UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Isogenies and cryptography

Sotáková, J.

Publication date
2024
Document Version
Final published version

Link to publication

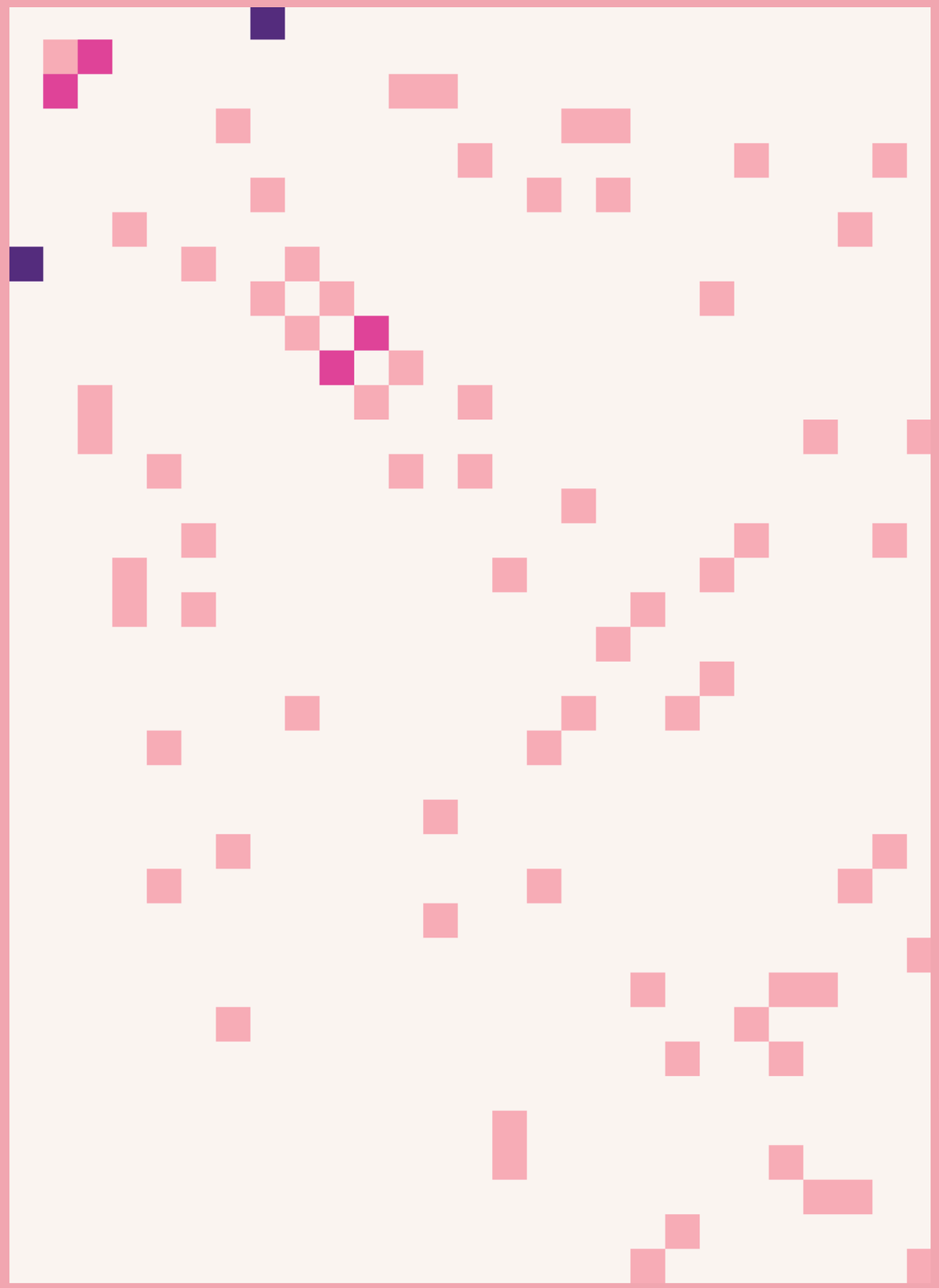
Citation for published version (APA):
Sotáková, J. (2024). *Isogenies and cryptography*. [Thesis, fully internal, Universiteit van Amsterdam].

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:08 Nov 2026



Isogenies and cryptography

Jana Sotáková

Isogenies and cryptography

Jana Sotáková

Isogenies and cryptography

Jana Sotáková

Isogenies and cryptography

ILLC Dissertation Series DS-2024-06

Institute for Logic, Language and Computation

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

This research was supported by the Dutch Research Council (NWO) through Gravitation-grant Quantum Software Consortium - 024.003.037.

Copyright © 2024 by Jana Sotáková

Cover design by Jana Sotáková and Drukkerij Haveka. It shows the pixel graph of the 2-isogeny graph of supersingular elliptic curves for $p = 431$.
Printed and bound by Drukkerij Haveka.

ISBN: 978-90-361-0757-0

Isogenies and cryptography

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit
op vrijdag 28 juni 2024, te 11.00 uur

door Jana Sotáková
geboren te Brno

Promotiecommissie

Promotores:	prof. dr. C. Schaffner	Universiteit van Amsterdam
	prof. dr. S.O. Fehr	Universiteit Leiden
Copromotores:	dr. P.J. Bruin	Universiteit Leiden
Overige leden:	prof. dr. D.R. Kohel	Université d'Aix-Marseille
	prof. dr. T. Lange	TU Eindhoven
	dr. B.P.C. Wesolowski	ENS Lyon
	prof. dr. S.M. Jeffery	Universiteit van Amsterdam
	prof. dr. L.D.J. Taelman	Universiteit van Amsterdam
	dr. K. Guo	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Contents

Acknowledgments ix

List of publications 1

1 Introduction 3

2 Background on isogeny graphs 11
2.1 Elliptic curves . . . 12
2.2 Isogenies . . . 14
2.2.1 Torsion . . . 16
2.2.2 Dual isogenies . . . 17
2.2.3 Equivalent isogenies . . . 18
2.3 Endomorphism rings . . . 19
2.4 Complex multiplication theory . . . 22
2.4.1 The group action . . . 22
2.4.2 CM graph . . . 23
2.5 Isogeny volcanoes . . . 24
2.5.1 Isogenies and conductors . . . 25
2.5.2 Isogeny volcanoes . . . 26
2.5.3 Level structure . . . 28
2.6 Supersingular isogeny graphs . . . 31
2.6.1 Isogeny graphs . . . 31
2.6.2 Isogeny graphs in isogeny-based cryptography . . . 36

