NIST Security Category I, this gives us

$$\ell_G = 676 \text{ bytes}, \quad \ell_X = 1144 \text{ bytes}, \quad \lambda = 128.$$

Table 25 then gives the performance for the different parameter sets, with sizes measured in KBytes and speed in group action evaluations (gae).

	Sizes (Kbytes)		Times (gae)	
	Pk	**Sig.**	**Sign.**	**Verif.**
Set 1 (with Seed Tree)	8.0	15.6	84	84
Set 2	524.0	9.6	16	16
Set 3 (new)	8.0	10.0	4756	314

Table 25: Performance of ALTEQ for three different parameter sets.

5.4 ISOGENY-BASED GROUP ACTIONS

Beyond the three signature schemes LESS, MEDS and ALTEQ that appear in the NIST competition, there are a few more signature schemes based on group actions. We briefly discuss these schemes and their practicality in this section.

SIGNATURES FROM CLASS-GROUP ACTIONS. The *commutative* action of class groups on sets of elliptic curves gives rise to a family of isogeny-based signature schemes. Stolbunov [360] describes how to instantiate signatures from this group action. The main example of such a group action is CSIDH [96], which we have thoroughly discussed in Part 1 as the basis for a non-interactive key exchange mechanism. This section discusses approaches towards signature schemes based on this group action.

The CSIDH group action is not effective: as the structure of the class group is highly non-trivial to compute, it is difficult to compute the action of a randomly sampled element $g \mathsf{rand} G$ on a given set element $x \in X$. This is a

major roadblock to efficient signature schemes. It can be solved with rejection-sampling, as shown in SeaSign [144, 155]. However, this requires a large amount of isogeny evaluations, making the protocol unpractical. Furthermore, the transformations described in this work, no longer apply. We therefore do not discuss SeaSign in depth.

A few solutions are currently known to make the group action effective:

CSI-FiSh. One solution is to compute the structure of the class group. This was done in a record-breaking computation for the class group referred to as CSIDH-512 by Beullens, Kleinjung, and Vercauteren [54]. Thus, for this class group, the action can be considered effective and Stolbunov's signature scheme can be instantiated, resulting in CSI-FiSh. All techniques described in this work apply, in particular Hypercube, as the group action is commutative.

SCALLOP and SCALLOP-HD. Other solutions are described in SCALLOP [143], extended to the HD-framework in SCALLOP-HD [104]. By cleverly choosing the correct prime, one can easily compute the structure of the class group used in SCALLOP, making the group action effective.

Clapoti(s). Clapoti(s) [304] instead uses higher-dimensional techniques to enable a polynomial-time evaluation of *any* class group action. We do not consider specific scheme optimisations for these group actions: currently, a single group action evaluation is impractically slow, such that scheme optimizations with respect to signature size and public key size are only of theoretical interest.

HARDNESS ASSUMPTION. For all these class group actions, the security is based on the difficulty, for any adversary, to find a secret isogeny $\varphi : E \to E'$ given only E and E', where φ is derived from a secret group element (ideal) $\mathfrak{a} \in \mathcal{C}\ell(\mathcal{O})$, and both E and E' have endomorphism rings isomorphic to $\mathcal{O}$. Wesolowski [377] studies this problem, and shows the equivalence to the oriented version of the endomorphism ring problem.

SIZE OF ELEMENTS. Group elements, for CSI-FiSh or SCALLOP(-HD), require 256 bits or 512 bits for security levels equivalent to CSIDH-512 and CSIDH-1024 respectively. For CSI-FiSh [54], set elements require one $\mathbb{F}_p$-element, i.e. 512 bits. SCALLOP [143] gives concrete instantations targeting the security level of CSIDH-512 and CSIDH-1024 in which set elements can be compressed to approximately 1600 bits and 2300 bits respectively. The HD-variant, SCALLOP-HD [104], does not give any concrete parameter sets, nor a precise description

of the size of set element representations. A quick calculation shows that a set element naively requires three $\mathbb{F}_{p^2}$-elements. This can be compressed[7] to two $\mathbb{F}_p$-elements and three integers of e bits, where e is some scheme parameter. This gives a size of approximately 1800 bits for a security level equivalent to CSIDH-512, and 3200 bits for a security level equivalent to CSIDH-1024.

SCHEME OPTIMISATION. CSI-FiSh [54, § 5.2] considers several optimisations[8] similar to the ones in this work, most notably Multiple Keys. In CSI-FiSh, we can double the number of non-trivial public keys using twisted curves thus we consider only even values for s. As set 1, we take the CSI-FiSh parameters minimizing the combined public key and signature size, with $s = 14$. As set 2, we consider the simple instantiation with $s = 2$ and no other optimisations beyond Multiple Keys. For set 3, since the action is commutative, we use Hypercube with $n = 16$, which requires us to compute a Seed Tree of size 2^{16} per round. Moreover, we can apply Multiple Keys with $s = 2$ thanks to the twist. Increasing s further does not seem to meaningfully reduce the number of group action evaluations. Other transformations notably increase the number of group action evaluations, which makes the resulting schemes unpractical with the current speed of a single group action evaluation.

CONCRETE VALUES. For CSI-FiSh with CSIDH-512 parameters, this gives us

$$\ell_G = 32 \text{ bytes}, \quad \ell_X = 64 \text{ bytes}, \quad \lambda = 128.$$

Table 26 then gives the performance for the different parameter sets, with sizes measured in bytes and speed in group action evaluations (gae).
The results from set 3 show that Hypercube can potentially improve scheme performance. However, these improvements are minor as the number of group action evaluations cannot be decreased substantially. If $\ell_X \gg \lambda$, the advantage (at parity of group actions) would be more substantial.

7 The elements $\omega(P)$ and $\omega(Q)$ can be described in terms of P and Q at a cost of four integers of e bits, or three integers using pairing techniques [258].

8 CSI-FiSh [54] also uses a slowed-down hash function, to decrease the required soundness level. This not only reduces signature sizes: when group action evaluations are slower than such a slow hash function, this improves signing and verification time. For consistency compared to the other schemes, we do not consider slow hashes in this analysis.

	Sizes (bytes)		Times (gae)	
	Pk	**Sig.**	**Sign.**	**Verif.**
Set 1	448	1104	33	33
Set 2	64	2640	81	81
Set 3	64	2304	128	128

Table 26: Performance of CSI-FiSh for three different parameter sets.

TOWARDS A BRIGHTER FUTURE?

At the time of writing this thesis, MEDS is a solid scheme. We may expect some improvements in cryptodesing, and should expect improvements in cryptanalysis, as attacks always get better, never worse. Nevertheless, with current parameters, the scheme is neither among the post-quantum signatures with favorable sizes, nor with impressive speeds. Yet MEDS is very simple: at its core, MEDS relies on matrix multiplication and repetitions of Sigma protocols. Thus, implementing MEDS is relatively easy compared to other schemes and parallelisation of the matrix operations and the rounds of the Sigma protocol could yield impressive results.

THE LIMITS OF REPETITIONS OF 1-BIT SIGMA PROTOCOLS

For isometry-based cryptography to rival other signature schemes requires more than the current approach that is at the core of LESS, MEDS and ALTEQ: All these schemes start from the 1-bit Protocol 2 and apply general or scheme-specific techniques to achieve λ-bit security, as described in Chapter XIV. Improvements of such techniques, or improvements in scheme-specific techniques, such as [111], can only somewhat reduce signature size or scheme efficiency. Most remarkably, recent results [112, 310] push the representations of isometries in LESS close to fundamental limits, thus achieving close to the optimal performance in signature size for signature schemes based on repetitions of 1-bit Sigma protocols: This achieves signatures as small as 2.8 kilobytes. While this is an impressive result, it shows that the best possible scheme based on such an approach still has rather large signatures.

A BOLD APPROACH

We must therefore be bold and explore other approaches. What is needed is a larger challenge space, preferably exponential. Ideally, as inspired by Part 2, we want the challenge to be any random isometry μ instead of a bit $c \in \{0, 1\}$, such that we require only a single isometry as a signature too, with verification reduced to a simple recomputation of this isometry.

A SQISIGN-BASED APPROACH

We will sketch such an approach[9], although many technical difficulties remain. The interested reader may compare this approach with the approach in SQIsign, where these technical difficulties are resolved in a most miraculous manner. We follow the same set-up: $\mathcal{C}_0$ is a publicly-known random code, and in **key generation**, we sample a random isometry $\mu_{\mathsf{sk}} : \mathcal{C}_0 \to \mathcal{C}_{\mathsf{pk}}$ as the secret key, with $\mathcal{C}_{\mathsf{pk}}$ as the public key. To **commit**, we sample a random isometry $\mu_{\mathsf{cmt}} : \mathcal{C}_0 \to \mathcal{C}_{\mathsf{cmt}}$ and commit to $\mathcal{C}_{\mathsf{cmt}}$. As a **challenge**, the verifier sends a random isometry $\mu_{\mathsf{ch}} : \mathcal{C}_{\mathsf{cmt}} \to \mathcal{C}_{\mathsf{ch}}$. As a **response**, we should find an 'appropriate' isometry $\mu_{\mathsf{resp}} : \mathcal{C}_{\mathsf{pk}} \to \mathcal{C}_{\mathsf{ch}}$. We visualise this in Figure 34.

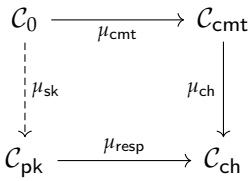


Figure 34: The sketched approach for isometry-based signatures.

This protocol runs into several difficulties. The response μ_{resp} should convince the verifier that the prover knows μ_{sk}. Thus, we must prevent that an adversary can replace the commitment in step 2 by a random isometry $\mu' : \mathcal{C}_{\mathsf{pk}} \to \mathcal{C}_{\mathsf{cmt}}$ and is able to send a valid response in step 4 by composing μ' with μ_{ch}. This leads to the following crucial observation: if the automorphism group of $\mathcal{C}_0$ is trivial, than the response μ_{resp} is unique. Therefore, such a composition would be the only valid response and a verifier would not be able to distinguish between an adversary and someone who knows μ_{sk}.

NON-TRIVIAL AUTOMORPHISM GROUPS

If we want an approach similar to SQIsign to succeed, we must therefore use codes with non-trivial automorphism groups. Starting out with $\mathcal{C}_0$ with non-trivial $\mathrm{Aut}(\mathcal{C}_0)$, the knowledge of μ_{sk} allows the prover to learn $\mathrm{Aut}(\mathcal{C}_{\mathsf{pk}})$, and as such, the prover can precompose $\mu_{\mathsf{ch}} \circ \mu_{\mathsf{cmt}} \circ \mu_{\mathsf{sk}}^{-1}$ with any $\gamma \in \mathrm{Aut}(\mathcal{C}_{\mathsf{pk}})$. However, we know too little about matrix code equivalence for instances $\mathcal{C} \to \mathcal{D}$

9 The content of this approach were sketched by me in a talk I gave at the Department of Mathematics of the University of Zagreb on June 14, 2023.

where $\mathrm{Aut}(\mathcal{C})$ and $\mathrm{Aut}(\mathcal{D})$ are unknown, but non-trivial. Further research in this area of coding theory may either validate or refute the viability of this approach.

USING ESOTERIC ISOMETRIES

Beyond these questions, the more esoteric non-lifting isometries and automorphisms, i.e. those isometries $\mu : \mathcal{C} \to \mathcal{D}$ in the rank metric that do not lift to isometries of the full space, have been explored very little for cryptographical applications. A study of these isometries as elements in $\mathrm{GL}(\mathrm{Vec}\,\mathcal{C})$, i.e. as those matrices in $\mathbb{F}_q^{(mn)\times(mn)}$ that do not decompose as Kronecker products, yet still preserve the rank metric on $\mathcal{C}$, seems a fruitful approach for which there was unfortunately never time these past four years.

We may therefore draw the following conclusion for this part of the thesis: post-quantum cryptography based on isometries in the rank metric has been fruitful in recent years, but to reach a brighter future more fundamental research in coding theory is needed to achieve competitive schemes in terms of speed and size.

BIBLIOGRAPHY

[1] Marius A. Aardal, Gora Adj, Arwa Alblooshi, Diego F. Aranha, Isaac Canales-Martínez, Jorge Chávez-Saab, Décio Luiz Gazzoni Filho, Krijn Reijnders, and Francisco Rodríguez-Henríquez. "Optimized One-Dimensional SQIsign Verification on Intel and Cortex-M4". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025.1 (2025), pp. 1–25 (cit. on pp. xviii, 229, 473).

[2] Gora Adj, Jesús-Javier Chi-Domínguez, Víctor Mateu, and Francisco Rodríguez-Henríquez. "Faulty isogenies: a new kind of leakage". In: *CoRR* abs/2202.04896 (2022). arXiv: 2202.04896. URL: https://arxiv.org/abs/2202.04896 (cit. on pp. 53, 54, 70, 71, 77, 78).

[3] Gora Adj, Jesús-Javier Chi-Domínguez, and Francisco Rodríguez-Henríquez. "Karatsuba-based square-root Vélu's formulas applied to two isogeny-based protocols". In: *J. Cryptogr. Eng.* 13.1 (2023), pp. 89–106. DOI: 10.1007/S13389-022-00293-Y. URL: https://doi.org/10.1007/s13389-022-00293-y (cit. on pp. 106, 107, 131, 221).

[4] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and Jurjen Bos. *HQC*. NIST PQC Submission. 2020 (cit. on p. 354).

[5] Carlos Aguilar-Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. "The return of the SDitH". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 564–596 (cit. on pp. 374, 377, 399).

[6] Toru Akishita and Tsuyoshi Takagi. "Zero-Value Point Attacks on Elliptic Curve Cryptosystem". In: *Information Security, 6th International Conference, ISC 2003, Bristol, UK, October 1-3, 2003, Proceedings*. Ed. by Colin Boyd and Wenbo Mao. Vol. 2851. Lecture Notes in Computer Science. Springer, 2003, pp. 218–233. URL: https://doi.org/10.1007/10958513%5C_17 (cit. on p. 53).

[7] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. "Cryptographic group actions and applications". In: *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer. 2020, pp. 411–439 (cit. on pp. 16, 18).

[8] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. *Classic McEliece*. NIST PQC Submission. 2020 (cit. on pp. 354, 370).

[9] Gianira N Alfarano, Francisco Javier Lobillo, Alessandro Neri, and Antonia Wachter-Zeh. "Sum-rank product codes and bounds on the minimum distance". In: *Finite Fields and Their Applications* 80 (2022), p. 102013 (cit. on p. 334).

[10] Tom M. Apostol. "Resultants of cyclotomic polynomials". In: *Proceedings of the American Mathematical Society* 24.3 (1970), pp. 457–462 (cit. on p. 200).

[11] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zémor, Valentin Vasseur, and Santosh Ghosh. *BIKE*. NIST PQC Submission. 2020 (cit. on p. 354).

[12] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zemor, Carlos Aguilar Melchor, Slim Bettaieb, Loic Bidoux, Magali Bardet, and Ayoub Otmani. *ROLLO (Rank-Ouroboros, LAKE and LOCKER)*. 2019. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/%20round-2-submissions (cit. on p. 318).