3 Adventures in Supersingularland 39
3.1 Introduction . . . 40
3.2 Special j-invariants . . . 41
3.3 Structure of the $\mathbb{F}_p$-subgraph: the spine $\mathcal{S}$ . . . 42
3.3.1 Structure of the $\mathbb{F}_p$-Graph $\mathcal{G}_\ell(\mathbb{F}_p)$ . . . 43
3.3.2 Passing from the graph $\mathcal{G}_\ell(\mathbb{F}_p)$ to the spine $\mathcal{S} \subset \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ . . . 45
3.3.3 The spine for $\ell > 2$ . . . 49
3.3.4 The spine for $\ell = 3$. . . . 53
3.3.5 The spine for $\ell = 2$ . . . 54
3.3.6 Distances between the components of the $\mathcal{S} \subset \mathcal{G}_2(\mathbb{F}_p)$ . . . 61

3.4 Conjugate vertices, distances, and the spine . . . 62
3.4.1 Distance between conjugate pairs . . . 63
3.4.2 How often do shortest paths go through the spine . . . 64
3.4.3 Distance to spine . . . 66
3.5 When are conjugates ℓ-isogenous? . . . 68
3.5.1 Experimental data: 2-isogenies . . . 69
3.5.2 Experimental data: 3-isogenies . . . 70
3.5.3 Further questions . . . 71
3.6 Diameter . . . 72
3.7 Conclusions . . . 74
3.8 Related work . . . 75

4 Background on CSIDH **77**
4.1 Group action in cryptography . . . 78
4.1.1 Diffie–Hellman key exchange . . . 78
4.1.2 Group actions in cryptography . . . 81
4.1.3 Isogeny-based group action . . . 83
4.2 CSIDH setup . . . 84
4.2.1 Choosing group elements to act by . . . 85
4.2.2 CSIDH primes . . . 87
4.3 Algorithmic aspects . . . 88
4.3.1 Determining the kernel of the isogenies . . . 89
4.3.2 Computing one isogeny . . . 91
4.3.3 Strategies . . . 92
4.3.4 Key spaces and constant-time implementations . . . 94

5 Breaking DDH for class group actions **97**
5.1 Introduction . . . 98
5.2 High level overview of the attack . . . 98
5.3 Background . . . 100
5.3.1 Genus theory . . . 100
5.3.2 The Tate pairing on elliptic curves . . . 101
5.4 Characters for ordinary curves . . . 102
5.4.1 Computing the characters χ_i . . . 103
5.4.2 Computing the characters δ, $\delta\epsilon$ and ϵ . . . 105
5.5 Characters for supersingular curves . . . 106
5.6 Impact on DDH and countermeasures . . . 108
5.6.1 Impact on DDH for class group actions . . . 108
5.6.2 Implementation results . . . 110
5.6.3 Countermeasures . . . 110
5.7 Conclusion . . . 111
5.8 Follow-up work . . . 112
5.9 Appendix: Not walking to the floor . . . 112

6 CTIDH: constant-time CSIDH **115**
6.1 Contributions of CTIDH . . . 116
6.2 Batching and key spaces . . . 117
6.3 Isogeny atomic blocks . . . 118
6.3.1 Square-free atomic blocks . . . 118
6.3.2 Restricted square-free atomic blocks . . . 119
6.4 Evaluating atomic blocks in constant time . . . 120
6.4.1 Square-free atomic blocks for isogeny evaluation . . . 121
6.4.2 Restricted square-free atomic blocks . . . 123
6.5 Strategies and parameters for CTIDH . . . 125
6.6 Constant-time software for the action . . . 128
6.6.1 Computing one isogeny . . . 130
6.6.2 Computing the action . . . 131
6.6.3 Automated constant-time verification . . . 133
6.7 Software speeds . . . 134
6.7.1 CTIDH parameters . . . 134
6.7.2 Comparisons . . . 135

7 Disorientation faults in CSIDH **139**
7.1 Introduction . . . 140
7.2 Fault attacks . . . 140
7.3 Attack scenario and fault model . . . 141
7.4 Exploiting orientation flips . . . 142
7.4.1 Implications of flipping the orientation of a point . . . 143
7.4.2 Faulty curves and full-order points . . . 144
7.4.3 Missing torsion . . . 145
7.4.4 Torsion noise . . . 147
7.4.5 Connecting curves from the same round . . . 148
7.4.6 Connecting the components $G^{r,s}$ . . . 149
7.4.7 Revealing the private key . . . 151
7.4.8 Complexity of recovering the secret $\mathfrak{a}$ . . . 151
7.5 Case studies: CSIDH and CTIDH . . . 152
7.5.1 Breaking CSIDH-512 . . . 152
7.5.2 Breaking CTIDH-512 . . . 155
7.5.3 Other variants of CSIDH . . . 157
7.6 The `pubcrawl` tool . . . 158
7.7 Hashed version . . . 160
7.8 Twisting and precomputing . . . 161
7.9 Countermeasures . . . 163
7.9.1 Previous fault attacks and countermeasures . . . 163
7.9.2 Protecting square checks against fault attacks . . . 164
7.9.3 Implementation costs . . . 167

Bibliography 169

Samenvatting 187

Summary 191

Curriculum Vitae 195

Acknowledgments

I would like to thank my advisors Chris, Serge, and Peter, for giving me the opportunity to study isogenies and for the supervision during my PhD. I would also like to thank my thesis committee for evaluating my thesis.

I also thank my advisors for supporting my many academic activities that went beyond research – giving talks at summer schools and interacting with many junior researchers, and getting involved with diversity, equity, and inclusion efforts. These activities surely detracted from the time I could spend on research, but were an integral part of my job as an academic. Moreover, these opportunities strongly aligned with my ethical values and my moral compass, and without engaging in them I would have been lacking many of the skills, knowledge, and connections that have helped me get all the way to writing this thesis.

A big part of my PhD time was affected by the COVID-19 pandemic, and I spent most of my PhD time in my living room, my home office, or, later when it was possible again, traveling. The people I had connected with through academia were important to me beyond research and online collaborations. They also did their best to keep me sane, healthy, and happy throughout what turned out to be very challenging years for me.

Among my collaborators, there are almost too many to name. You have all formed my thinking about mathematics and cryptography. I appreciate all I have learned from you. Working on projects with you was the best part of my PhD.

There are many people who deserve special personal thanks, though I believe that I have always been open about sharing my gratitude for their help:

Lorenz, many thanks for helping me through all the downs and sharing many of the ups in my PhD; for helping me bounce ideas, fixing all of Sage, fixing my code, proofreading my slides and giving feedback on all my writing, and many other kindnesses small and large that you have extended to me over the years.

Travis, thank you for helping me focus on what is important in life, which made the math we did together all the more worth it; for the morning coffee under Mt Timpanogos and for many other dangerously math-filled hikes.

Gustavo, for all the jokes, questions about math, and supporting me through the darker times, the chatter and memes that got us through many working days.

Chloe, for supporting me, advocating for me and always believing in my mathematical (and human) abilities. You have been with me on the mathematical journey the longest and I am extremely grateful for all you have done for me.

Garazi, for reality-checking me and supporting me and helping me grow. We unfortunately share many experiences with academia at its worst, but our many conversations were essential to helping me protect myself and keep on fighting.

Chris M, for all the coffees, support, and clarity, for being my mentor since we overlapped in Amsterdam. You have been my sounding board, and helped me understand many of the confusing things that happened in my PhD.

Edgar, for asking the right questions, the many long discussing we have had, and for giving me the missing pieces to make decisions about my academic future.

Harold, especially for the support in the last few months of writing and trying to graduate, and for the many long chats online and offline.

Dom, for always supporting me, reality-checking me, helping me navigate the world and focusing on what matters. And for all the little cryptographic tidbits we have shared over the years.

There were many QuSofters who have made my experience there enjoyable: Arjan, Lynn, Yfke, Jelena, Yaroslav, Subha, thank you for the time we have spent together. Some of our conversations will stay with me for a long time.

I will also not shy away from admitting that I was not doing well for a great part of my PhD. So I would like to thank everyone who has supported me on this journey, Terezka and Kuba for comparing our PhD experiences and supporting each other, Kristina for remaining my close friend since Berkeley who never judged me and always wanted the best for me, Ivanka for not letting me give up in the winter rain and sleet and cold.

It is impossible for me to do quality research when research is the only thing I have the energy for. But I have been lucky to have been a part of two communities that help me balance my life. I would like to thank everyone at Leiden Atletiek, and the bachata scene, thank you everyone at Bachata Romántica, at Dos Bailadores, and in Seattle.

And even if I did not name you, thank you for sharing this journey with me.

I would have never gotten this far without my family who have stood by my side through all these years and supported me unconditionally, despite my constant urge to eat all the berries whenever we hike. Thank you, my mum and my sister Mirka. Táta by z nás všech měl radost a babička taky.

And finally, no words can describe the unwavering support and love I have received from my partner, my anchor, my everything. Thank you, Stefan.

Oegstgeest
May, 2024

Jana Sotáková

List of publications

" In most areas of mathematics, joint research is a sharing of ideas and skills that cannot be attributed to the individuals separately. The roles of researchers are seldom differentiated (in the way they are in laboratory sciences, for example). Determining which person contributed which ideas is often meaningless because the ideas grow from complex discussions among all partners. Naming a "senior" researcher may indicate the relative status of the participants, but its purpose is not to indicate the relative merit of the contributions. Joint work in mathematics almost always involves a small number of researchers contributing equally to a research project. " [Ame04].

The content of this dissertation is based on the following articles. The author names are ordered alphabetically, and all authors contributed equally.

[ACL+23] **Adventures in Supersingularland**
Sarah Arpin, Catalina Camacho-Navarro, Kristin Lauter, Joelle Lim, Kristina Nelson, Travis Scholl, and Jana Sotáková.
Experimental Mathematics, 32(2):241–268, Taylor & Francis, 2023.
https://doi.org/10.1080/10586458.2021.1926009.

[CSV20] **Breaking the Decisional Diffie–Hellman Problem for Class Group Actions Using Genus Theory**
Wouter Castryck, Jana Sotáková, and Frederik Vercauteren.
CRYPTO 2020, Lectures Notes in Computer Science, 12171(2), 92–120. Springer, 2020.
https://doi.org/10.1007/978-3-030-56880-1_4.

[CSV22a] **Breaking the Decisional Diffie–Hellman Problem for Class Group Actions Using Genus Theory: Extended version**
Wouter Castryck, Jana Sotáková, and Frederik Vercauteren.
Journal of Cryptology, 35(4):24, Springer, 2022.
https://doi.org/10.1007/s00145-022-09435-1.

[BBC+21] **CTIDH: faster constant-time CSIDH**
Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, Tung Chou, Tanja Lange, Michael Meyer, Benjamin Smith, and Jana Sotáková.
CHES 2021, IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(4), 351–387.
https://doi.org/10.46586/tches.v2021.i4.351-387.

[BKL+23] **Disorientation Faults in CSIDH**
Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska.
EUROCRYPT 2023, Lecture Notes in Computer Science, 14008, 310–342, Springer, 2023.
https://doi.org/10.1007/978-3-031-30589-4_11.

Chapter 1

Introduction

Cryptography is the art of keeping information private. This is especially handy and desirable if the information is exchanged over the Internet, but also when paying with your debit card instead of by cash.

To achieve privacy, we use *encryption*: before sending our data, we use an algorithm that transforms the data into a random-looking string of ones and zeros using a *secret key*. Upon receipt, this random-looking string can be *decrypted* to recover the message we sent. The most efficient way to encrypt data is by an algorithm that uses the *same* secret key to encrypt and decrypt the data, this is called *symmetric* cryptography.

In many situations, we need encryption when communicating with somebody we have never met before (think of a new vendor on the internet, or even visiting a new website). For this, we use *asymmetric* cryptography, also known as *public-key cryptography*. In public-key cryptography, everybody has two keys: a *private* key, which is kept secret at all times, and a *public* key, which everyone can see. The public key can be used by anyone to encrypt a message, but only the person holding the secret key (that is, the intended recipient) can decrypt the message.

The ElGamal protocol. A fundamental public-key cryptography protocol is the Diffie–Hellman key exchange [DH76], which we will return to in Section 4.1.1. For now, we will take as an example the ElGamal encryption protocol [ElG84]. Suppose Anouk wishes to make it possible for others to contact them. Anouk selects a prime p, which defines a finite group $((\mathbb{Z}/p\mathbb{Z})^\times, \cdot)$, a generating element $g \in (\mathbb{Z}/p\mathbb{Z})^\times$, and a random exponent $a \in \{1, \ldots, p-1\}$ that will be the secret key. Anouk's public key will be g^a, computed $\bmod p$ (the public key should also include p and g). If Bas wants to send a message $m \in (\mathbb{Z}/p\mathbb{Z})^\times$ to Anouk, then Bas can generate a random $b \in \{1, \ldots, p-1\}$, compute $g^b \bmod p$, and send to Anouk $(g^b, (g^a)^b \cdot m)$. Anouk can decrypt the encrypted message from (c_1, c_2) by computing $c_2 \cdot c_1^{-a}$.

One property we certainly wish for is that nobody can impersonate either

of the communicating parties: that nobody who only holds the public key can recover the secret key. More formally, upon seeing g^a, it should be *hard* to recover a. This is the *discrete logarithm problem* (DLP). However, recovering the secret key is not the only way the protocol might be attacked: rather than recovering the secret key, an attacker might simply wish to recover the contents of the message.

The ElGamal protocol is secure by definition (technically, IND-CPA secure) if nobody can guess which message m was encrypted just by looking at the ciphertext. More precisely, if Bas sends either message m_0 or m_1, it should be hard for an attacker (without the secret key) to determine which message was encrypted.