[13] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, and Gilles Zémor. "Low Rank Parity Check Codes: New Decoding Algorithms and Applications to Cryptography". In: *IEEE Transactions on Information Theory* 65 (2019), pp. 7697–7717 (cit. on p. 318).

[14] V. Arlazarov, E. Dinic, M. Kronrod, and I. Faradžev. "On Economical Construction of the Transitive Closure of an Oriented Graph". In: *Doklady Akademii Nauk*. Vol. 194. Russian Academy of Sciences. 1970, pp. 487–488 (cit. on p. 369).

[15] Roland Auer and Jaap Top. "Legendre Elliptic Curves over Finite Fields". In: *Journal of Number Theory* 95.2 (2002), pp. 303–312. ISSN: 0022-314X. DOI: https://doi.org/10.1006/jnth.2001.2760. URL: https://www.sciencedirect.com/science/article/pii/S0022314X0192760X (cit. on pp. 211, 239).

[16] Reza Azarderakhsh, David Jao, and Christopher Leonardi. "Post-Quantum Static-Static Key Agreement Using Multiple Protocol Instances". In: 2017, pp. 45–63. DOI: 10.1007/978-3-319-72565-9_3 (cit. on p. 131).

[17] Jean-Claude Bajard and Sylvain Duquesne. "Montgomery-friendly primes and applications to cryptography". In: 11.4 (Nov. 2021), pp. 399–415. DOI: 10.1007/s13389-021-00260-z (cit. on pp. 143, 144).

[18] Jean-Claude Bajard and Sylvain Duquesne. "Montgomery-friendly primes and applications to cryptography". In: *J. Cryptogr. Eng.* 11.4 (2021), pp. 399–415. DOI: 10.1007/s13389-021-00260-z. URL: https://doi.org/10.1007/s13389-021-00260-z (cit. on p. 221).

[19] Marco Baldi, Alessandro Barenghi, Luke Beckwith, Jean-François Biasse, Andre Esser, Kriss Gaj, Kamyar Mohajerani, Gerardo Pelosi, Edoardo Persichetti, Markku-Juhani O. Saarinen, Paolo Santini, and Robert Wallace. *Linear Equivalence Signature Scheme*. https://www.less-project.com/LESS-2023-08-18.pdf. Accessed: 2023-09-15. 2023 (cit. on pp. 403, 409–411).

[20] Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, Tung Chou, Tanja Lange, Michael Meyer, Benjamin Smith, and Jana Sotáková. "CTIDH: faster constant-time CSIDH". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.4 (2021), pp. 351–387. DOI: 10.46586/tches.v2021.i4.351-387. URL: https://doi.org/10.46586/tches.v2021.i4.351-387 (cit. on pp. 30, 50, 53, 54, 65, 67, 82, 84, 86, 87, 99, 103, 118, 125, 128, 129, 131, 133, 136, 137, 141, 152, 168, 172, 175, 179, 224).

[21] Gustavo Banegas, Thomas Debris-Alazard, Milena Nedeljković, and Benjamin Smith. *Wavelet: Code-based postquantum signatures with fast verification on microcontrollers*. Cryptology ePrint Archive, Paper 2021/1432. 2021. URL: https://eprint.iacr.org/2021/1432 (cit. on pp. 374, 377).

[22] Gustavo Banegas, Valerie Gilchrist, Anaëlle Le Dévéhat, and Benjamin Smith. "Fast and Frobenius: Rational Isogeny Evaluation over Finite Fields". In: *LATINCRYPT 2023 - 8th International Conference on Cryptology and Information Security in Latin America*. Springer. 2023, pp. 129–148 (cit. on p. 196).

[23] Gustavo Banegas, Valerie Gilchrist, and Benjamin Smith. "Efficient supersingularity testing over $\mathbb{F}_p$ and CSIDH key validation". In: *Mathematical Cryptology* 2.1 (2022), pp. 21–35 (cit. on pp. 152, 168, 170, 173).

[24] Gustavo Banegas, Valerie Gilchrist, and Benjamin Smith. *Efficient supersingularity testing over $\mathbb{F}_p$ and CSIDH key validation*. Cryptology ePrint Archive, Report 2022/880. 2022. URL: https://eprint.iacr.org/2022/880 (cit. on p. 139).

[25] Gustavo Banegas, Valerie Gilchrist, and Benjamin Smith. "Efficient supersingularity testing over GF(p) and CSIDH key validation". In: *Mathematical Cryptology* 2.1 (2022), pp. 21–35. URL: https://journals.flvc.org/mathcryptology/article/view/132125 (cit. on p. 117).

[26] Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska. "Disorientation faults in CSIDH". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 310–342 (cit. on pp. xvi, 81, 473).

[27] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. "Improvements of Algebraic Attacks for Solving the Rank Decoding and MinRank Problems". In: *ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12491. LNCS. Springer, 2020, pp. 507–536. DOI: 10.1007/978-3-030-64837-4_17. URL: https://doi.org/10.1007/978-3-030-64837-4%5C_17 (cit. on pp. 355, 364, 366).

[28] Alessandro Barenghi, Jean-Francois Biasse, Edoardo Persichetti, and Paolo Santini. *On the Computational Hardness of the Code Equivalence Problem in Cryptography*. Cryptology ePrint Archive, Paper 2022/967. 2022. URL: https://eprint.iacr.org/2022/967 (cit. on p. 322).

[29] Alessandro Barenghi, Jean-François Biasse, Tran Ngo, Edoardo Persichetti, and Paolo Santini. "Advanced signature functionalities from the code equivalence problem". In: *Int. J. Comput. Math. Comput. Syst. Theory* 7.2 (2022), pp. 112–128 (cit. on p. 354).

[30] Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. "LESS-FM: fine-tuning signatures from the code equivalence problem". In: *Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12*. Ed. by Jung Hee Cheon and Jean-Pierre Tillich. Vol. 12841. LNCS. Springer. Springer, 2021, pp. 23–43 (cit. on pp. 315, 316, 318, 322, 333, 354, 355, 358, 360, 374, 377, 380, 387, 403, 409).

[31] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. "Efficient Implementation of Pairing-Based Cryptosystems". In: 17.4 (Sept. 2004), pp. 321–334. DOI: 10.1007/s00145-004-0311-z (cit. on p. 156).

[32] Paulo SLM Barreto, Hae Y Kim, Ben Lynn, and Michael Scott. "Efficient algorithms for pairing-based cryptosystems". In: *Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings 22*. Springer. 2002, pp. 354–369 (cit. on pp. 156, 157, 162, 276).

[33] Paul Barrett. "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor". In: *CRYPTO' 86*. Ed. by Andrew M. Odlyzko. Springer Berlin Heidelberg, 1987, pp. 311–323 (cit. on p. 369).

[34] Andrea Basso, Luca De Feo, Pierrick Dartois, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. *SQIsign2D-West: The Fast, the Small, and the Safer*. Cryptology ePrint Archive, Paper 2024/760. 2024. URL: https://eprint.iacr.org/2024/760 (cit. on pp. 37, 180, 184, 230–232, 238, 250, 251, 254, 257, 303).

[35] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. "Horizontal Collision Correlation Attack on Elliptic Curves". In: *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*. Ed. by Tanja Lange, Kristin E. Lauter, and Petr Lisonek. Vol. 8282. Lecture Notes in Computer Science. Springer, 2013, pp. 553–570. URL: https://doi.org/10.1007/978-3-662-43414-7%5C_28 (cit. on p. 53).

[36] Genrich R. Belitskii, Vyacheslav Futorny, Mikhail Muzychuk, and Vladimir V. Sergeichuk. "Congruence of matrix spaces, matrix tuples, and multilinear maps". In: *Linear Algebra and its Applications* 609 (2021), pp. 317–331. ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2020.09.018. URL: https://www.sciencedirect.com/science/article/pii/S0024379520304407 (cit. on pp. 319, 329).

[37] Emanuele Bellini, Florian Caullery, Philippe Gaborit, Marcos Manzano, and Víctor Mateu. "Improved Veron Identification and Signature Schemes in the Rank Metric". In: *2019 IEEE International Symposium on Information Theory (ISIT)* (2019), pp. 1872–1876 (cit. on p. 318).

[38] Thierry P. Berger. "Isometries for rank distance and permutation group of Gabidulin codes". In: *IEEE Trans. Inf. Theory* 49 (2003), pp. 3016–3019 (cit. on pp. 45, 315, 318).

[39] Daniel J Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Peter Schwabe. "Kummer strikes back: new DH speed records". In: *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014. Proceedings, Part I 20*. Springer. 2014, pp. 317–337 (cit. on pp. 243, 255, 257, 264, 303).

[40] Daniel J Bernstein and Tanja Lange. "Hyper-and-elliptic-curve cryptography". In: *LMS Journal of computation and Mathematics* 17.A (2014), pp. 181–202 (cit. on p. 273).

[41] Daniel J. Bernstein. *Differential addition chains*. 2006. URL: http://cr.yp.to/ecdh/diffchain-20060219.pdf (cit. on p. 223).

[42] Daniel J. Bernstein. "Elliptic vs. hyperelliptic, part 1". In: (2006). URL: http://cr.yp.to/talks.html#2006.09.20 (cit. on pp. 262, 264).

[43] Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. "Faster computation of isogenies of large prime degree". In: *Open Book Series* 4.1 (2020), pp. 39–55 (cit. on p. 26).

[44] Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. "Faster computation of isogenies of large prime degree". In: *CoRR* abs/2003.10118 (2020). Ed. by Steven D. Galbraith. https://msp.org/obs/2020/4-1/obs-v4-n1-p04-p.pdf, pp. 39–55. DOI: 10.2140/obs.2020.4.39. arXiv: 2003.10118. URL: https://arxiv.org/abs/2003.10118 (cit. on pp. 49, 55, 85, 131, 135).

[45] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. "Elligator: elliptic-curve points indistinguishable from uniform random strings". In: *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM, 2013, pp. 967–980. DOI: 10.1145/2508859.2516734. URL: https://ia.cr/2013/325 (cit. on pp. 85, 88).

[46] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. "Elligator: elliptic-curve points indistinguishable from uniform random strings". In: 2013, pp. 967–980. DOI: 10.1145/2508859.2516734 (cit. on p. 170).

[47] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. "The SPHINCS+ Signature Framework". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 2129–2146. ISBN: 9781450367479. DOI: 10.1145/3319535.3363229 (cit. on p. 254).

[48] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. "Quantum Circuits for the CSIDH: Optimizing Quantum Evaluation of Isogenies". In: 2019, pp. 409–441. DOI: 10.1007/978-3-030-17656-3_15 (cit. on pp. 129, 131).

[49] 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 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11477. Lecture Notes in Computer Science. Springer, 2019, pp. 409–441. DOI: 10.1007/978-3-030-17656-3_15. URL: https://doi.org/10.1007/978-3-030-17656-3%5C_15 (cit. on pp. 30, 49, 56, 82, 84, 86, 179).

[50] Daniel J. Bernstein and Bo-Yin Yang. "Fast constant-time gcd computation and modular inversion". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.3 (2019), pp. 340–398 (cit. on p. 247).

[51] Ward Beullens. "Not enough LESS: An improved algorithm for solving code equivalence problems over $\mathbb{F}_q$". In: *International Conference on Selected Areas in Cryptography*. Ed. by Orr Dunkelman, Michael J. Jacobson, and Colin O'Flynn. Vol. 12804. LNCS. https://ia.cr/2020/801. Springer. Springer, 2020, pp. 387–403 (cit. on pp. 318, 366).

[52] Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jhih Shih, and Bo-Yin Yang. *Oil and Vinegar: Modern Parameters and Implementations*. Cryptology ePrint Archive, Paper 2023/059. 2023. URL: https://eprint.iacr.org/2023/059 (cit. on pp. 360, 374, 377).

[53] Ward Beullens, Shuichi Katsumata, and Federico Pintore. "Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices". In: *ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer, 2020, pp. 464–492 (cit. on pp. 316, 354, 359, 360, 380, 388, 389, 391).

[54] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. "CSI-FiSh: Efficient Isogeny Based Signatures Through Class Group Computations". In: *ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11921. LNCS. Springer, 2019, pp. 227–247. DOI: 10.1007/978-3-030-34578-5_9. URL: https://ia.cr/2019/498 (cit. on pp. 29, 49, 82, 179, 180, 358, 360, 380, 401, 409, 416, 417).

[55] Ward Beullens and Bart Preneel. "Field Lifting for Smaller UOV Public Keys". In: *Progress in Cryptology – INDOCRYPT 2017*. Ed. by Arpita Patra and Nigel P. Smart. Cham: Springer International Publishing, 2017, pp. 227–246. ISBN: 978-3-319-71667-1 (cit. on p. 318).

[56] Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. "LESS is more: code-based signatures without syndromes". In: *Progress in Cryptology-AFRICACRYPT 2020: 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20–22, 2020, Proceedings 12*. Springer. 2020, pp. 45–65 (cit. on pp. 315, 318, 322, 354, 380).

[57] Joseph Birr-Pixton. *A modern TLS library in Rust*. https://github.com/rustls/rustls (accessed 2021-12-20) (cit. on pp. 130, 146).

[58] Markus Blaser, Dung Hoang Duong, Anand Kumar Narayanan, Thomas Plantard, Youming Qiao, Arnaud Sipasseuth, and Gang Tang. *ALTEQ Signature Specification*. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/ALTEQ-Spec-web.pdf. Accessed: 2023-10-11. 2023 (cit. on pp. 403, 409, 413, 414).

[59] Dan Boneh and Brent Waters. "Constrained Pseudorandom Functions and Their Applications". In: *Advances in Cryptology - ASIACRYPT 2013*. Ed. by Kazue Sako and Palash Sarkar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 280–300. ISBN: 978-3-642-42045-0 (cit. on pp. 360, 389).

[60] Xavier Bonnetain and André Schrottenloher. "Quantum Security Analysis of CSIDH". In: *EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. LNCS. Springer, 2020, pp. 493–522. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_17. URL: https://doi.org/10.1007/978-3-030-45724-2%5C_17 (cit. on pp. 30, 49, 84, 128).

[61] Giacomo Borin, Edoardo Persichetti, Federico Pintore, Krijn Reijnders, and Paolo Santini. "A Guide to the Design of Digital Signatures based on Cryptographic Group Actions". In: *Journal of Cryptology* **TODO:** fix.**TODO:** fix (2024), **TODO:** fix (cit. on pp. xx, 379, 390, 395, 474).

[62] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM". In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2018, pp. 353–367 (cit. on p. 254).

[63] Joppe W Bos, Craig Costello, Huseyin Hisil, and Kristin Lauter. "Fast cryptography in genus 2". In: *Journal of Cryptology* 29 (2016), pp. 28–60 (cit. on pp. 257, 262, 264, 265, 286).

[64] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. "Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem". In: *2015 IEEE Symposium on Security and Privacy*. 2015, pp. 553–570. DOI: 10.1109/SP.2015.40 (cit. on p. 9).

[65] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. "Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem". In: 2015, pp. 553–570. DOI: 10.1109/SP.2015.40 (cit. on p. 145).

[66] W. Bosma, J. Cannon, and C. Playoust. "The Magma algebra system. I. The user language". In: *J. Symbolic Comput.* 24.3-4 (1997). Computational algebra and number theory, pp. 235–265. ISSN: 0747-7171. DOI: 10.1006/jsco.1996.0125. URL: http://dx.doi.org/10.1006/jsco.1996.0125 (cit. on pp. 256, 351).