Suppose we can guess that m_0 was the encrypted message. Then $c_2 \cdot m_0^{-1} = g^{ab}$, and $c_2 \cdot m_1^{-1}$ looks like a random element in $(\mathbb{Z}/p\mathbb{Z})^\times$, and *vice versa* if m_1 is the message Bas sent. Therefore, a successful attacker can distinguish between tuples of the form (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, h) where h is just a random element. Efficiently distinguishing between these tuples is the *decisional Diffie–Hellman* problem (abbreviated DDH).

DDH is a *stronger* cryptographic assumption than DLP: if DDH is hard, then DLP also is hard. In the group $(\mathbb{Z}/p\mathbb{Z})^\times$, the DDH problem is in fact known to be easy, and there is a very elegant distinguisher (see Example 4.1.5). We can replace the group $(\mathbb{Z}/p\mathbb{Z})^\times$ by any cyclic group G with generator g.

We stress that the hardness of the DLP or DDH problem are not a property of the group, but of the computational representation (how the elements and the group law are represented). For instance, the groups $(\mathbb{Z}/(p-1)\mathbb{Z}, +)$ and $((\mathbb{Z}/p\mathbb{Z})^\times, \cdot)$ are both cyclic groups of order $p-1$, but the DLP problem is trivial in the former and believed to be hard in the latter. Indeed, the bijection $a \mapsto g^a$ for some fixed g is easy to compute one way, but difficult to invert (this is exactly the DLP problem). For practical choices for computational representations see the recommended groups and sizes in [CSA18].

However, DLP in $(\mathbb{Z}/p\mathbb{Z})^\times$ is only hard if we assume a classical model of computation: that the eavesdropper who wants to know Bas's message to Anouk does not have a quantum computer. On a large-scale quantum computer, Shor's algorithm [Sho99] could be used to break all of the major public-key cryptography protocols we use nowadays.

Post-quantum cryptography. The answer to this threat is *quantum-safe cryptography*, also called *post-quantum cryptography*, which develops public-key cryptography protocols that we believe to be secure even against attackers with quantum computers. With billions of devices currently connected to the internet, changing the encryption protocols in practice is a major undertaking. At the time of writing in 2024, the cryptographic community has largely reached consensus on which new protocols to use, and various government agencies are in various stages of preparing *standards*: real-world implementations of these protocols that

have been tested extensively to fine tune the parameters and avoid as many bugs as possible. See for instance the NIST PQC standardization effort [AAC+22] or the BSI recommendations [Fed22].

Most post-quantum protocols are based on hard computational problems in one of the following five areas: lattices, codes, multivariate, hash, or isogenies. The focus of this thesis is on isogeny-based cryptography. While it is admittedly not a frontrunner for cryptographic solutions, studying isogenies is interesting to have more variety in the computational assumptions we can use.

Isogeny-based cryptography. Isogenies are maps between elliptic curves, which are certain abelian algebraic groups. Isogenies have remarkable algebraic properties and structure (which we explore in Chapter 2), we can also think of them as relations between elliptic curves. This allows us to define *isogeny graphs*: the vertices are elliptic curves, and there is an edge between two elliptic curves if they are *isogenous*. To make isogeny graphs finite, we restrict ourselves to elliptic curves defined over a finite field $\mathbb{F}_q$, identify isomorphic elliptic curves, and consider isogenies of specific degree.

Protocols based on computational assumptions related to isogenies are called *isogeny-based protocols*. Such protocols are slower than other post-quantum protocols (and in particular, lattice-based protocols), but can offer other benefits: small key sizes, non-interactive key exchange, or *extra structure* that allows us to build advanced cryptographic protocols. Some isogeny-based protocols are:

- key exchange: SIDH [JD11, DJP14, SIK17];
- *non-interactive* key-exchange CRS and CSIDH [Cou06, RS06, CLM+18];
- signatures: SeaSign and CSI-FiSh [DG19, BKV19], GPS signatures [GPS17] and SQISign [DKL+20];
- hash functions [CLG09];
- threshold schemes [DM20];
- oblivious transfer [ADMP20, LGD21].

The same abundance of structure has led to many successful breaks of some of the aforementioned protocols. Most protocols in the SIDH family [JD11, SIK17] were completely broken in an astonishing way [CD23, MM22, Rob23]. The new protocols, such as SQISign [DKL+20], have only been studied for a couple of years and need to build up public confidence. We stress that different choices of isogeny graphs lead to vastly different computational assumptions, so breaking one protocol *does not* break all isogeny-based protocols.

CSIDH. One family of isogeny-based protocols that was completely unaffected by the series of SIDH breaks in 2022, is the *Commutative Supersingular Isogeny Diffie–Hellman* (CSIDH) family of protocols [CLM+18]. This framework was first introduced by [Cou06, RS06] (the CRS protocol), extended by [DKS18], and brought to cryptographically competitive speeds by [CLM+18].

Abstractly, the CRS and CSIDH protocols are based on *group actions*. Assume that a finite abelian group G acts on a finite non-empty set X freely and transitively: denoting the group action by $\star$, then for any $x, y \in X$ there exists a unique $g \in G$ with $y = g \star x$. By selecting a *base point* $x \in X$, there is a bijection of *sets* $G \to X$ given as $g \mapsto g \star x$.

Cryptographic group actions assume that the computation of $g \mapsto g \star x$ is efficient but the inverse is *hard* (starting from some element $y \in X$ and computing a $g \in G$ such that $y = g \star x$). Before we considered different computational representations of a finite cyclic group of a fixed size and the DLP was easy in one and believed hard in another. Here we have a set in bijection with a finite abelian group, but the lack of an efficiently computable mapping $y \in X \mapsto g \in G$ means that operations or problems which are easy in the group (such as composing two elements) are impossible in the set (it is not equipped with any further structure). In particular, it is hard to take two elements of the form $a \star x$ and $b \star x$, and produce the "composition" $(ab) \star x$.

Group-action based protocols replace the generator g in the Diffie–Hellman paradigm with some base point $x \in X$ and the exponentiation with the group action: Anouk and Bas's secrets would be $a, b \in G$, public keys $a \star x$, respectively $b \star x$, and the shared secret $a \star (b \star x) = b \star (a \star x) = (ab) \star x$ (more details in Section 4.1.2). The computation is performed efficiently by acting by elements of the group G, but the public keys and other elements are set elements with no other structure. Working with the set elements (and not the group elements) is believed to hide the group structure so that Shor's algorithm no longer applies, and so it is believed that the analogues of the DLP problem are hard.

In CRS and CSIDH, the group G is the class group $\mathrm{Cl}(\mathcal{O})$ of an imaginary quadratic order $\mathcal{O}$, and the set X is the set of isomorphism classes of elliptic curves over $\mathbb{F}_q$ with complex multiplication (CM) by $\mathcal{O}$ with fixed trace. Acting by a general element $a \in G$ is in general very costly. In CSIDH, one chooses a set of elements $\mathfrak{l}_1, \ldots, \mathfrak{l}_n \in G$ such that the action by each of these can be evaluated cheaply, and only acts by elements of the form $\prod_{1 \leq i \leq n} \mathfrak{l}_i^{e_i}$. Elements are therefore represented by *exponent vectors* $(e_1, \ldots, e_n) \in \mathbb{Z}^n$. Exponent vectors are chosen from some finite *key space* $\mathcal{K} \subset \mathbb{Z}^n$, which is a part of the parameters of the protocol. The original protocol chose key spaces $\mathcal{K}_m = \{-m, \ldots m\}^n$ for some m but many other choices have been proposed since (see Section 4.3.4).

The shape of the key space has a big influence on the efficiency of the protocol. Especially so if we consider *constant-time* implementations: algorithms for computing the group action $a \star E$ such that the running time does not depend on the secret key a. *A priori*, the running time of computing the group action

by $\mathfrak{l}_i$ depends on i. Very simplistically, looking at the running time of the group action would reveal how many times each $\mathfrak{l}_i$ was used, but this is the secret key!

Organization of this thesis

In this thesis, we study cryptographic problems relating to the protocol CSIDH. The topics we discuss range from cryptanalyzing some of the computational assumptions, to fast *constant-time* implementations, to studying security while giving an attacker physical access to the device that is running a CSIDH key exchange. We also study isogeny graphs in general.

This thesis includes two background chapters and four chapters based on published work [ACL+23, CSV20, BBC+21, BKL+23]. We organize the material as follows: Chapter 2 contains background on elliptic curves, isogenies, and isogeny graphs, focusing primarily on the mathematical backgrounds of isogeny-based cryptography. Then we continue with Chapter 3 studying supersingular isogeny graphs, based on the paper [ACL+23].

Afterwards, we leave the general world of isogeny graphs, and focus on the isogeny protocol CSIDH [CLM+18]. We start with a self-contained and detailed discussion of the protocol in Chapter 4, discussing the general framework, parameter choices, and implementations, to set the stage for the remaining chapters.

Chapters 5 to 7 are based on the papers [CSV20, BBC+21, BKL+23], dealing with different aspects of the CSIDH protocol: analyzing some of the hard problems underlying CSIDH (variants of the decisional Diffie–Hellman problem), implementing CSIDH in *constant time* (not leaking information about secrets via the timing of the computation), and developing a *fault attack* on CSIDH (roughly speaking an attack on a physical device running the CSIDH protocol).

Below we give more details on what is included in the chapters based on the papers [ACL+23, CSV20, CSV22a, BBC+21, BKL+23].

Adventures in Supersingularland. In Chapter 3, we take a look at supersingular isogeny graphs in general. This chapter is based on the paper *Adventures in Supersingularland* [ACL+23], which is joint work with Sarah Arpin, Catalina Camacho-Navarro, Kristin Lauter, Joelle Lim, Kristina Nelson, and Travis Scholl. Isogeny graphs have as vertices (isomorphism classes of) elliptic curves, and as edges (equivalence classes of) isogenies between elliptic curves. We look at the supersingular isogeny graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ of supersingular elliptic curves over $\overline{\mathbb{F}}_p$, together with isogenies over $\overline{\mathbb{F}}_p$ of degree ℓ (defined in Definition 2.6.3). We know many of the *graph* properties of the graph: it is connected, $\ell+1$-regular, Ramanujan, with short diameter (we collect these statements in Proposition 2.6.7).

We ask a number of previously unexplored questions about the properties that are specific to supersingular elliptic curves. The vertices in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ that correspond to j-invariants in the *subfield* $\mathbb{F}_p$ were studied by Delfs and Galbraith [DG16] in

the context of a path-finding algorithm. Their algorithm finds a path to a vertex in $\mathbb{F}_p$ and reduces the problem to finding a path among vertices in $\mathbb{F}_p$, for which subexponential quantum algorithms are known [BJS14]. We study not just the vertices but the *subgraph* $\mathcal{S} \subset \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ induced by the $\mathbb{F}_p$-vertices, which we call the *spine*. The graph $\mathcal{S}$ is intimately linked to the graph $\mathcal{G}_\ell(\mathbb{F}_p)$ of curves defined over $\mathbb{F}_p$, together with degree-ℓ isogenies over $\mathbb{F}_p$. We prove that the graph $\mathcal{S}$ is formed from $\mathcal{G}_\ell(\mathbb{F}_p)$ by a sequence of steps we call *stacking, folding, and attaching*. The main result is that the structure of $\mathcal{S}$ is very close to that obtained by identifying the $\mathbb{F}_{p^2}$-equivalent edges and vertices in $\mathcal{G}_\ell(\mathbb{F}_p)$ and we determine a full list of the subtle differences. This gives new structural understanding of the $\mathbb{F}_p$-vertices within the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$.

There is an *involution* on $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ induced by the Frobenius map $x \mapsto x^p$, which we call the *mirror* symmetry. Note that the fixed vertices of the mirror symmetry are exactly the vertices of $\mathcal{S}$. We study the paths in the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ that are fixed under this involution, and in particular experiment with the shortest such paths: pairs of ℓ-isogenous conjugate j-invariants. We collect experimental data on the distribution of these paths, and also on the diameter of the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$.

In all our experiments, there is a clear bias depending on the congruence class of the prime p mod 12, and we propose explanations based on the shape of $\mathcal{S}$. Our results help understand the properties of the $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ beyond the general graph properties listed above.

Breaking the Decisional Diffie–Hellman assumption. Chapter 5 is based on the joint work with Frederik Vercauteren and Wouter Castryck, *Breaking the decisional Diffie-Hellman problem for class group actions using genus theory* [CSV20], and the extended version in [CSV22a].

In Chapter 5, we study the group-action analogue for the DDH problem for CRS and CSIDH: distinguishing tuples of the form $(x, a \star x, b \star x, ab \star x)$ from tuples of elements $(x, a \star x, b \star x, c \star x)$ for $a, b, c \in G$ random. The group G is the class group $\mathrm{Cl}(\mathcal{O})$ of some imaginary quadratic order $\mathcal{O}$.

Very little is known about the structure of the group $\mathrm{Cl}(\mathcal{O})$: even computing its order in practical instances is a highly non-trivial task [HM89, BKV19]. Previously it was assumed that the group structure of $\mathrm{Cl}(\mathcal{O})$ (such as the possible existence of small subgroups) was sufficiently hidden in the group action and did not influence the hardness of the computational problems.