[67] C. Bouillaguet, J.-C. Faugère, P.A. Fouque, and L. Perret. "Practical Cryptanalysis of the Identification Scheme Based on the Isomorphism of Polynomial With One Secret Problem". In: *Public Key Cryptography – PKC 2011*. Vol. 6571. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 441–458 (cit. on pp. 325, 332, 347, 349).

[68] Charles Bouillaguet. "Algorithms for some hard problems and cryptographic attacks against specific cryptographic primitives. (Études d'hypothèses algorithmiques et attaques de primitives cryptographiques)". PhD thesis. Paris Diderot University, France, 2011. URL: https://tel.archives-ouvertes.fr/tel-03630843 (cit. on pp. 320, 325, 339, 347–349, 351, 352).

[69] Charles Bouillaguet, Pierre-Alain Fouque, and Amandine Véber. "Graph-Theoretic Algorithms for the "Isomorphism of Polynomials" Problem". In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 211–227. DOI: 10.1007/978-3-642-38348-9_13. URL: https://doi.org/10.1007/978-3-642-38348-9%5C_13 (cit. on pp. 320, 325, 337, 339, 341, 347, 348, 351).

[70] Matt Braithwaite. *Experimenting with Post-Quantum Cryptography*. Google Online Security Blog. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html. 2016. (Visited on 12/20/2021) (cit. on p. 145).

[71] Gilles Brassard and Moti Yung. "One-way group actions". In: *Conference on the Theory and Application of Cryptography*. Springer. 1990, pp. 94–107 (cit. on pp. 17, 380).

[72] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. "Towards Post-Quantum Security for Signal's X3DH Handshake". In: 2020, pp. 404–430. DOI: 10.1007/978-3-030-81652-0_16 (cit. on p. 145).

[73] Bradley Wayne Brock. *Superspecial curves of genera two and three*. Princeton University, 1993 (cit. on p. 259).

[74] Peter Bruin. "The Tate pairing for abelian varieties over finite fields". In: *Journal de theorie des nombres de Bordeaux* 23.2 (2011), pp. 323–328 (cit. on pp. 38, 278).

[75] Giacomo Bruno, Maria Corte-Real Santos, Craig Costello, Jonathan Komada Eriksen, Michael Meyer, Michael Naehrig, and Bruno Sterner. "Cryptographic Smooth Neighbors". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1439. URL: https://eprint.iacr.org/2022/1439 (cit. on pp. 189, 194).

[76] Fabio Campos, Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Michael Meyer, Krijn Reijnders, Francisco Rodríguez-Henríquez Francisco, Peter Schwabe, and Thom Wiggers. "Optimizations and Practicality of High-Security CSIDH". In: *IACR Communications in Cryptology* 1.1 (2024) (cit. on pp. xvii, 127, 474).

[77] Fabio Campos, Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Michael Meyer, Krijn Reijnders, Francisco Rodríguez-Henríquez, Peter Schwabe, and Thom Wiggers. *On the Practicality of Post-Quantum TLS Using Large-Parameter CSIDH*. Cryptology ePrint Archive, Paper 2023/793. 2023. URL: https://eprint.iacr.org/2023/793 (cit. on p. 173).

[78] Fabio Campos, Matthias J. Kannwischer, Michael Meyer, Hiroshi Onuki, and Marc Stöttinger. "Trouble at the CSIDH: Protecting CSIDH with Dummy-Operations Against Fault Injection Attacks". In: *17th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2020, Milan, Italy, September 13, 2020*. IEEE. IEEE, 2020, pp. 57–65. DOI: 10.1109/FDTC51366.2020.00015. URL: https://doi.org/10.1109/FDTC51366.2020.00015 (cit. on pp. 49, 53, 84, 113–115, 128, 152).

[79] Fabio Campos, Juliane Krämer, and Marcel Müller. "Safe-Error Attacks on SIKE and CSIDH". In: *Security, Privacy, and Applied Cryptography Engineering - 11th International Conference, SPACE 2021, Kolkata, India, December 10-13, 2021, Proceedings*. Ed. by Lejla Batina, Stjepan Picek, and Mainack Mondal. Vol. 13162. Lecture Notes in Computer Science. Springer, 2021, pp. 104–125. DOI: 10.1007/978-3-030-95085-9_6. URL: https://doi.org/10.1007/978-3-030-95085-9%5C_6 (cit. on pp. 53, 84, 114).

[80] Fabio Campos, Michael Meyer, Krijn Reijnders, and Marc Stöttinger. "Patient Zero & Patient Six: Zero-Value and Correlation Attacks on CSIDH and SIKE". In: *International Conference on Selected Areas in Cryptography*. Springer. 2022, pp. 234–262 (cit. on pp. xiv, 51, 473).

[81] Anne Canteaut and Yuval Ishai, eds. *Advances in Cryptology – EUROCRYPT 2020 – 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*. Vol. 12106. Lecture Notes in Computer Science. Springer, 2020. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2.

[82] David G Cantor. "On arithmetical algorithms over finite fields". In: *Journal of Combinatorial Theory, Series A* 50.2 (1989), pp. 285–300 (cit. on p. 23).

[83] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. "GeMSS: A Great Multivariate Short Signature". In: 2017 (cit. on p. 318).

[84] J. W. S. Cassels and E. V. Flynn. *Prolegomena to a Middlebrow Arithmetic of Curves of Genus 2*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1996 (cit. on pp. 22, 259, 261, 280, 292, 305).

[85] W. Castryck, T. Decru, and B. Smith. "Hash functions from superspecial genus-2 curves using Richelot isogenies". In: *Journal of Mathematical Cryptology* 14.1 (2020), pp. 268–292 (cit. on p. 259).

[86] Wouter Castryck and Thomas Decru. "An Efficient Key Recovery Attack on SIDH". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 423–447. DOI: 10.1007/978-3-031-30589-4_15. URL: https://doi.org/10.1007/978-3-031-30589-4%5C_15 (cit. on pp. 34, 49, 51, 83, 131, 183, 230, 254).

[87] Wouter Castryck and Thomas Decru. "CSIDH on the Surface". In: 2020, pp. 111–129. DOI: 10.1007/978-3-030-44223-1_7 (cit. on p. 131).

[88] Wouter Castryck and Thomas Decru. "CSIDH on the Surface". In: *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*. Ed. by Jintai Ding and Jean-Pierre Tillich. Vol. 12100. Lecture Notes in Computer Science. Springer, 2020, pp. 111–129. DOI: 10.1007/978-3-030-44223-1_7. URL: https://doi.org/10.1007/978-3-030-44223-1%5C_7 (cit. on pp. 49, 78, 121, 125).

[89] Wouter Castryck and Thomas Decru. *Multiradical isogenies*. Cryptology ePrint Archive, Report 2021/1133. https://ia.cr/2021/1133. 2021 (cit. on p. 179).

[90] Wouter Castryck, Thomas Decru, Marc Houben, and Frederik Vercauteren. "Horizontal Racewalking Using Radical Isogenies". In: 2022, pp. 67–96. DOI: 10.1007/978-3-031-22966-4_3 (cit. on p. 131).

[91] Wouter Castryck, Thomas Decru, Marc Houben, and Frederik Vercauteren. "Horizontal racewalking using radical isogenies". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2022, pp. 67–96 (cit. on p. 179).

[92] Wouter Castryck, Thomas Decru, and Frederik Vercauteren. "Radical Isogenies". In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. Lecture Notes in

Computer Science. Springer, 2020, pp. 493–519. DOI: 10.1007/978-3-030-64834-3_17. URL: https://doi.org/10.1007/978-3-030-64834-3%5C_17 (cit. on pp. 49, 50, 79, 120–124, 179).

[93] Wouter Castryck, Marc Houben, Simon-Philipp Merz, Marzio Mula, Sam van Buuren, and Frederik Vercauteren. "Weak instances of class group action based cryptography via self-pairings". In: *Annual International Cryptology Conference*. Springer. 2023, pp. 762–792 (cit. on pp. 38, 39).

[94] Wouter Castryck, Marc Houben, Simon-Philipp Merz, Marzio Mula, Sam van Buuren, and Frederik Vercauteren. "Weak instances of class group action based cryptography via self-pairings". In: *Annual International Cryptology Conference*. Springer. 2023, pp. 762–792 (cit. on p. 278).

[95] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. "CSIDH: An Efficient Post-Quantum Commutative Group Action". In: 2018, pp. 395–427. DOI: 10.1007/978-3-030-03332-3_15 (cit. on pp. 128, 129, 131, 133, 137, 140, 141, 152, 154, 171, 173).

[96] 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: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24*. Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11274. LNCS. Springer. Springer, 2018, pp. 395–427. DOI: 10.1007/978-3-030-03332-3_15. URL: https://doi.org/10.1007/978-3-030-03332-3%5C_15 (cit. on pp. 8, 28, 30, 49, 52, 59, 63, 82, 86–88, 99, 100, 117, 354, 380, 415).

[97] Daniel Cervantes-Vázquez, Mathilde Chenu, Jesús-Javier Chi-Domínguez, Luca De Feo, Francisco Rodríguez-Henríquez, and Benjamin Smith. "Stronger and Faster Side-Channel Protections for CSIDH". In: 2019, pp. 173–193. DOI: 10.1007/978-3-030-30530-7_9 (cit. on pp. 128, 129, 131, 133, 135, 139).

[98] Daniel Cervantes-Vázquez, Mathilde Chenu, Jesús-Javier Chi-Domínguez, Luca De Feo, Francisco Rodríguez-Henríquez, and Benjamin Smith. "Stronger and Faster Side-Channel Protections for CSIDH". In: *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*. Ed. by Peter Schwabe and Nicolas Thériault. Vol. 11774. Lecture Notes in Computer Science. Springer. Springer, 2019, pp. 173–193. DOI: 10.1007/978-3-030-30530-7_9. URL:

https://doi.org/10.1007/978-3-030-30530-7%5C_9 (cit. on pp. 30, 49, 59, 87, 106, 107, 114, 117, 152, 154, 155, 223).

[99] Sanjit Chatterjee, Palash Sarkar, and Rana Barua. "Efficient Computation of Tate Pairing in Projective Coordinate over General Characteristic Fields". In: 2005, pp. 168–181. DOI: 10.1007/11496618_13 (cit. on p. 162).

[100] Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Francisco Rodríguez-Henríquez, Sina Schaeffler, and Benjamin Wesolowski. *SQIsign: Algorithm specifications and supporting documentation*. National Institute of Standards and Technology. 2023. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/sqisign-spec-web.pdf (cit. on pp. 33, 189, 190, 193, 194, 215, 219, 221, 230, 237, 254, 255, 299).

[101] Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, et al. *SQIsign - Submission to the NIST Digital Signature Scheme standardization process (Round 2, 2024)*. 2023 (cit. on pp. 232, 249).

[102] Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. "The SQALE of CSIDH: sublinear Vélu quantum-resistant isogeny action with low exponents". In: *Journal of Cryptographic Engineering* (2021). DOI: 10.1007/s13389-021-00271-w (cit. on pp. 30, 49, 53, 54, 63, 87, 106, 107, 152, 154, 155, 172).

[103] Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. "The SQALE of CSIDH: sublinear Vélu quantum-resistant isogeny action with low exponents". In: 12.3 (Sept. 2022), pp. 349–368. DOI: 10.1007/s13389-021-00271-w (cit. on pp. 29, 128, 129, 131, 133–135, 139, 141, 144, 145).

[104] Mingjie Chen, Antonin Leroux, and Lorenz Panny. "SCALLOP-HD: group action from 2-dimensional isogenies". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2024, pp. 190–216 (cit. on pp. 179, 380, 416).

[105] Jesús-Javier Chi-Domínguez, Amalia Pizarro-Madariaga, and Edgardo Riquelme. *Computing Isogenies of Power-Smooth Degrees Between PPAVs*. Cryptology ePrint Archive, Paper 2023/508. 2023. URL: https://eprint.iacr.org/2023/508 (cit. on p. 299).

[106] Jesús-Javier Chi-Domínguez and Krijn Reijnders. "Fully projective radical isogenies in constant-time". In: *Cryptographers Track at the RSA Conference*. Springer. 2022, pp. 73–95 (cit. on pp. xvi, 119, 473).

[107] Jesús-Javier Chi-Domínguez and Francisco Rodríguez-Henríquez. "Optimal strategies for CSIDH". In: *Adv. Math. Commun.* 16.2 (2022), pp. 383–411. DOI: 10.3934/amc.2020116. URL: https://doi.org/10.3934/amc.2020116 (cit. on pp. 30, 49, 64, 86, 106, 107, 131, 152, 175, 213).

[108] Andrew M. Childs, David Jao, and Vladimir Soukharev. "Constructing elliptic curve isogenies in quantum subexponential time". In: *J. Mathematical Cryptology* 8.1 (2014), pp. 1–29. DOI: 10.1515/jmc-2012-0016. URL: https://arxiv.org/abs/1012.4019 (cit. on p. 84).

[109] Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Lars Ran, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. *MEDS - Submission to the NIST Digital Signature Scheme standardization process (2023)*. 2023 (cit. on pp. 317, 353, 409, 412, 474).

[110] Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. "Take your MEDS: digital signatures from matrix code equivalence". In: *International conference on cryptology in Africa*. Springer. 2023, pp. 28–52 (cit. on pp. xix, 353, 372, 473).

[111] Tung Chou, Ruben Niederhagen, Lars Ran, and Simona Samardjiska. "Reducing Signature Size of Matrix-Code-Based Signature Schemes". In: *International Conference on Post-Quantum Cryptography*. Springer. 2024, pp. 107–134 (cit. on pp. 315, 412, 419).

[112] Tung Chou, Edoardo Persichetti, and Paolo Santini. *On Linear Equivalence, Canonical Forms, and Digital Signatures*. Cryptology ePrint Archive, Paper 2023/1533. 2023. URL: https://eprint.iacr.org/2023/1533 (cit. on pp. 410, 411, 419).

[113] David V Chudnovsky and Gregory V Chudnovsky. "Sequences of numbers generated by addition in formal groups and new primality and factorization tests". In: *Advances in Applied Mathematics* 7.4 (1986), pp. 385–434 (cit. on pp. 257, 262).

[114] Ping Ngai Chung, Craig Costello, and Benjamin Smith. "Fast, uniform scalar multiplication for genus 2 Jacobians with fast Kummers". In: *Selected Areas in Cryptography–SAC 2016: 23rd International Conference, St.*

John's, NL, Canada, August 10-12, 2016, Revised Selected Papers 23. Springer. 2017, pp. 465–481 (cit. on pp. 265, 269, 270).

[115] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. CRC press, 2005 (cit. on p. 38).

[116] John H. Conway and Neil J. A. Sloane. "Low dimensional lattices VII: Coordination sequences". In: *Proceedings of the Royal Society of London, Series A 453*. 1997, pp. 2369–2389 (cit. on p. 108).

[117] 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 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13509. Lecture Notes in Computer Science. Springer, 2022, pp. 285–314. DOI: 10.1007/978-3-031-15982-4_10 (cit. on p. 287).

[118] Maria Corte-Real Santos, Craig Costello, and Benjamin Smith. *Efficient (3,3)-isogenies on fast Kummer surfaces*. Cryptology ePrint Archive, Paper 2024/144. 2024. URL: https://eprint.iacr.org/2024/144 (cit. on p. 35).

[119] Maria Corte-Real Santos, Craig Costello, and Benjamin Smith. "Efficient (3,3)-isogenies on fast Kummer surfaces". In: *arXiv preprint arXiv:2402.01223* (2024) (cit. on pp. 294, 299).

[120] Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Krijn Reijnders. "AprèsSQI: extra fast verification for SQIsign using extension-field signing". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2024, pp. 63–93 (cit. on pp. xvii, 187, 473).

[121] Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Krijn Reijnders. "AprèsSQI: Extra Fast Verification for SQIsign Using Extension-Field Signing". In: *Advances in Cryptology - EUROCRYPT 2024 - 43nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. 2024 (cit. on p. 276).

[122] Maria Corte-Real Santos and Krijn Reijnders. *Return of the Kummer: a Toolbox for Genus-2 Cryptography*. Cryptology ePrint Archive, Paper 2024/948. 2024. URL: https://eprint.iacr.org/2024/948 (cit. on pp. xviii, 253, 474).

[123] Romain Cosset. "Applications of theta functions for hyperelliptic curve cryptography". PhD thesis. Ph. D Thesis, Université Henri Poincaré-Nancy I, 2011 (cit. on p. 265).

[124] Craig Costello. "B-SIDH: Supersingular Isogeny Diffie-Hellman Using Twisted Torsion". In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. Lecture Notes in Computer Science. Springer, 2020, pp. 440–463. DOI: 10.1007/978-3-030-64834-3_15. URL: https://doi.org/10.1007/978-3-030-64834-3%5C_15 (cit. on pp. 189, 194).

[125] Craig Costello. "Computing Supersingular Isogenies on Kummer Surfaces". In: *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*. Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11274. Lecture Notes in Computer Science. Springer, 2018, pp. 428–456. DOI: 10.1007/978-3-030-03332-3_16. URL: https://doi.org/10.1007/978-3-030-03332-3%5C_16 (cit. on pp. 184, 211, 239, 255–257, 269, 272, 273, 290, 295, 296).

[126] Craig Costello. *Pairings for beginners*. 2015. URL: https://www.craigcostello.com.au/ (cit. on pp. 156, 157).

[127] Craig Costello. "The case for SIKE: a decade of the supersingular isogeny problem". In: *Cryptology ePrint Archive* (2021) (cit. on pp. 52, 53).

[128] Craig Costello and Hüseyin Hisil. "A Simple and Compact Algorithm for SIDH with Arbitrary Degree Isogenies". In: *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10625. Lecture Notes in Computer Science. Springer, 2017, pp. 303–329. DOI: 10.1007/978-3-319-70697-9_11. URL: https://doi.org/10.1007/978-3-319-70697-9%5C_11 (cit. on p. 207).

[129] Craig Costello, David Jao, Patrick Longa, Michael Naehrig, Joost Renes, and David Urbanik. "Efficient Compression of SIDH Public Keys". In: 2017, pp. 679–706. DOI: 10.1007/978-3-319-56620-7_24 (cit. on p. 138).

[130] Craig Costello, David Jao, Patrick Longa, Michael Naehrig, Joost Renes, and David Urbanik. "Efficient Compression of SIDH Public Keys". In: *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International*

Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. Lecture Notes in Computer Science. Springer. 2017, pp. 679–706. DOI: 10.1007/978-3-319-56620-7_24 (cit. on pp. 152, 161, 162, 168, 172, 210, 214, 276).

[131] Craig Costello, Patrick Longa, and Michael Naehrig. "Efficient Algorithms for Supersingular Isogeny Diffie-Hellman". In: 2016, pp. 572–601. DOI: 10.1007/978-3-662-53018-4_21 (cit. on pp. 131, 152, 161, 168).

[132] Craig Costello, Patrick Longa, and Michael Naehrig. "Efficient Algorithms for Supersingular Isogeny Diffie-Hellman". In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. Lecture Notes in Computer Science. Springer, 2016, pp. 572–601. DOI: 10.1007/978-3-662-53018-4_21. URL: https://doi.org/10.1007/978-3-662-53018-4%5C_21 (cit. on pp. 53, 71).

[133] Craig Costello, Patrick Longa, Michael Naehrig, Joost Renes, and Fernando Virdia. "Improved classical cryptanalysis of SIKE in practice". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2020, pp. 505–534 (cit. on p. 79).

[134] Craig Costello, Michael Meyer, and Michael Naehrig. "Sieving for Twin Smooth Integers with Solutions to the Prouhet-Tarry-Escott Problem". In: *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12696. Lecture Notes in Computer Science. Springer, 2021, pp. 272–301. DOI: 10.1007/978-3-030-77870-5_10. URL: https://doi.org/10.1007/978-3-030-77870-5%5C_10 (cit. on pp. 189, 194).

[135] Nicolas Courtois, Er Klimov, Jacques Patarin, and Adi Shamir. "Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations". In: *In Advances in Cryptology, Eurocrypt'00, LNCS 1807*. Ed. by Bart Preneel. Vol. 1807. EUROCRYPT'00. Bruges, Belgium: Springer-Verlag, 2000, pp. 392–407. ISBN: 3-540-67517-5 (cit. on p. 364).

[136] Nicolas Tadeusz Courtois. "Efficient zero-knowledge authentication based on a linear algebra problem MinRank". In: *Advances in Cryptology – ASIACRYPT 2001*. Vol. 2248. LNCS. Springer, 2001, pp. 402–421 (cit. on p. 366).

[137] Jean-Marc Couveignes. *Hard Homogeneous Spaces*. Cryptology ePrint Archive, Paper 2006/291. http://eprint.iacr.org/2006/291. 2006. URL: https://ia.cr/2006/291 (cit. on p. 83).

[138] Alain Couvreur, Thomas Debris-Alazard, and Philippe Gaborit. *On the hardness of code equivalence problems in rank metric*. 2021. arXiv: 2011.04611 [cs.IT]. URL: https://arxiv.org/abs/2011.04611 (cit. on pp. 45, 315, 318, 319, 321–323, 332, 333, 336, 354).

[139] Ronald Cramer. "Modular design of secure yet practical cryptographic protocols". In: *Ph. D.-thesis, CWI and U. of Amsterdam* 2 (1996) (cit. on pp. 12, 13).

[140] Ronald Cramer and Victor Shoup. "Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack". In: *SIAM Journal on Computing* 33.1 (2003), pp. 167–226. DOI: 10.1137/S0097539702403773. URL: https://doi.org/10.1137/S0097539702403773 (cit. on pp. 9, 10).

[141] Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. "SQISignHD: new dimensions in cryptography". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2024, pp. 3–32 (cit. on pp. 37, 180, 183, 190, 192, 230, 254).

[142] Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert. *An Algorithmic Approach to (2,2)-isogenies in the Theta Model and Applications to Isogeny-based Cryptography*. Cryptology ePrint Archive, Paper 2023/1747. 2023. URL: https://eprint.iacr.org/2023/1747 (cit. on pp. 254, 255, 257, 292, 296, 299).

[143] Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. "SCALLOP: scaling the CSI-FiSh". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2023, pp. 345–375 (cit. on pp. 8, 179, 380, 409, 416).

[144] Luca De Feo and Steven D. Galbraith. "SeaSign: Compact Isogeny Signatures from Class Group Actions". In: *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. Lecture Notes in Computer Science. Springer, 2019, pp. 759–

789. DOI: 10.1007/978-3-030-17659-4_26. URL: https://ia.cr/2018/824 (cit. on pp. 49, 82, 180, 316, 318, 354, 358–360, 380, 385, 416).

[145] Luca De Feo, David Jao, and Jérôme Plût. "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies". In: *J. Math. Cryptol.* 8.3 (2014), pp. 209–247. DOI: 10.1515/jmc-2012-0015. URL: https://doi.org/10.1515/jmc-2012-0015 (cit. on p. 49).

[146] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. "SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies". In: 2020, pp. 64–93. DOI: 10.1007/978-3-030-64837-4_3 (cit. on pp. 152, 154).

[147] 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. Cham: Springer International Publishing, 2020, pp. 64–93. ISBN: 978-3-030-64837-4 (cit. on pp. 183, 318).

[148] 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 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12491. Lecture Notes in Computer Science. Springer, 2020, pp. 64–93. DOI: 10.1007/978-3-030-64837-4_3. URL: https://doi.org/10.1007/978-3-030-64837-4%5C_3 (cit. on pp. 33, 188–190, 192–194, 214, 215, 230, 234, 237, 254, 255, 299).

[149] Luca De Feo, Antonin Leroux, Patrick Longa, and Benjamin Wesolowski. "New Algorithms for the Deuring Correspondence - Towards Practical and Secure SQISign Signatures". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 659–690. DOI: 10.1007/978-3-031-30589-4_23. URL: https://doi.org/10.1007/978-3-031-30589-4%5C_23 (cit. on pp. 33, 183, 190, 192–194, 197, 202, 215, 230, 235, 237, 254, 255).

[150] Luca De Feo and Michael Meyer. "Threshold Schemes from Isogeny Assumptions". In: *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12111. Lecture Notes in Computer Science. Springer, 2020, pp. 187–212. DOI: 10.1007/978-3-030-45388-6_7. URL: https://ia.cr/2019/1288 (cit. on pp. 49, 82, 180).

[151] Luca De Feo, Nadia El Mrabet, Aymeric Genêt, Novak Kaluderovic, Natacha Linard de Guertechin, Simon Pontié, and Élise Tasso. "SIKE Channels". In: *IACR Cryptol. ePrint Arch.* (2022), p. 54. URL: https://eprint.iacr.org/2022/054 (cit. on pp. 52, 53, 59, 60, 72, 74, 77).

[152] Bor de Kock. "A non-interactive key exchange based on ring-learning with errors". PhD thesis. Master's thesis. Master's thesis, Eindhoven University of Technology, 2018 (cit. on pp. 8, 132).

[153] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. "Wave: A New Family of Trapdoor One-Way Preimage Sampleable Functions Based on Codes". In: *Advances in Cryptology – ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Cham: Springer International Publishing, 2019, pp. 21–51. ISBN: 978-3-030-34578-5 (cit. on p. 318).

[154] Thomas Decru. "Radical Vélu Isogeny Formulae". In: *Annual International Cryptology Conference*. Springer. 2024 (cit. on p. 179).

[155] Thomas Decru, Lorenz Panny, and Frederik Vercauteren. "Faster SeaSign signatures through improved rejection sampling". In: *Post-Quantum Cryptography: 10th International Conference, PQCrypto 2019, Chongqing, China, May 8–10, 2019 Revised Selected Papers 10*. Springer. 2019, pp. 271–285 (cit. on pp. 49, 180, 380, 416).

[156] Christina Delfs and Steven D. Galbraith. "Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$". In: *Des. Codes Cryptography* 78.2 (2016), pp. 425–440. DOI: 10.1007/s10623-014-0010-1. URL: https://arxiv.org/abs/1310.7789 (cit. on p. 84).

[157] Max Deuring. "Die typen der multiplikatorenringe elliptischer funktionenkörper". In: *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*. Vol. 14. 1. Springer Berlin/Heidelberg. 1941, pp. 197–272 (cit. on p. 31).

[158] Whitfield Diffie and Martin E Hellman. "New directions in cryptography". In: *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*. 2022, pp. 365–390 (cit. on p. 6).

[159] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias Kannwischer, and Jacques Patarin. *Rainbow*. Tech. rep. National Institute of Standards and Technology, 2020. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions (cit. on p. 360).

[160] Jintai Ding and Dieter Schmidt. "Rainbow, a New Multivariable Polynomial Signature Scheme". In: *ACNS*. Ed. by John Ioannidis, Angelos D. Keromytis, and Moti Yung. Vol. 3531. Lecture Notes in Computer Science. 2005, pp. 164–175. ISBN: 3-540-26223-7 (cit. on p. 318).

[161] Javad Doliskani. "On division polynomial PIT and supersingularity". In: *Applicable Algebra in Engineering, Communication and Computing* 29.5 (2018), pp. 393–407 (cit. on pp. 139, 170, 173).

[162] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. "Security of the Fiat-Shamir transformation in the quantum random-oracle model". In: *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*. Springer. 2019, pp. 356–383 (cit. on p. 12).

[163] Vivien Dubois, Louis Granboulan, and Jacques Stern. "An efficient provable distinguisher for HFE". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2006, pp. 156–167 (cit. on pp. 324, 340).

[164] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "Crystals-dilithium: A lattice-based digital signature scheme". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 238–268 (cit. on pp. 254, 354, 374, 377).

[165] Léo Ducas and Wessel van Woerden. "On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography". In: *Advances in Cryptology – EUROCRYPT 2022*. Ed. by Orr Dunkelman and Stefan Dziembowski. Cham: Springer International Publishing, 2022, pp. 643–673. ISBN: 978-3-031-07082-2 (cit. on pp. 318, 380).

[166] Max Duparc and Tako Boris Fouotsa. *SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies*. Cryptology ePrint Archive, Paper 2024/773. 2024. URL: https://eprint.iacr.org/2024/773 (cit. on pp. 37, 184, 230, 254, 257, 303).

[167] Andres Erbsen, Jade Philipoom, Jason Gross, Robert Sloan, and Adam Chlipala. "Simple High-Level Code for Cryptographic Arithmetic - With Proofs, Without Compromises". In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 1202–1219. DOI: 10.1109/SP.2019.00005 (cit. on pp. 244, 245).

[168] 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: *IACR Cryptol. ePrint Arch.* (2023), p. 106. URL: https://eprint.iacr.org/2023/106 (cit. on pp. 190, 196).

[169] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. "Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1, 1): Algorithms and complexity". In: *J. Symb. Comput.* 46.4 (2011), pp. 406–437 (cit. on p. 364).

[170] Jean-Charles Faugère, Françoise Levy-dit-Vehel, and Ludovic Perret. "Cryptanalysis of MinRank". In: *CRYPTO*. Ed. by David A. Wagner. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 280–296. ISBN: 978-3-540-85173-8 (cit. on pp. 355, 364, 366).

[171] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. "Folding Alternant and Goppa Codes With Non-Trivial Automorphism Groups". In: *IEEE Trans. Inf. Theory* 62.1 (2016), pp. 184–198. DOI: 10.1109/TIT.2015.2493539. URL: https://doi.org/10.1109/TIT.2015.2493539 (cit. on p. 329).

[172] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric Portzamparc, and Jean-Pierre Tillich. "Structural Cryptanalysis of McEliece Schemes with Compact Keys". In: *Des. Codes Cryptography* 79.1 (Apr. 2016), pp. 87–112. ISSN: 0925-1022. DOI: 10.1007/s10623-015-0036-z. URL: https://doi.org/10.1007/s10623-015-0036-z (cit. on p. 329).

[173] Jean-Charles Faugère and Ludovic Perret. "Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects". In: *EUROCRYPT '06*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. Springer, July 5, 2006, pp. 30–47. ISBN: 3-540-34546-9 (cit. on pp. 324, 325, 337, 339, 347, 348).