We show that this assumption is not true, and that the hardness of the DDH problem is related to the 2-part of the class group $\mathrm{Cl}(\mathcal{O})$. This 2-part is further described by *genus theory*, which goes back to Gauss, and relates to ramified primes in the order $\mathcal{O}$. Each ramified prime m induces an *assigned* quadratic character χ on $\mathrm{Cl}(\mathcal{O})$, and we show how to evaluate the character $\chi(a)$ from a pair of elliptic curves $E, E'/\mathbb{F}_q$ satisfying $E' = a \star E$.

We compute this character using the Tate pairing on E and E', and a discrete

logarithm computation in the group of m-th roots of unity in $\mathbb{F}_q$. Because the discrete logarithm computation is only efficient if m is small, we break the DDH problem *efficiently* whenever there exists a *non-trivial* assigned character for a *small* prime m dividing the discriminant $\mathcal{O}$: a prime m satisfying $m = O(\log q)$.

Our attack approach works for both ordinary curves over $\mathbb{F}_q$ for q arbitrary and supersingular curves over $\mathbb{F}_p$ for p prime. In the ordinary case, there is a suitable non-trivial character for a density 1 set of quadratic orders $\mathcal{O}$. In the supersingular case, however, non-trivial characters only exist for $p \equiv 1 \pmod 4$, so the attack does not impact the CSIDH scheme, which uses $p \equiv 3 \pmod 4$.

CTIDH: constant-time CSIDH. Chapter 6 is based on the paper *CTIDH: constant-time CSIDH* [BBC+21], which is joint work with Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, Tung Chou, Tanja Lange, Michael Meyer, and Benjamin Smith. In this paper, we develop a new keyspace for CSIDH and a new algorithm for *constant-time* evaluation of the CSIDH group action $a \star E$. Used separately, the new algorithm and the new key space do not produce faster implementations, but together they produce speed records.

Recall that in CSIDH, we restrict to acting by select group elements $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$. In the CTIDH key space, we split these elements into *batches*. Say one batch includes elements $\mathfrak{l}_1, \ldots \mathfrak{l}_j$, we then bound the 1-norm of the *slice* of the secret vector $(e_1, \ldots, e_j)$. Using n batches is the CSIDH key space; using one batch was studied in [BLMP19] but the overhead is massive, and in [NOTT23] but only for variable-time computation.

In the CTIDH algorithm, we use the *same code* (which we describe below) to evaluate the action by each of the elements in the *same batch*. In [BLMP19], the authors noted that the isogeny evaluation has a *Matryoshka doll* structure: the code for computing an ℓ-isogeny includes all the computation to compute ℓ'-isogenies for any $\ell' < \ell$. We extend this to the faster $\sqrt{\text{élu}}$ formulas [BDLS20]. If the isogeny degrees are of similar size, the overhead from using Matryoshka-like code is very small.

Combined, we split the primes into batches of roughly the same size, and use the Matryoskha-isogeny algorithm (implemented in constant time) to compute the isogeny evaluation per batch. So the Matryoskha-isogeny does not reveal information on the particular prime within a batch. The batching key space allows for smaller entries in the secret vectors, just as in [NOTT23]. Therefore, we also save on the number of isogenies we need to compute. To achieve constant time, we compute the same number of isogenies per batch. This number is public and depends on the choice of batches, so again does not reveal information about the exponent vector, that is, the secret.

The resulting CTIDH-512 protocol is almost twice as fast as the previous constant-time implementations of CSIDH-512. We also give larger parameter sets, and a greedy algorithm to look for (locally optimal) batch configurations.

Disorientation fault attacks on CSIDH. The final Chapter 7 is based on the paper *Disorientation faults in CSIDH* [BKL+23], which is joint work with Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, and Monika Trimoska. In this chapter, we design a *fault attack* on the CSIDH family of protocols. Our attack is different from other fault attacks on isogeny protocols in that it does not attack a particular implementation of CSIDH protocols, but is more general and affects most implementations. For most implementations, the attack would only require a couple hundred successful fault injections.

The fault attack assumes the following scenario: the attacker has (physical) access to a device computing the group action $E \mapsto a \star E$ for a *fixed* secret key a, and is able to induce a *fault* (error) in the computation. Implementations of isogeny group-action protocols compute the action in *rounds*: in every round, the algorithm selects a certain subset $S \subset \{1, \dots n\}$ and an *orientation* $s = \pm 1$ fixed for the round, and the action by $\prod_{i \in S} \mathfrak{l}_i^s$ is computed. A *disorientation* attack assumes we are able to *flip* this orientation, and that instead the action by $\prod_{i \in S} \mathfrak{l}_i^{-s}$ is computed.

The faulted curve differs from $a \star E$ by the action of $t = \prod_{i \in S} \mathfrak{l}_i^{2s}$. But the set S depends on the secret key a and the round in which it was computed (and possibly previous rounds), and this depends on the secret a. We reconstruct the whole secret a from collecting faulted curves from different rounds and finding the corresponding sets S. To achieve this in practice, we optimize the strategy and also develop meet-in-the-middle software `pubcrawl` to find the connecting element t. We show that the recovery is practical for CSIDH-512 and even more efficient for CTIDH-512, and discuss other implementations.

We also discuss more realistic scenarios in which one does not recover the faulted curve directly, but a *hash* of it: the output is passed through a key-derivation function. We show that our attack can be mounted in this situation as well. Moreover, we take advantage of the *quadratic twisting* property of isogenies to allow for efficient precomputation.

Finally, we discuss countermeasures to previous fault attacks, and give new lightweight countermeasures.

Chapter 2

Background on isogeny graphs

In this section, we set the stage for isogeny graphs. We discuss the theory of elliptic curves over finite fields, isogenies, endomorphism rings, and isogeny graphs.

Details on group actions in isogeny-based cryptography, the CSIDH family of protocols, implementation and engineering aspects, are treated in Chapter 4.

A comprehensive reference for elliptic curves and isogenies is [Sil09], which contains all the proofs for Sections 2.1 and 2.2; Section 2.3 is based on [Wat69]; for complex multiplication theory in Section 2.4 the most complete resource is [Cox89]; for isogeny volcanoes in Section 2.5 one can consult the original source [Koh96] or the approach via modular polynomials by [Sut13b]. Section 2.6 is partially based on the introduction to [ACL+23].

2.1 Elliptic curves

Let p be a prime, $n \in \mathbb{N}$ a positive integer and set $q = p^n$. Denote by $\mathbb{F}_q$ a finite field with q elements, and choose an algebraic closure $\mathbb{F}_p \subset \mathbb{F}_q \subset \overline{\mathbb{F}}_q$. Assume throughout that $p > 3$.