[174] Jean-Charles Faugère and Ludovic Perret. "Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects". In: *EUROCRYPT*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, 2006, pp. 30–47. ISBN: 3-540-34546-9 (cit. on p. 363).

[175] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. "Shared permutation for syndrome decoding: New zero-knowledge protocol and code-based signature". In: *Designs, Codes and Cryptography* 91.2 (2023), pp. 563–608 (cit. on p. 380).

[176] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. "Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs". In: *Advances in Cryptology – CRYPTO 2022*. Vol. 13508. LNCS. Springer, 2022, pp. 541–572 (cit. on p. 380).

[177] Luca De Feo, David Jao, and Jérôme Plût. "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies". In: *J. Math. Cryptol.* 8.3 (2014), pp. 209–247. DOI: 10.1515/JMC-2012-0015. URL: https://doi.org/10.1515/jmc-2012-0015 (cit. on pp. 34, 207, 299).

[178] Amos Fiat and Adi Shamir. "How to prove yourself: Practical solutions to identification and signature problems". In: *Conference on the theory and application of cryptographic techniques*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer. Santa Barbara, California, United States: Springer-Verlag, 1986, pp. 186–194 (cit. on pp. 12, 193, 235, 318, 325, 355, 356, 381).

[179] Pierre-Alain Fouque, Louis Granboulan, and Jacques Stern. "Differential Cryptanalysis for Multivariate Schemes". In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 341–353. ISBN: 978-3-540-25910-7 (cit. on p. 324).

[180] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. "Falcon: Fast-Fourier lattice-based compact signatures over NTRU". In: *Submission to the NIST's post-quantum cryptography standardization process* 36.5 (2018), pp. 1–75 (cit. on pp. 254, 354, 374, 377).

[181] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. "Non-Interactive Key Exchange". In: *Public-Key Cryptography – PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Berlin,

Heidelberg: Springer Berlin Heidelberg, 2013, pp. 254–271. ISBN: 978-3-642-36362-7 (cit. on pp. 9, 10).

[182] Vyacheslav Futorny, Joshua A. Grochow, and Vladimir V. Sergeichuk. "Wildness for tensors". In: *Linear Algebra and its Applications* 566 (2019), pp. 212–244. ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2018.12.022. URL: https://www.sciencedirect.com/science/article/pii/S0024379518305937 (cit. on p. 319).

[183] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. *Swoosh: Efficient Lattice-Based Non-Interactive Key Exchange*. 2024 (cit. on pp. 8, 128, 132, 150).

[184] Steven D Galbraith, Florian Hess, and Frederik Vercauteren. "Hyperelliptic pairings". In: *International Conference on Pairing-Based Cryptography*. Springer. 2007, pp. 108–131 (cit. on pp. 38, 282).

[185] Steven D Galbraith and Xibin Lin. "Computing pairings using x-coordinates only". In: *Designs, Codes and Cryptography* 50.3 (2009), pp. 305–324 (cit. on p. 161).

[186] Steven D Galbraith, Christophe Petit, and Javier Silva. "Identification protocols and signature schemes based on supersingular isogeny problems". In: *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer. 2017, pp. 3–33 (cit. on p. 183).

[187] Steven D. Galbraith. *Advances in Elliptic Curve Cryptography, Chapter IX*. Ed. by Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. Vol. 317. Cambridge University Press, 2005 (cit. on pp. 38, 39, 210).

[188] Steven D. Galbraith. *Mathematics of public key cryptography*. Cambridge University Press, 2012 (cit. on pp. 5, 19, 156).

[189] Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. "On the Security of Supersingular Isogeny Cryptosystems". In: 2016, pp. 63–91. DOI: 10.1007/978-3-662-53887-6_3 (cit. on p. 131).

[190] Steven D. Galbraith and Frederik Vercauteren. "Computational problems in supersingular elliptic curve isogenies". In: *Quantum Inf. Process.* 17.10 (2018), p. 265 (cit. on p. 77).

[191] Theodoulos Garefalakis. "The generalized Weil pairing and the discrete logarithm problem on elliptic curves". In: *LATIN 2002: Theoretical Informatics: 5th Latin American Symposium Cancun, Mexico, April 3–6, 2002 Proceedings 5*. Springer. 2002, pp. 118–130 (cit. on p. 38).

[192] Pierrick Gaudry. "Fast genus 2 arithmetic based on Theta functions". In: *Journal of Mathematical Cryptology* 1.3 (2007), pp. 243–265 (cit. on pp. 257, 262, 263, 266, 270, 283, 286, 290, 298, 305).

[193] Alexandre Gélin and Benjamin Wesolowski. "Loop-Abort Faults on Supersingular Isogeny Cryptosystems". In: *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*. Ed. by Tanja Lange and Tsuyoshi Takagi. Vol. 10346. Lecture Notes in Computer Science. Springer, 2017, pp. 93–106. DOI: 10.1007/978-3-319-59879-6_6. URL: https://doi.org/10.1007/978-3-319-59879-6%5C_6 (cit. on pp. 53, 84).

[194] Aymeric Genêt, Natacha Linard de Guertechin, and Novak Kaluderovic. "Full Key Recovery Side-Channel Attack Against Ephemeral SIKE on the Cortex-M4". In: *Constructive Side-Channel Analysis and Secure Design - 12th International Workshop, COSADE 2021, Lugano, Switzerland, October 25-27, 2021, Proceedings*. Ed. by Shivam Bhasin and Fabrizio De Santis. Vol. 12910. Lecture Notes in Computer Science. Springer, 2021, pp. 228–254. URL: https://doi.org/10.1007/978-3-030-89915-8%5C_11 (cit. on p. 53).

[195] Marc Girault. "A (Non-Practical) Three-Pass Identification Protocol Using Coding Theory". In: *Proceedings of the International Conference on Cryptology on Advances in Cryptology*. AUSCRYPT '90. Sydney, Australia: Springer-Verlag, 1990, pp. 265–272. ISBN: 0387530002 (cit. on p. 318).

[196] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems". In: *Journal of the ACM (JACM)* 38.3 (1991), pp. 690–728 (cit. on p. 17).

[197] Ruben Gonzalez and Thom Wiggers. "KEMTLS vs. Post-quantum TLS: Performance on Embedded Systems". In: *Security, Privacy, and Applied Cryptography Engineering*. Ed. by Lejla Batina, Stjepan Picek, and Mainack Mondal. Cham: Springer Nature Switzerland, pp. 99–117. ISBN: 978-3-031-22829-2. DOI: 10.1007/978-3-031-22829-2. URL: https://thomwiggers.nl/publication/kemtls-embedded/ (cit. on p. 149).

[198] Elisa Gorla. "Rank-metric codes". In: *CoRR* abs/1902.02650 (2019). arXiv: 1902.02650. URL: http://arxiv.org/abs/1902.02650 (cit. on pp. 41, 323).

[199] Elisa Gorla and Flavio Salizzoni. "MacWilliams' Extension Theorem for rank-metric codes". In: *Journal of Symbolic Computation* 122 (2024), p. 102263 (cit. on p. 43).

[200] Louis Goubin. "A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems". In: *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*. Ed. by Yvo Desmedt. Vol. 2567. Lecture Notes in Computer Science. Springer, 2003, pp. 199–210. URL: https://doi.org/10.1007/3-540-36288-6%5C_15 (cit. on p. 53).

[201] Torbjörn Granlund and Peter L. Montgomery. "Division by Invariant Integers Using Multiplication". In: *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*. PLDI '94. Orlando, Florida, USA: Association for Computing Machinery, 1994, pp. 61–72 (cit. on p. 369).

[202] Joshua A. Grochow and Youming Qiao. *Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions*. 2019. DOI: 10.48550/ARXIV.1907.00309. URL: https://arxiv.org/abs/1907.00309 (cit. on pp. 315, 318, 319).

[203] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: ACM, 1996, pp. 212–219. ISBN: 0-89791-785-5. DOI: 10.1145/237814.237866 (cit. on p. 367).

[204] Shay Gueron, Edoardo Persichetti, and Paolo Santini. "Designing a Practical Code-Based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup". In: *Cryptography* 6.1 (2022), p. 5 (cit. on pp. 360, 380).

[205] Aurore Guillevic. "Comparing the Pairing Efficiency over Composite-Order and Prime-Order Elliptic Curves". In: 2013, pp. 357–372. DOI: 10.1007/978-3-642-38980-1_22 (cit. on p. 152).

[206] Mike Hamburg. *Computing the Jacobi symbol using Bernstein-Yang*. Cryptology ePrint Archive, Report 2021/1271. 2021. URL: https://eprint.iacr.org/2021/1271 (cit. on p. 140).

[207] Mike Hamburg. *Computing the Jacobi symbol using Bernstein-Yang*. Cryptology ePrint Archive, Paper 2021/1271. 2021. URL: https://eprint.iacr.org/2021/1271 (cit. on p. 248).

[208] Ryuichi Harasawa, Junji Shikata, Joe Suzuki, and Hideki Imai. "Comparing the MOV and FR reductions in elliptic curve cryptography". In: *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic,*

May 2–6, 1999 Proceedings 18. Springer. 1999, pp. 190–205 (cit. on pp. 152, 276).

[209] Robin Hartshorne. *Algebraic geometry*. Vol. 52. Springer Science & Business Media, 2013 (cit. on p. 19).

[210] Ishay Haviv and Oded Regev. "On the lattice isomorphism problem". In: *SODA 2014*. Ed. by Chandra Chekuri. ACM SIAM, 2014, pp. 391–404 (cit. on p. 354).

[211] Andreas Hulsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. *SPHINCS+*. NIST PQC Submission. 2020 (cit. on pp. 354, 374, 377).

[212] Dale Husemöller. *Elliptic Curves, 2nd edition*. Springer, 2004 (cit. on pp. 40, 210, 277).

[213] Aaron Hutchinson, Jason T. LeGrow, Brian Koziel, and Reza Azarderakhsh. "Further Optimizations of CSIDH: A Systematic Approach to Efficient Strategies, Permutations, and Bound Vectors". In: *Applied Cryptography and Network Security – 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part I*. Ed. by Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi. Vol. 12146. Lecture Notes in Computer Science. https://ia.cr/2019/1121. Springer. Springer, 2020, pp. 481–501. DOI: 10.1007/978-3-030-57808-4_24. URL: https://ia.cr/2019/1121 (cit. on pp. 49, 82, 125, 131, 152).

[214] Benjamin Salling Hvass, Diego F. Aranha, and Bas Spitters. "High-Assurance Field Inversion for Curve-Based Cryptography". In: *CSF*. IEEE, 2023, pp. 552–567 (cit. on p. 247).

[215] Jun-ichi Igusa. "Arithmetic variety of moduli for genus two". In: *Annals of Mathematics* 72.3 (1960), pp. 612–649 (cit. on p. 267).

[216] Sorina Ionica and Antoine Joux. "Pairing the volcano". In: *Mathematics of Computation* 82.281 (2013), pp. 581–603 (cit. on p. 312).

[217] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-knowledge from secure multiparty computation". In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 21–30 (cit. on p. 392).

[218] Tetsuya Izu and Tsuyoshi Takagi. "Exceptional Procedure Attack on Elliptic Curve Cryptosystems". In: *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*. Ed. by Yvo Desmedt. Vol. 2567. Lecture Notes in Computer Science. Springer, 2003, pp. 224–239. URL: https://doi.org/10.1007/3-540-36288-6%5C_17 (cit. on p. 53).

[219] David Jacquemin, Anisha Mukherjee, Péter Kutas, and Sujoy Sinha Roy. "Ready to SQI? Safety First! Towards a constant-time implementation of isogeny-based signature, SQIsign". In: *Cryptology ePrint Archive* (2023) (cit. on p. 311).

[220] 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, David Urbanik, and Geovandro Pereira. *SIKE–Supersingular Isogeny Key Encapsulation*. https://sike.org/. 2017 (cit. on p. 52).

[221] 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, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. *SIKE*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions. National Institute of Standards and Technology, 2022 (cit. on pp. 34, 49, 206, 207, 213).

[222] David Jao and Luca De Feo. "Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies". In: 2011, pp. 19–34. DOI: 10.1007/978-3-642-25405-5_2 (cit. on p. 131).

[223] David Jao and Luca De Feo. "Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies". In: *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*. Ed. by Bo-Yin Yang. Vol. 7071. Lecture Notes in Computer Science. Springer, 2011, pp. 19–34. DOI: 10.1007/978-3-642-25405-5_2. URL: https://doi.org/10.1007/978-3-642-25405-5%5C_2 (cit. on p. 52).

[224] Don Johnson, Alfred Menezes, and Scott A. Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)". In: *Int. J. Inf. Sec.* 1.1 (2001), pp. 36–63. DOI: 10.1007/s102070100002. URL: https://doi.org/10.1007/s102070100002 (cit. on p. 188).

[225] Simon Josefsson and Ilari Liusvaara. "Edwards-Curve Digital Signature Algorithm (EdDSA)". In: *RFC* 8032 (2017), pp. 1–60. DOI: 10.17487/RFC8032. URL: https://doi.org/10.17487/RFC8032 (cit. on p. 188).

[226] Antoine Joux. "A one round protocol for tripartite Diffie–Hellman". In: *Journal of cryptology* 17.4 (2004), pp. 263–276 (cit. on pp. 152, 276).

[227] Antoine Joux. "MPC in the head for isomorphisms and group actions". In: *Cryptology ePrint Archive* (2023) (cit. on pp. 316, 380, 392, 394).

[228] Marc Joye and Jean-Jacques Quisquater. "On the Importance of Securing Your Bins: The Garbage-man-in-the-middle Attack". In: 1997, pp. 135–141. DOI: 10.1145/266420.266449 (cit. on p. 158).

[229] Marc Joye and Sung-Ming Yen. "The Montgomery Powering Ladder". In: 2003, pp. 291–302. DOI: 10.1007/3-540-36400-5_22 (cit. on p. 158).

[230] Daniel Kales and Greg Zaverucha. "Improving the performance of the picnic signature scheme". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), pp. 154–188 (cit. on p. 380).

[231] Ernst Kani. "The number of curves of genus two with elliptic differentials." In: (1997) (cit. on pp. 34, 35).

[232] Matthias J. Kannwischer, Markus Krausz, Richard Petri, and Shang-Yi Yang. *pqm4: Benchmarking NIST Additional Post-Quantum Signature Schemes on Microcontrollers*. Cryptology ePrint Archive, Paper 2024/112. 2024. URL: https://eprint.iacr.org/2024/112 (cit. on pp. 232, 246, 248).

[233] Anatolii Karatsuba and Yuri Ofman. "Multiplication of multidigit numbers on automata". In: *Soviet Physics Doklady* (1963). Translated from Doklady Akademii Nauk SSSR, Vol. 145, No. 2, pp. 293–294, July 1962., pp. 595–596 (cit. on pp. 141, 142).

[234] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. "Improved non-interactive zero knowledge with applications to post-quantum signatures". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 525–537 (cit. on p. 380).

[235] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007 (cit. on p. 5).

[236] Eike Kiltz, Daniel Masny, and Jiaxin Pan. "Optimal security proofs for signatures from identification schemes". In: *Annual International Cryptology Conference*. Springer. 2016, pp. 33–61 (cit. on pp. 13, 14).

[237] Aviad Kipnis and Adi Shamir. "Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization". In: *CRYPTO*. Vol. 1666. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1999, pp. 19–30 (cit. on p. 366).

[238] Yutaro Kiyomura and Tsuyoshi Takagi. "Efficient algorithm for Tate pairing of composite order". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 97.10 (2014), pp. 2055–2063 (cit. on pp. 163, 166).

[239] Tetsutaro Kobayashi, Kazumaro Aoki, and Hideki Imai. "Efficient algorithms for Tate pairing". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 89.1 (2006), pp. 134–143 (cit. on pp. 163, 166).

[240] Neal Koblitz. "Elliptic curve cryptosystems". In: *Mathematics of computation* 48.177 (1987), pp. 203–209 (cit. on p. 20).

[241] Neal Koblitz. "Hyperelliptic cryptosystems". In: *Journal of cryptology* 1 (1989), pp. 139–150 (cit. on p. 21).

[242] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. "On the quaternion-isogeny path problem". In: *LMS Journal of Computation and Mathematics* 17.A (2014), pp. 418–432 (cit. on pp. 32, 183).

[243] Dmitrii Koshelev. "Subgroup membership testing on elliptic curves via the Tate pairing". In: *Journal of Cryptographic Engineering* 13.1 (2023), pp. 125–128 (cit. on p. 277).

[244] Brian Koziel, Reza Azarderakhsh, and David Jao. "Side-Channel Attacks on Quantum-Resistant Supersingular Isogeny Diffie-Hellman". In: *Selected Areas in Cryptography - SAC 2017 - 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers*. Ed. by Carlisle Adams and Jan Camenisch. Vol. 10719. Lecture Notes in Computer Science. Springer, 2017, pp. 64–81. URL: https://doi.org/10.1007/978-3-319-72565-9%5C_4 (cit. on pp. 52–54, 78, 79).

[245] Hugo Krawczyk and Hoeteck Wee. "The OPTLS Protocol and TLS 1.3". In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 81–96. DOI: 10.1109/EuroSP.2016.18 (cit. on pp. 128, 130, 145).

[246] Wouter Kuhnen. "OPTLS revisited". https://www.ru.nl/publish/pages/769526/thesis-final.pdf. MA thesis. Radboud University, 2018 (cit. on p. 145).

[247] Greg Kuperberg. "A subexponential-time quantum algorithm for the dihedral hidden subgroup problem". In: *SIAM Journal on Computing* 35.1 (2005), pp. 170–188 (cit. on p. 29).

[248] Greg Kuperberg. "Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem". In: *TQC 2013*. Ed. by Simone Severini and Fernando G. S. L. Brandão. Vol. 22. LIPIcs. Schloss Dagstuhl, 2013, pp. 20–34 (cit. on pp. 29, 46).

[249] Kris Kwiatkowski and Luke Valenta. *The TLS Post-Quantum Experiment*. Cloudflare blog. https://blog.cloudflare.com/the-tls-post-quantum-experiment/. 2019. (Visited on 01/06/2022) (cit. on p. 145).

[250] Yi-Fu Lai, Steven D. Galbraith, and Cyprien Delpech de Saint Guilhem. "Compact, Efficient and UC-Secure Isogeny-Based Oblivious Transfer". In: *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12696. Lecture Notes in Computer Science. Springer, 2021, pp. 213–241. DOI: 10.1007/978-3-030-77870-5_8. URL: https://ia.cr/2020/1012 (cit. on pp. 49, 82, 180).

[251] Georg Landsberg. "Ueber eine Anzahlbestimmung und eine damit zusammenhängende Reihe." In: (1893) (cit. on pp. 341, 367).

[252] Adam Langley. *CECPQ2*. ImperialViolet blog. https://www.imperialviolet.org/2018/12/12/cecpq2.html. 2018. (Visited on 12/20/2021) (cit. on p. 145).

[253] Younho Lee, Il-Hee Kim, and Yongsu Park. "Improved multi-precision squaring for low-end RISC microcontrollers". In: *J. Syst. Softw.* 86.1 (2013), pp. 60–71. DOI: 10.1016/j.jss.2012.06.074. URL: https://doi.org/10.1016/j.jss.2012.06.074 (cit. on p. 142).

[254] Jason LeGrow and Aaron Hutchinson. *An Analysis of Fault Attacks on CSIDH*. Cryptology ePrint Archive, Report 2020/1006. 2020. URL: https://eprint.iacr.org/2020/1006 (cit. on p. 128).

[255] Jason T. LeGrow and Aaron Hutchinson. "(Short Paper) Analysis of a Strong Fault Attack on Static/Ephemeral CSIDH". In: *Advances in Information and Computer Security - 16th International Workshop on Security, IWSEC 2021, Virtual Event, September 8-10, 2021, Proceedings*. Ed. by Toru Nakanishi and Ryo Nojima. Vol. 12835. Lecture Notes in Computer Science. Springer. Springer, 2021, pp. 216–226. DOI: 10.1007/978-3-030-

85987-9_12. URL: https://doi.org/10.1007/978-3-030-85987-9%5C_12 (cit. on pp. 49, 53, 78, 84, 113, 152).

[256] Jeffrey Leon. "Computing automorphism groups of error-correcting codes". In: *IEEE Transactions on Information Theory* 28.3 (1982), pp. 496–511 (cit. on pp. 318, 355, 363, 366).

[257] Ke-Zheng Li and Frans Oort. *Moduli of supersingular abelian varieties*. Vol. 1680. Springer Science & Business Media, 1998 (cit. on p. 259).

[258] Kaizhan Lin, Weize Wang, Zheng Xu, and Chang-An Zhao. "A faster software implementation of SQISign". In: *IEEE Transactions on Information Theory* (2024) (cit. on pp. 154, 183, 190, 210–212, 214, 218, 221, 230, 276, 417).

[259] Patrick Longa. "Efficient Algorithms for Large Prime Characteristic Fields and Their Application to Bilinear Pairings". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023 (2023), pp. 445–472. DOI: 10.46586/tches.v2023.i3.445-472. URL: https://doi.org/10.46586/tches.v2023.i3.445-472 (cit. on p. 144).

[260] David Lubicz and Damien Robert. "A generalisation of Miller's algorithm and applications to pairing computations on abelian varieties". In: *Journal of Symbolic Computation* 67 (2015), pp. 68–92 (cit. on pp. 157, 175).

[261] David Lubicz and Damien Robert. "Arithmetic on abelian and Kummer varieties". In: *Finite Fields Appl.* 39 (2016), pp. 130–158. ISSN: 1071-5797,1090-2465. DOI: 10.1016/j.ffa.2016.01.009. URL: https://doi.org/10.1016/j.ffa.2016.01.009 (cit. on p. 257).

[262] David Lubicz and Damien Robert. "Efficient pairing computation with theta functions". In: *International Algorithmic Number Theory Symposium*. Springer. 2010, pp. 251–269 (cit. on pp. 157, 175, 257, 283).

[263] Vadim Lyubashevsky. *Converting NewHope/LWE key exchange to a Diffe-Hellman-like algorithm*. Crypto Stack Exchange. [Online:] https://crypto.stackexchange.com/questions/48146/converting-newhope-lwe-key-exchange-to-a-diffe-hellman-like-algorithm. 2017. URL: https://crypto.stackexchange.com/questions/48146/converting-newhope-lwe-key-exchange-to-a-diffe-hellman-like-algorithm (visited on 06/10/2017) (cit. on p. 132).

[264] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. *CRYSTALS-DILITHIUM*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022 (cit. on p. 130).

[265] Florence Jessie MacWilliams. "Combinatorial problems of elementary abelian groups". PhD thesis. 1962 (cit. on p. 43).

[266] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. "A Direct Key Recovery Attack on SIDH". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 448–471. DOI: 10.1007/978-3-031-30589-4_16. URL: https://doi.org/10.1007/978-3-031-30589-4%5C_16 (cit. on pp. 34, 49, 51, 83, 131, 183, 230, 254).

[267] Moxie Marlinspike and Trevor Perrin. *The X3DH Key Agreement Protocol*. Signal Specifications. https://signal.org/docs/specifications/x3dh/(accessed2022-01-04). 2016 (cit. on p. 9).

[268] Moxie Marlinspike and Trevor Perrin. *The X3DH Key Agreement Protocol*. Signal Specifications. https://signal.org/docs/specifications/x3dh/(accessed2022-01-04). 2016 (cit. on pp. 128, 145, 149).

[269] R. J. McEliece. "A Public-Key System Based on Algebraic Coding Theory". In: *Jet Propulsion Laboratory, California Institute of Technology* (1978). DSN Progress Report 44, pp. 114–116 (cit. on pp. 318, 354).

[270] Robert J McEliece. *Finite fields for computer scientists and engineers*. Vol. 23. Springer Science & Business Media, 2012 (cit. on p. 158).

[271] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Gilles Zemor, Alain Couvreur, and Adrien Hauteville. *RQC*. 2019. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/%20round-2-submissions (cit. on p. 318).

[272] Ralph C Merkle. "Secure communications over insecure channels". In: *Communications of the ACM* 21.4 (1978), pp. 294–299 (cit. on p. 6).

[273] Michael Meyer, Fabio Campos, and Steffen Reith. "On Lions and Elligators: An Efficient Constant-Time Implementation of CSIDH". In: 2019, pp. 307–325. DOI: 10.1007/978-3-030-25510-7_17 (cit. on pp. 128, 131).

[274] Michael Meyer, Fabio Campos, and Steffen Reith. "On Lions and Elligators: An Efficient Constant-Time Implementation of CSIDH". In: *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*. Ed. by Jintai Ding and Rainer Steinwandt. Vol. 11505. Lecture Notes in Computer Science. Springer. Springer, 2019, pp. 307–325. DOI: 10.1007/978-3-030-25510-7_17. URL: https://doi.org/10.1007/978-3-030-25510-7%5C_17 (cit. on pp. 30, 49, 82, 86, 87, 106, 113, 125, 152, 175).

[275] Michael Meyer and Steffen Reith. "A Faster Way to the CSIDH". In: 2018, pp. 137–152. DOI: 10.1007/978-3-030-05378-9_8 (cit. on p. 131).

[276] Michael Meyer and Steffen Reith. "A Faster Way to the CSIDH". In: *Progress in Cryptology - INDOCRYPT 2018 - 19th International Conference on Cryptology in India, New Delhi, India, December 9-12, 2018, Proceedings*. Ed. by Debrup Chakraborty and Tetsu Iwata. Vol. 11356. Lecture Notes in Computer Science. Springer. Springer, 2018, pp. 137–152. DOI: 10.1007/978-3-030-05378-9_8. URL: https://doi.org/10.1007/978-3-030-05378-9%5C_8 (cit. on pp. 30, 49, 82, 86, 152, 207).

[277] Victor S Miller. "The Weil pairing, and its efficient calculation". In: *Journal of cryptology* 17.4 (2004), pp. 235–261 (cit. on pp. 39, 166).

[278] Victor S Miller. "Use of elliptic curves in cryptography". In: *Conference on the theory and application of cryptographic techniques*. CRYPTO '85. Springer. London, UK: Springer-Verlag, 1985, pp. 417–426. ISBN: 3-540-16463-4 (cit. on p. 20).

[279] Victor S. Miller. "The Weil Pairing, and Its Efficient Calculation". In: 17.4 (Sept. 2004), pp. 235–261. DOI: 10.1007/s00145-004-0315-8 (cit. on pp. 156, 157).

[280] Peter L Montgomery. "Speeding the Pollard and elliptic curve methods of factorization". In: *Mathematics of computation* 48.177 (1987), pp. 243–264 (cit. on pp. 20, 164).

[281] Peter L. Montgomery. "Modular multiplication without trial division". In: *Mathematics of Computation* 44 (1985), pp. 519–521. ISSN: 0025–5718 (cit. on p. 369).

[282] Dustin Moody and Daniel Shumow. "Analogues of Vélu's formulas for isogenies on alternate models of elliptic curves". In: *Math. Comput.* 85.300 (2016), pp. 1929–1951. URL: https://doi.org/10.1090/mcom/3036 (cit. on p. 55).

[283] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. "Correlation-Enhanced Power Analysis Collision Attack". In: *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. Lecture Notes in Computer Science. Springer, 2010, pp. 125–139. URL: https://doi.org/10.1007/978-3-642-15031-9%5C_9 (cit. on pp. 53, 72).

[284] Tomoki Moriya, Hiroshi Onuki, and Tsuyoshi Takagi. "How to Construct CSIDH on Edwards Curves". In: 2020, pp. 512–537. DOI: 10.1007/978-3-030-40186-3_22 (cit. on p. 131).

[285] D. Mumford, C. Musili, M. Nori, E. Previato, M. Stillman, and H. Umemura. *Tata Lectures on Theta II: Jacobian theta functions and differential equations*. Modern Birkhäuser Classics. Birkhäuser Boston, 2012. ISBN: 9780817645786. URL: https://books.google.nl/books?id=xaNCAAAAQBAJ (cit. on p. 22).

[286] Michael Naehrig and Joost Renes. "Dual isogenies and their application to public-key compression for isogeny-based cryptography". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2019, pp. 243–272 (cit. on p. 71).

[287] Kohei Nakagawa and Hiroshi Onuki. "QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras". In: *Cryptology ePrint Archive* (2023) (cit. on pp. 37, 184, 230).

[288] Kohei Nakagawa and Hiroshi Onuki. *SQIsign2D-East: A New Signature Scheme Using 2-dimensional Isogenies*. Cryptology ePrint Archive, Paper 2024/771. 2024. URL: https://eprint.iacr.org/2024/771 (cit. on pp. 37, 180, 184, 230, 254, 257, 303).

[289] Anand Kumar Narayanan, Youming Qiao, and Gang Tang. "Algorithms for Matrix Code and Alternating Trilinear Form Equivalences via New Isomorphism Invariants". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2024, pp. 160–187 (cit. on pp. 315, 317, 353).

[290] Erick Nascimento and Lukasz Chmielewski. "Applying Horizontal Clustering Side-Channel Attacks on Embedded ECC Implementations". In: *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*. Ed. by Thomas Eisenbarth and Yannick Teglia. Vol. 10728.

Lecture Notes in Computer Science. Springer, 2017, pp. 213–231. URL: https://doi.org/10.1007/978-3-319-75208-2%5C_13 (cit. on pp. 54, 72).

[291] National Institute for Standards and Technology. *NIST Workshop on Cybersecurity in a Post-Quantum World*. URL: http://www.nist.gov/itl/csd/ct/post-quantum-crypto-workshop-2015.cfm (cit. on p. 318).

[292] National Institute of Standards and Technology. *Security Requirements for Cryptographic Modules*. Tech. rep. Federal Information Processing Standards Publications (FIPS PUBS) 203, Initial Draft. Washington, D.C.: U.S. Department of Commerce, 2023. DOI: 10.6028/NIST.FIPS.203.ipd (cit. on p. 128).

[293] Alessandro Neri. "Twisted linearized Reed-Solomon codes: A skew polynomial framework". In: *arXiv preprint arXiv:2105.10451* (2021) (cit. on p. 334).