Definition 2.1.1 (Elliptic curve)**.** An *elliptic curve* E over a finite field $\mathbb{F}_q$ is an algebraic curve given by the projective closure of the affine curve with equation

$$E : y^2 = x^3 + ax + b, \quad a, b \in \mathbb{F}_q,\ 4a^3 + 27b^2 \neq 0. \tag{2.1}$$

When we want to emphasize that E is defined over $\mathbb{F}_q$, we write $E/\mathbb{F}_q$. The elliptic curve E can be given an abelian group structure such that the projective point $(0 : 1 : 0)$, called the point at infinity ∞_E, is the neutral element. The usual group structure on E is called the *tangent-and-chord law* [Sil09, III.3].

The *points of* E are pairs $P = (x_P, y_P) \in (\overline{\mathbb{F}}_q)^2$ satisfying the equation (2.1), together with the point at infinity. The geometric definition of the group law is:

$$P + Q + R = \infty_E \quad \longleftrightarrow \quad P, Q, R \text{ lie on a line;}$$

and can also be given by algebraic formulas that only depend on the coefficients of the defining equation of E. The group law can be computed very efficiently: if $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are points on E, then we can compute the coordinates of the point $R = (x_R, y_R)$ using simple algebraic formulas. As an example, in the generic case $x_P \neq x_Q$, the coordinate x_R is given as

$$x_R = \left(\frac{y_P - y_Q}{x_P - x_Q}\right)^2 - x_P - x_Q. \tag{2.2}$$

Rational points. The subgroup of *rational points* $E(\mathbb{F}_q)$ is given by the point at infinity ∞_E and points (x, y) of E with both coordinates in $\mathbb{F}_q$. Since the group law (see Equation (2.2)) is given by formulas in the coordinates of the points and $a, b \in \mathbb{F}_q$, the set of rational points forms a subgroup of E.

The subgroup of rational points $E(\mathbb{F}_q)$ is a finite group. More can be said:

Theorem 2.1.2 (Hasse-Weil)**.** *The number of rational points of* $E/\mathbb{F}_q$ *satisfies* $\#E(\mathbb{F}_q) = q + 1 - t$ *for some integer* t *such that* $|t| \leq 2\sqrt{q}$.

The integer t from Theorem 2.1.2 has a recurring role throughout the chapter.

Definition 2.1.3 (Trace of Frobenius)**.** The *trace of Frobenius* is the integer t satisfying $\#E(\mathbb{F}_q) = q + 1 - t$.

Determining the number of points of $E/\mathbb{F}_q$ is equivalent to computing the trace of Frobenius. This can be done very fast (in polynomial time in $\log p$), for instance using the SEA algorithm [Sch95].

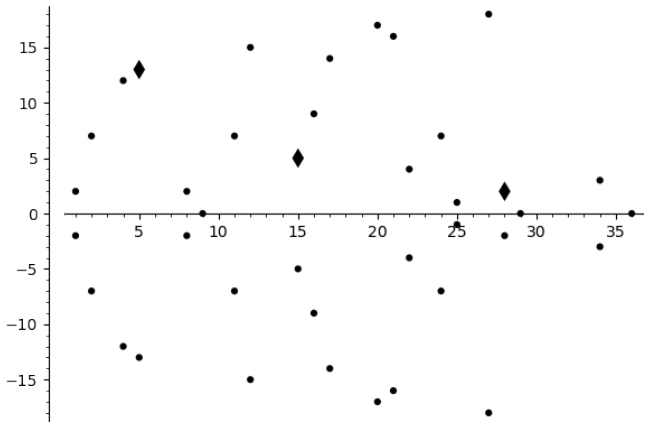


Figure 2.1: Rational points of $E : y^2 = x^3 + x + 2$ over $\mathbb{F}_{37}$. There are 40 rational points, including ∞_E (not shown). Points with diamond shape $(5, 13), (15, 5)$ and $(28, 2)$ lie on the same line $L : -11x + 14y + 16$ and thus sum to ∞ on E.

Further, there is a simple recurrence formula for the number of points over an extension [Sil09, V, Exercise 5.13]: if $t_n = q^n + 1 - \#E(\mathbb{F}_{q^n})$ and $t_0 = 2$, then

$$t_{n+2} = t_1 \cdot t_{n+1} - q t_n. \tag{2.3}$$

Definition 2.1.4. Let $E/\mathbb{F}_q$ be an elliptic curve with $q + 1 - t$ points.

- If $p \nmid t$ then E is called *ordinary*,
- if $p \mid t$ then E is called *supersingular*.

Change of coordinates. Any elliptic curve $E/\mathbb{F}_q$ admits different models: different equations that describe the same algebraic group, and which can be obtained by a change of coordinates. The short Weierstrass model (2.1) is the simplest model one encounters in mathematics; in cryptography, other models, such as the Montgomery curves (see Definition 4.2.5) are more popular.

For simplicity, let us only define the change of coordinates for short Weierstrass models (2.1): any change of coordinates is a map

$$(x, y) \mapsto (u^2 x, u^3 y) \quad \text{for some } u \in \overline{\mathbb{F}}_q. \tag{2.4}$$

It is easily seen that this transformation takes the equation $y^2 = x^3 + ax + b$ into another short Weierstrass equation $y^2 = x^3 + au^{-4}x + bu^{-6}$.

Elliptic curves that differ by a change of coordinates are called *isomorphic*; if this change of coordinates can be defined over $\mathbb{F}_{q^n}$, they are said to be isomorphic over $\mathbb{F}_{q^n}$. A change of coordinates as in Equation (2.4) is an isomorphism over $\mathbb{F}_{q^n}$ if and only if $u \in \mathbb{F}_{q^n}$. An isomorphism $\varphi : E \to E$ is called an *automorphism*.

The following quantity is always preserved by the coordinate changes in (2.4):

Definition 2.1.5. For an elliptic curve $E : y^2 = x^3 + ax + b$, the j-invariant is

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

The j-invariant is an $\overline{\mathbb{F}}_q$-isomorphism invariant; an elliptic curve E is isomorphic to an elliptic curve E'/F for any field F such that $j(E) \in F$ [Sil09, III.1, Prop. 1.4]. Curves that are $\overline{\mathbb{F}}_q$-isomorphic but *not* $\mathbb{F}_q$-isomorphic are called *twists*.

Example 2.1.6 (Twists)**.** Let d be a non-square in $\mathbb{F}_q$ and $E/\mathbb{F}_q$ be an elliptic curve given by $y^2 = x^3 + ax + b$. Then $\tilde{E} : y^2 = x^3 + d^2ax + d^3b$ is a twist of E. The curves E and $\tilde{E}$ are isomorphic over $\mathbb{F}_{q^2}$ by setting $u = \frac{1}{\sqrt{d}}$ in Equation (2.4).