[294] P Nguyen and C Wolf. *International Workshop on Post-Quantum Cryptography*. 2006 (cit. on p. 354).

[295] H. Niederreiter. "Knapsack-type cryptosystems and algebraic coding theory." English. In: *Probl. Control Inf. Theory* 15 (1986), pp. 159–166 (cit. on p. 318).

[296] NIST. *Post-Quantum Cryptography Standardization*. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography. 2017 (cit. on p. 354).

[297] Roberto W Nóbrega and Bartolomeu F Uchôa-Filho. "Multishot codes for network coding using rank-metric codes". In: *2010 Third IEEE International Workshop on Wireless Network Coding*. IEEE. 2010, pp. 1–6 (cit. on p. 333).

[298] Ryo Ohashi. "On the Rosenhain forms of superspecial curves of genus two". In: *arXiv preprint arXiv:2308.11963* (2023) (cit. on p. 290).

[299] Hiroshi Onuki. "On oriented supersingular elliptic curves". In: *Finite Fields and Their Applications* 69 (2021), p. 101777 (cit. on p. 27).

[300] Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi. "(Short Paper) A Faster Constant-Time Algorithm of CSIDH Keeping Two Points". In: 2019, pp. 23–33. DOI: 10.1007/978-3-030-26834-3_2 (cit. on pp. 128, 131).

[301] Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi. "(Short Paper) A Faster Constant-Time Algorithm of CSIDH Keeping Two Points". In: *Advances in Information and Computer Security - 14th International Workshop on Security, IWSEC 2019, Tokyo, Japan, August 28-30, 2019, Proceedings*. Ed. by Nuttapong Attrapadung and Takeshi Yagi. Vol. 11689. Lecture Notes in Computer Science. Springer, 2019, pp. 23–33. DOI: 10.1007/978-3-030-26834-3_2. URL: https://doi.org/10.1007/978-3-030-26834-3%5C_2 (cit. on pp. 30, 49, 87, 113, 114, 125, 145, 152).

[302] Hiroshi Onuki and Tomoki Moriya. "Radical Isogenies on Montgomery Curves". In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 699. URL: https://eprint.iacr.org/2021/699 (cit. on p. 179).

[303] Hiroshi Onuki and Kohei Nakagawa. "Ideal-to-isogeny algorithm using 2-dimensional isogenies and its application to SQIsign". In: *Cryptology ePrint Archive* ***TODO:*** *add asiacrypt for all 2D papers* (2024) (cit. on pp. 184, 231, 237, 251, 311).

[304] Aurel Page and Damien Robert. "Introducing Clapoti (s): Evaluating the isogeny class group action in polynomial time". In: *Cryptology ePrint Archive* (2023) (cit. on pp. 29, 179, 230, 416).

[305] Jacques Patarin. "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms". In: *Advances in Cryptology – EUROCRYPT '96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Saragossa, Spain: Springer-Verlag, 1996, pp. 33–48. ISBN: 3-540-61186-X (cit. on pp. 318, 324, 325, 354).

[306] Jacques Patarin, Louis Goubin, and Nicolas Courtois. "Improved Algorithms for Isomorphisms of Polynomials". In: *EUROCRYPT '98*. Vol. 1403. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 1998, pp. 184–200 (cit. on pp. 321, 325).

[307] Chris Peikert. "He Gives C-Sieves on the CSIDH". In: *Advances in Cryptology – EUROCRYPT 2020 – 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Springer, 2020, pp. 463–492. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_16. URL: https://doi.org/10.1007/978-3-030-45724-2%5C_16 (cit. on pp. 30, 49, 84, 128).

[308] Ray Perlner and Daniel Smith-Tone. *Rainbow Band Separation is Better than we Thought*. Cryptology ePrint Archive, Paper 2020/702. 2020. URL: https://eprint.iacr.org/2020/702 (cit. on pp. 364, 365).

[309] Ludovic Perret. "A Fast Cryptanalysis of the Isomorphism of Polynomials with One Secret Problem". In: *EUROCRYPT*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 354–370. ISBN: 3-540-25910-4 (cit. on pp. 325, 332).

[310] Edoardo Persichetti and Paolo Santini. "A New Formulation of the Linear Equivalence Problem and Shorter LESS Signatures". In: *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VII*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14444. Lecture Notes in Computer Science. Springer, 2023, pp. 351–378. DOI: 10.1007/978-981-99-8739-9_12. URL: https://doi.org/10.1007/978-981-99-8739-9%5C_12 (cit. on pp. 411, 419).

[311] Christiane Peters. "Information-set decoding for linear codes over $\mathbb{F}_q$". In: *International Workshop on Post-Quantum Cryptography*. Springer. 2010, pp. 81–94 (cit. on p. 333).

[312] Christophe Petit and Spike Smith. "An improvement to the quaternion analogue of the l-isogeny problem". In: *Presentation at MathCrypt* (2018) (cit. on p. 33).

[313] Stephen Pohlig and Martin Hellman. "An improved algorithm for computing logarithms over GF (p) and its cryptographic significance (corresp.)" In: *IEEE Transactions on information Theory* 24.1 (1978), pp. 106–110 (cit. on p. 288).

[314] David Pointcheval and Jacques Stern. "Security proofs for signature schemes". In: *International conference on the theory and applications of cryptographic techniques*. Springer. 1996, pp. 387–398 (cit. on p. 12).

[315] Thomas Pornin. *Faster Modular Inversion and Legendre Symbol, and an X25519 Speed Record*. https://research.nccgroup.com/2020/09/28/faster-modular-inversion-and-legendre-symbol-and-an-x25519-speed-record/. 2020 (cit. on p. 247).

[316] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. *FALCON*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-

algorithms-2022. National Institute of Standards and Technology, 2022 (cit. on p. 130).

[317] Valentina Pribanić. "Radical isogenies and modular curves". In: *Advances in Mathematics of Communications* (2023). ISSN: 1930-5346. DOI: 10.3934/amc.2023019. URL: https://www.aimsciences.org/article/id/6486c0aa1778ab0a95207fed (cit. on p. 179).

[318] Lars Ran and Simona Samardjiska. *Rare structures in tensor graphs - Bermuda triangles for cryptosystems based on the Tensor Isomorphism problem*. Cryptology ePrint Archive, Paper 2024/1396. 2024. URL: https://eprint.iacr.org/2024/1396 (cit. on pp. 315, 317, 353).

[319] Dana Randall. *Efficient Generation of Random Nonsingular Matrices*. Tech. rep. UCB/CSD-91-658. EECS Department, UC Berkeley, 1991 (cit. on p. 370).

[320] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Theory of computing*. Ed. by Harold N. Gabow and Ronald Fagin. ACM, 2005, pp. 84–93 (cit. on p. 354).

[321] Krijn Reijnders. "Effective Pairings in Isogeny-based Cryptography". In: *International Conference on Cryptology and Information Security in Latin America*. Springer. 2023, pp. 109–128 (cit. on pp. xvii, 151, 473).

[322] Krijn Reijnders. *Isogenies & Isometries*. 2024. DOI: 10.5281/zenodo.13293882. URL: https://doi.org/10.5281/zenodo.13293882 (cit. on p. xx).

[323] Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. "Hardness estimates of the code equivalence problem in the rank metric". In: *Designs, Codes and Cryptography* 92.3 (2024), pp. 833–862 (cit. on pp. xix, 317, 474).

[324] George W Reitwiesner. "Binary arithmetic". In: *Advances in computers*. Vol. 1. Elsevier, 1960, pp. 231–308 (cit. on p. 163).

[325] Joost Renes, Peter Schwabe, Benjamin Smith, and Lejla Batina. "μKummer: efficient hyperelliptic signatures and key exchange on microcontrollers". In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2016, pp. 301–320 (cit. on pp. 257, 264).

[326] Joost Renes and Benjamin Smith. "qDSA: small and secure digital signatures with curve-based Diffie–Hellman key pairs". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2017, pp. 273–302 (cit. on pp. 224, 257, 286, 287).

[327] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. IETF RFC 8446. https://rfc-editor.org/rfc/rfc8446.txt. 2018 (cit. on p. 145).

[328] Eric Rescorla, Nick Sullivan, and Christopher A. Wood. *Semi-Static Diffie-Hellman Key Establishment for TLS 1.3*. Internet-Draft draft-ietf-tls-semistatic-dh-01. Work in Progress. Internet Engineering Task Force, Mar. 2020. 7 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-tls-semistatic-dh/01/ (cit. on pp. 128, 145).

[329] Damien Robert. "Breaking SIDH in Polynomial Time". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 472–503. DOI: 10.1007/978-3-031-30589-4_17. URL: https://doi.org/10.1007/978-3-031-30589-4%5C_17 (cit. on pp. 34, 49, 51, 83, 131, 183, 230, 254).

[330] Damien Robert. "Evaluating isogenies in polylogarithmic time". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1068. URL: https://eprint.iacr.org/2022/1068 (cit. on pp. 36, 37, 183).

[331] Damien Robert. *Fast pairings via biextensions and cubical arithmetic*. Cryptology ePrint Archive, Paper 2024/517. 2024. URL: https://eprint.iacr.org/2024/517 (cit. on pp. 257, 283, 307, 312).

[332] Damien Robert. "Some notes on algorithms for abelian varieties". In: *IACR Cryptol. ePrint Arch.* (2024), p. 406. URL: https://eprint.iacr.org/2024/406 (cit. on p. 292).

[333] Damien Robert. *The geometric interpretation of the Tate pairing and its applications*. Cryptology ePrint Archive, Paper 2023/177. 2023. URL: https://eprint.iacr.org/2023/177 (cit. on pp. 38, 277–279, 312).

[334] Georg Rosenhain. *Abhandlung über die Functionen zweier Variabler mit vier Perioden: welche die Inversen sind der ultra-elliptischen Integrale erster Klasse*. 65. W. Engelmann, 1895 (cit. on p. 21).

[335] Alexander Rostovtsev and Anton Stolbunov. *PUBLIC-KEY CRYPTOSYSTEM BASED ON ISOGENIES*. Cryptology ePrint Archive, Paper 2006/145. 2006. URL: https://ia.cr/2006/145 (cit. on p. 83).

[336] Tobias Schneider and Amir Moradi. "Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations". In: *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*. Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. Lecture Notes in Computer Science. Springer, 2015, pp. 495–513. URL: https://doi.org/10.1007/978-3-662-48324-4%5C_25 (cit. on pp. 53, 72).

[337] Claus-Peter Schnorr. "Efficient identification and signatures for smart cards". In: *Advances in Cryptology—CRYPTO'89 Proceedings 9*. Springer. 1990, pp. 239–252 (cit. on p. 12).

[338] Jasper Scholten. "Weil restriction of an elliptic curve over a quadratic extension". In: *Preprint* (2003) (cit. on pp. 184, 255, 271, 272).

[339] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "More Efficient Post-quantum KEMTLS with Pre-distributed Public Keys". In: 2021, pp. 3–22. DOI: 10.1007/978-3-030-88418-5_1 (cit. on pp. 146, 147).

[340] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Post-Quantum TLS Without Handshake Signatures". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1461–1480. ISBN: 9781450370899. DOI: 10.1145/3372297.3423350. URL: https://doi.org/10.1145/3372297.3423350 (cit. on pp. 9, 146).

[341] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Post-Quantum TLS Without Handshake Signatures". In: 2020, pp. 1461–1480. DOI: 10.1145/3372297.3423350 (cit. on pp. 130, 146, 147).

[342] Michael Scott. *A note on the calculation of some functions in finite fields: Tricks of the Trade*. Cryptology ePrint Archive, Paper 2020/1497. 2020. URL: https://eprint.iacr.org/2020/1497 (cit. on p. 248).

[343] Michael Scott. "A note on the calculation of some functions in finite fields: Tricks of the trade". In: *Cryptology ePrint Archive* (2020) (cit. on p. 224).

[344] Michael Scott. "Computing the Tate Pairing". In: 2005, pp. 293–304. DOI: 10.1007/978-3-540-30574-3_20 (cit. on p. 162).

[345] Michael Scott. *Elliptic Curve Cryptography for the masses: Simple and fast finite field arithmetic*. Cryptology ePrint Archive, Paper 2024/779. 2024. URL: https://eprint.iacr.org/2024/779 (cit. on pp. 246, 247).

[346] Michael Scott. "Pairing implementation revisited". In: *Cryptology ePrint Archive* (2019) (cit. on p. 165).

[347] Michael Scott. *Understanding the Tate pairing*. 2004. URL: http://www.computing.dcu.ie/~mike/tate.html (cit. on p. 156).

[348] Michael Scott and Paulo S. L. M. Barreto. "Compressed Pairings". In: 2004, pp. 140–156. DOI: 10.1007/978-3-540-28628-8_9 (cit. on pp. 158, 159).

[349] Nicolas Sendrier. "Finding the permutation between equivalent linear codes: The support splitting algorithm". In: *IEEE Trans. Inf. Theory* 46 (2000), pp. 1193–1203 (cit. on p. 318).

[350] V V Sergeichuk. "CLASSIFICATION PROBLEMS FOR SYSTEMS OF FORMS AND LINEAR MAPPINGS". In: *Mathematics of the USSR-Izvestiya* 31.3 (June 1988), pp. 481–501. DOI: 10.1070/im1988v031n03abeh001086. URL: https://doi.org/10.1070/im1988v031n03abeh001086 (cit. on p. 329).

[351] Peter W. Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332 (cit. on pp. xi, 7, 23, 187).

[352] Victor Shoup. "Efficient computation of minimal polynomials in algebraic extensions of finite fields". In: *Proceedings of the 1999 international symposium on Symbolic and algebraic computation*. 1999, pp. 53–58 (cit. on p. 196).

[353] Joseph H Silverman. "A survey of local and global pairings on elliptic curves and abelian varieties". In: *Pairing-Based Cryptography-Pairing 2010: 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings 4*. Springer. 2010, pp. 377–396 (cit. on p. 38).

[354] Joseph H. Silverman. *The arithmetic of elliptic curves*. 2nd ed. Graduate Texts in Mathematics 106. Springer, 2009. ISBN: 978-0-387-09493-9 (cit. on pp. 19, 25).

[355] Joseph H. Silverman. *The arithmetic of elliptic curves*. Vol. 106. Springer, 2009 (cit. on pp. 19, 160).

[356] Benjamin Andrew Smith. "Explicit endomorphisms and correspondences". PhD thesis. 2005 (cit. on pp. 267, 293).

[357] National Institute of Standards and Technology (NIST). *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. 2022. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf (cit. on pp. 188, 230, 254, 315).

[358] Katherine E Stange. "The Tate pairing via elliptic nets". In: *Pairing Based Cryptography*. Springer. 2007, pp. 329–348 (cit. on p. 157).

[359] Anton Stolbunov. "Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves". In: *Adv. in Math. of Comm.* 4.2 (2010), pp. 215–235 (cit. on p. 18).

[360] Anton Stolbunov. "Cryptographic schemes based on isogenies". PhD Thesis. Norges teknisk-naturvitenskapelige universitet, 2012. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/262577/529395_FULLTEXT01.pdf (cit. on p. 415).

[361] Andrew V Sutherland. "Identifying supersingular elliptic curves". In: *LMS Journal of Computation and Mathematics* 15 (2012), pp. 317–325 (cit. on p. 170).

[362] Gang Tang, Dung Hoang Duong, Antoine Joux, Thomas Plantard, Youming Qiao, and Willy Susilo. "Practical Post-Quantum Signature Schemes from Isomorphism Problems of Trilinear Forms". In: *Advances in Cryptology – EUROCRYPT 2022*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. LNCS. Cham: Springer International Publishing, 2022, pp. 582–612. ISBN: 978-3-031-07082-2 (cit. on pp. 316, 318, 354, 370, 380, 403, 413).

[363] Élise Tasso, Luca De Feo, Nadia El Mrabet, and Simon Pontié. "Resistance of Isogeny-Based Cryptographic Implementations to a Fault Attack". In: *Constructive Side-Channel Analysis and Secure Design - 12th International Workshop, COSADE 2021, Lugano, Switzerland, October 25-27, 2021, Proceedings*. Ed. by Shivam Bhasin and Fabrizio De Santis. Vol. 12910. Lecture Notes in Computer Science. Springer, 2021, pp. 255–276. DOI: 10.1007/978-3-030-89915-8_12. URL: https://doi.org/10.1007/978-3-030-89915-8%5C_12 (cit. on pp. 53, 84).

[364] John Tate. "Endomorphisms of abelian varieties over finite fields". In: *Inventiones mathematicae* 2.2 (1966), pp. 134–144 (cit. on p. 24).

[365] "Theorie der quadratischen Formen in beliebigen Korpern". In: *J. Reine Angew. Math.* 176 (1937), pp. 31–44 (cit. on p. 326).

[366] Yan Bo Ti. "Fault Attack on Supersingular Isogeny Cryptosystems". In: *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*. Ed. by Tanja Lange and Tsuyoshi Takagi. Vol. 10346. Lecture Notes in Computer Science. Springer, 2017, pp. 107–122. DOI: 10.1007/978-3-319-59879-6_7. URL: https://doi.org/10.1007/978-3-319-59879-6%5C_7 (cit. on pp. 53, 84).

[367] Kiminori Tsukazaki. "Explicit isogenies of elliptic curves". PhD thesis. University of Warwick, 2013 (cit. on p. 196).

[368] Aleksei Udovenko and Giuseppe Vitto. *Breaking the $IKEp182 Challenge*. IACR Cryptology ePrint Archive 2021/1421. 2021. URL: https://ia.cr/2021/1421 (cit. on p. 112).

[369] "Untersuchungen über quadratische Formen in Korpern der Charakteristik 2, I". In: *J. Reine Angew. Math.* 183 (1941), pp. 148–167 (cit. on p. 326).

[370] Alexei Vambol. *A collection of Elliptic Curves for ZkCrypto traits*. https://github.com/privacy-scaling-explorations/halo2curves/pull/95. 2023 (cit. on p. 247).

[371] Paul C. van Oorschot and Michael J. Wiener. "Parallel Collision Search with Cryptanalytic Applications". In: 12.1 (Jan. 1999), pp. 1–28. DOI: 10.1007/PL00003816 (cit. on p. 134).

[372] Serge Vaudenay. *A Classical Introduction to Cryptography: Applications for Communications Security*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005 (cit. on p. 338).

[373] Jacques Vélu. "Isogénies entre courbes elliptiques". In: *Comptes-Rendus de l'Académie des Sciences* 273 (1971). https://gallica.bnf.fr/ark:/12148/cb34416987n/date, pp. 238–241. URL: https://gallica.bnf.fr/ark:/12148/cb34416987n/date (cit. on pp. 25, 55, 85).

[374] Frederik Vercauteren. "Optimal pairings". In: *IEEE transactions on information theory* 56.1 (2009), pp. 455–461 (cit. on p. 156).

[375] John Voight. *Quaternion algebras*. Springer Nature, 2021 (cit. on p. 19).

[376] Yingchen Wang, Riccardo Paccagnella, Elizabeth Tang He, Hovav Shacham, Christopher W Fletcher, and David Kohlbrenner. "Hertzbleed: Turning power {Side-Channel} attacks into remote timing attacks on x86". In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 679–697 (cit. on pp. 52, 53).

[377] Benjamin Wesolowski. "Orientations and the supersingular endomorphism ring problem". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2022, pp. 345–371 (cit. on p. 416).

[378] Fan Zhang, Bolin Yang, Xiaofei Dong, Sylvain Guilley, Zhe Liu, Wei He, Fangguo Zhang, and Kui Ren. "Side-Channel Analysis and Countermeasure Design on ARM-Based Quantum-Resistant SIKE". In: *IEEE Trans. Computers* 69.11 (2020), pp. 1681–1693. URL: https://doi.org/10.1109/TC.2020.3020407 (cit. on p. 53).

SUMMARY

Cryptography from antiquity up to medieval times is easily broken by pen, paper, a good set of brains, and a bit of spare time. It wasn't until the 19th century that more formal, scientific methods for cryptodesign and cryptoanalysis were developed. Soon, cryptography became mechanical, using intricate devices and clever rotors to secure messages. Anyone will know the example of Enigma. I can only recommend watching The Imitation Game to understand how the development of the modern computer broke such mechanical cryptography. With more advanced computational power, cryptography also became computational: its security now relies on the hardness of mathematical problems, whose difficulty we measure in the amount of computational effort required to solve these.

As brains broke antique cryptography, and computers broke mechanical cryptography, so too will quantum computers break the cryptography we are currently using. And so we venture out into the world of mathematics to search for new hard mathematical problems that quantum computers cannot solve easily. Cryptography based on such problems is called post-quantum cryptography: we can use it on an ordinary computer, laptop, or mobile phone, but we think they resist attacks from quantum computers. This thesis tries to answer questions about three different topics in post-quantum cryptography: 1) CSIDH, 2) SQIsign, and 3) MCE.

Part 1 of this thesis deals with questions about CSIDH, a relatively new cryptographic scheme which seems perfect for use in our daily lives if it wasn't so slow! We analyze how we can mprove the security of CSIDH against physical attacks and try to improve its speed. Ultimately, we find that CSIDH might never be practical unless we all become a bit more patient.

Part 2 of this thesis deals with questions about SQIsign, a brand-new cryptographic scheme, which again seems rather slow. We improve its speed with several new techniques with the hopes of making it practical. These improvements raise many more questions, for which this thesis was too short to answer. One day perhaps, SQIsign might be used by you, unknowingly, when you

browse the web or send an e-mail, if we can improve it quite a bit further.

Part 3 of this thesis deals with questions about MCE, one of these 'new' mathematical problems which we think are hard to solve for a quantum computer. We analyzed the hardness of this problem and designed cryptography based on MCE. We demonstrate the potential for post-quantum cryptography based on MCE, but further research is needed before we can use it in the real world.

And so we find ourselves at the end of this thesis with more questions than answers. But this is good news, as it means that more research into those questions will push the frontier of cryptographic research further and further.

SAMENVATTING

Cryptografie vanaf de oudheid tot aan de middeleeuwen kan eenvoudig ontcijferd worden met alleen een pen, papier, een goed stel hersenen, en een paar uurtjes vrije tijd. Pas in de 19de eeuw werd de cryptografie meer volwassen, met formelere en wetenschappelijkere methodes voor cryptografische ontwerpen en cryptoanalyse. Al gauw werd cryptografie mechanisch en bedacht men behendige apparaten met ingewikkelde tandwielen om berichten te versleutelen. Het voorbeeld Enigma zal voor bijna iedereen bekend zijn. Om te begrijpen hoe de ontwikkeling van de moderne computer bij heeft gedragen aan het breken van mechanische cryptografie, raad ik iedereen aan om The Imitation Game te kijken[10]. Door de verdere ontwikkeling van computers werd ook cryptografie computationeler. De veiligheid van de huidige cryptografie, zoals die iedere dag gebruikt wordt in jouw computer, laptop, of telefoon, is daarmee gebaseerd op de moeilijkheid van onderliggende wiskundige problemen, waarvan we de moeilijkheid meten in de hoeveelheid computationele kracht we nodig hebben om ze op te lossen.

Zoals denkwerk de antieke cryptografie brak, en computers de mechanische cryptografie, zo zal ook de quantum computer onze huidige cryptografie breken. Daarom zoeken we in het uitgestrekte wiskundige landschap naar nieuwe wiskundige problemen, die een quantum computer niet zomaar oplost. Cryptografie gebaseerd op dat soort problemen noemen we post-quantum cryptografie: we kunnen het gebruiken op dezelfde computers, laptops of telefoons als nu, maar we vermoeden (en hopen) dat ze niet eenvoudig opgelost kunnen worden door quantum computers. Dit proefschrift probeert bij te dragen aan de ontwikkeling van post-quantum cryptografie, door onderzoek te doen op drie verschillende thema's: 1) CSIDH, 2) SQIsign, and 3) MCE.

Deel 1 van dit proefschrift behandeld vragen over CSIDH, een vrij nieuw cryptografisch systeem, met veel potentie in ons dagelijks leven als het niet zo traag was! We onderzoeken hoe we CSIDH beter kunnen beveiligen tegen fysieke aanvallen en proberen de snelheid van CSIDH te verbeteren. Uitein-

10 Dit geeft bovendien een goede indruk van hoe het voelt om cryptografisch onderzoek te doen.

delijk concluderen we dat CSIDH mogelijk nooit praktisch zal worden, tenzij we allemaal wat meer geduld zouden hebben.

Deel 2 van dit proefschrift behandeld vragen over SQIsign, een gloednieuw cryptografisch systeem, maar wederom erg traag. Het lukt ons om met verschillende nieuwe technieken de snelheid flink te verbeteren met de hoop om SQIsign praktisch te maken. Deze verbeteringen roepen nog meer vragen op en helaas was het promotie-traject te kort om deze allemaal te beantwoorden. Mogelijk wordt SQIsign ooit op een dag ook gebruikt door jou, onbewust, wanneer je in de bus op je telefoon het nieuws leest of een berichtje verstuurd, als het ons lukt om SQIsign nog heel wat sneller en praktischer te maken.

Deel 3 van dit proefschrift behandeld vragen over MCE, een van deze 'nieuwe' wiskunde problemen waarvan we vermoeden dat ze zelfs moeilijk zijn op te lossen met een quantum computer. We onderzoeken hoe moeilijk het is om dit probleem op te lossen, en ontwikkelen cryptografie gebaseerd op MCE. Dit deel demonstreert de potentie voor post-quantum cryptografie gebaseerd op MCE, maar laat ook zien dat vervolgonderzoek hard nodig is voordat we deze cryptografie kunnen gebruiken in de 'echte wereld'.

En zo bevinden we ons op het eind van dit proefschrift met meer vragen dan antwoorden. Maar, in de onderzoekswereld is dat goed nieuws: het betekent dat we met meer onderzoek naar deze vragen de grenzen van onze cryptografische kennis verder en verder kunnen verleggen, om zo de wereld een klein stukje veiliger te maken.

LIST OF PUBLICATIONS

Conference Proceedings

- Jesús-Javier Chi-Domínguez and Krijn Reijnders. "Fully projective radical isogenies in constant-time". In: *Cryptographers Track at the RSA Conference*. Springer. 2022, pp. 73–95.
- Fabio Campos, Michael Meyer, Krijn Reijnders, and Marc Stöttinger. "Patient Zero & Patient Six: Zero-Value and Correlation Attacks on CSIDH and SIKE". in: *International Conference on Selected Areas in Cryptography*. Springer. 2022, pp. 234–262.
- Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska. "Disorientation faults in CSIDH". in: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 310–342.
- Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. "Take your MEDS: digital signatures from matrix code equivalence". In: *International conference on cryptology in Africa*. Springer. 2023, pp. 28–52.
- Krijn Reijnders. "Effective Pairings in Isogeny-based Cryptography". In: *International Conference on Cryptology and Information Security in Latin America*. Springer. 2023, pp. 109–128.
- Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Krijn Reijnders. "AprèsSQI: extra fast verification for SQIsign using extension-field signing". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2024, pp. 63–93.
- Marius A. Aardal, Gora Adj, Arwa Alblooshi, Diego F. Aranha, Isaac Canales-Martínez, Jorge Chávez-Saab, Décio Luiz Gazzoni Filho, Krijn Reijnders, and Francisco Rodríguez-Henríquez.

"Optimized One-Dimensional SQIsign Verification on Intel and Cortex-M4". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025.1 (2025), pp. 1–25.

Journal Publications

- Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. "Hardness estimates of the code equivalence problem in the rank metric". In: *Designs, Codes and Cryptography* 92.3 (2024), pp. 833–862.
- Fabio Campos, Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Michael Meyer, Krijn Reijnders, Francisco Rodríguez-Henríquez Francisco, Peter Schwabe, and Thom Wiggers. "Optimizations and Practicality of High-Security CSIDH". in: *IACR Communications in Cryptology* 1.1 (2024).
- Giacomo Borin, Edoardo Persichetti, Federico Pintore, Krijn Reijnders, and Paolo Santini. "A Guide to the Design of Digital Signatures based on Cryptographic Group Actions". In: *Journal of Cryptology* **TODO:** fix.**TODO:** fix (2024), **TODO:** fix.

Under Submission

- Maria Corte-Real Santos and Krijn Reijnders. *Return of the Kummer: a Toolbox for Genus-2 Cryptography*. Cryptology ePrint Archive, Paper 2024/948. 2024. URL: https://eprint.iacr.org/2024/948.

Others

- Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Lars Ran, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. *MEDS - Submission to the NIST Digital Signature Scheme standardization process (2023)*. 2023.
- **TODO:** FIX SQIsign round 2?

ABOUT THE AUTHOR

Krijn Reijnders was born in Milheeze, The Netherlands on January 12, 1995. He completed his Bachelor's degree in Mathematics (*cum laude*) at the Radboud University in Nijmegen, together with completing the joint Honours Programme of the Faculty of Science at the same university. His Bachelor's thesis was supervised by Ben Moonen and is titled *"Heisenberg Groups: On applications using representation theory"*.

Krijn then completed his Master's degree in Mathematics in Nijmegen, with parts completed at the Katholieke Universiteit Leuven, in Leuven. His Master's thesis was supervised by Arne Smeets and is titled *"Campana Curves: On integral points of bounded height"*.

After finishing his studies, Krijn worked for two years as a strategy consultant in payments, digital identity, and data sharing at INNOPAY in Amsterdam, before returning to Nijmegen in September 2020 to start a Ph.D. in *post-quantum cryptography* under the supervision of Simona Samardjiska.

In the winter of 2021, Krijn was an intern under the warm supervision of Francisco Rodríguez-Henríquez, at the Cryptographic Research Centre of the Technology Innovation Institute.

In 2022, Krijn co-founded The Isogeny Club together with Jonathan Komada Eriksen, a bi-weekly virtual seminar for young isogenists with a yearly affiliated event at Eurocrypt. He organized 25 talks and 2 events over the span of 2 years, before handing over to Maria Corte-Real Santos.

This thesis represents the research output of Krijn's work, excluding many failed projects, from September 2020 to September 2024. He hopes to continue with a post-doctoral position at the COSIC group of KU Leuven.

Beyond cryptography, Krijn enjoys nature and nature photography, music, travel, literature, running, cycling, and swimming.