Remark 2.1.7. There are many equivalent definitions of supersingularity, see [Sil09, V.3, Thm. 3.1]. Those make it easier to see that being ordinary or supersingular is a property of the j-invariant. So, we talk about supersingular (or ordinary) j-invariants in $\mathbb{F}_q$. All supersingular j-invariants already lie in $\mathbb{F}_{p^2}$: there are therefore only finitely many. The number of supersingular j-invariants can be given as a simple formula in p and is about $p/12$ [Sil09, V.4].

2.2 Isogenies

Definition 2.2.1. Let $E, E'/\mathbb{F}_q$ be elliptic curves. An *isogeny* $\varphi : E \to E'$ is a non-constant $\mathbb{F}_q$-morphism of algebraic curves satisfying $\varphi(\infty_E) = \infty_{E'}$.

Equivalently, an isogeny $\varphi : E \longrightarrow E'$ is an $\mathbb{F}_q$-rational map of the form

$$(x, y) \longmapsto (f(x), y \cdot g(x)) \tag{2.5}$$

for some rational functions $f, g \in \mathbb{F}_q(x)$. Any isogeny is a group homomorphism of the elliptic curves E and E' as algebraic groups [Sil09][III, Theorem 4.8]. Note that our definition restricts isogenies to be defined over the same field $\mathbb{F}_q$ as the curves E and E'. We choose for this restriction as those are the only isogenies we will encounter in this thesis.

The *degree* of the isogeny φ is its degree as a morphism, and hence is multiplicative for composition of isogenies. The isogeny is *separable* if $\deg \varphi = \#\ker \varphi$; if $p \mid \deg \varphi$, it may be that $\#\ker \varphi < \deg \varphi$, and such isogenies are *inseparable*. Isogenies of degree 1 are isomorphisms.

Example 2.2.2. An isogeny over $\mathbb{F}_{37}$ of degree 5, visualized in Figure 2.2:

$$\varphi : E : y^2 = x^3 + x + 2 \longrightarrow E' : y^2 = x^3 + 31x + 4$$
$$(x, y) \longmapsto \left(\frac{x^5 + 8x^4 + 3x^3 + 3x^2 + 3x + 14}{x^4 + 8x^3 + 9x^2 + 9x + 3}, y \cdot g(x)\right).$$

The function $g(x)$ is ommitted for brevity. The kernel of this isogeny is
$\ker \varphi = \{\infty_E\} \cup \{(x, y) : x^2 + 4x + 15 = 0\} = \{\infty_E, (8, \pm 2), (25, \pm 1)\}$.
Since the degree is equal to the size of the kernel, this isogeny is separable.

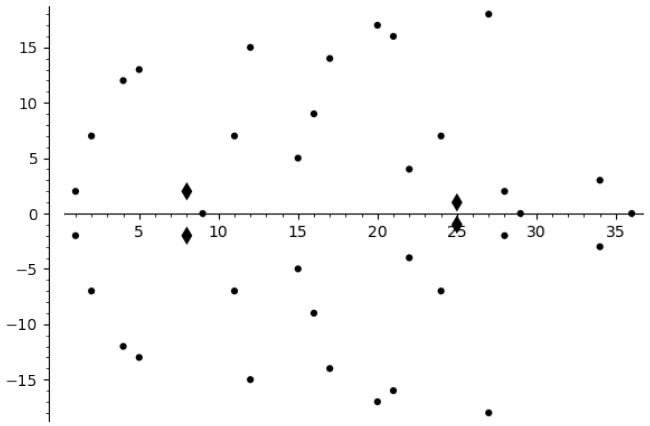


Figure 2.2: Points on $E(\mathbb{F}_{37})$. The points in the kernel of φ from Example 2.2.2 are labelled with diamonds.

Separable isogenies have the following factorization property: if $\varphi : E \to E'$ and $\psi : E \to E''$ are isogenies such that $\ker \varphi \supset \ker \psi$, then there exists an isogeny $\rho : E'' \to E'$ such that $\varphi = \rho \circ \psi$.

$$\begin{array}{ccc} E & \xrightarrow{\quad\varphi\quad} & E' \\ & {\scriptstyle\psi}\searrow \quad \nearrow{\scriptstyle\rho} & \\ & E'' & \end{array} \tag{2.6}$$

Hence, separable isogenies are determined by their kernel, up to post-composition with an isomorphism. Conversely, for any subgroup $G \subset E$, there exists an isogeny $\varphi : E \to E'$ with kernel $\ker \varphi = G$.

Inseparable isogenies enjoy a similar factorization property: if p^r is the inseparability degree of φ as a rational map $E \to E'$, then $\varphi = \psi \circ \pi_p^r$ where the isogeny ψ is separable and $\pi_p : E \to E^{(p)}$ is the Frobenius map $(x, y) \mapsto (x^p, y^p)$. We admit to a small abuse of notation as the power π_p^r is in fact a composition of several isogenies between possibly different curves $E \to E^{(p)} \to E^{(p^2)} \cdots \to E^{(p^r)}$. We will return to Frobenius in Definition 2.3.2.

The kernel $\ker \varphi$ is clearly a subgroup of E, and can be defined over the same field as the isogeny φ; in the form Equation (2.5), the kernel can be defined by the (radical of the) denominator of the function $f(x)$. Vélu's [Vél71], Kohel's [Koh96] or √élu [BDLS20] formulas are the most popular formulas to compute rational maps of φ from a description of the kernel $\ker \varphi$. More details on these formulas are in Section 4.3.2. For now, we note that evaluating an isogeny using Vélu's formulas requires $O(\deg \varphi)$ multiplications in the field over which the *points* in $\ker \varphi$ are defined. This is in general an extension of large degree. Therefore, we strive to select parameters such that also the points in the kernel of the isogenies

are defined over $\mathbb{F}_{p^2}$ (or very small extensions).

Finding isogenies of elliptic curves is the foundational problem of isogeny-based cryptography. Certifying existence amounts to point counting:

Theorem 2.2.3 (Tate)**.** *Elliptic curves E and E' over $\mathbb{F}_q$ are isogenous over $\mathbb{F}_q$ if and only if they have the same number of points, i.e.,*

$$\operatorname{tr} \pi_E = \operatorname{tr} \pi_{E'}.$$

The isogeny path-finding problem, in its most basic form, is the following:

Remark 2.2.4 (Isogeny path finding)**.** Given two isogenous elliptic curves E and E' defined over $\mathbb{F}_q$, find *any* isogeny $\varphi : E \to E'$.

For supersingular elliptic curves, the isogeny-path finding problem is equivalent to that of computing their endomorphism rings [KLPT14, Wes22]. Endomorphism rings (Section 2.3) play a big role in studying isogenies of elliptic curves (see Sections 2.4.1 and 2.5).

2.2.1 Torsion

The most basic isogenies are the scalar multiplication maps.

Example 2.2.5. Let $E/\mathbb{F}_q : y^2 = x^3 + ax + b$. Multiplication by n for $n \in \mathbb{Z}$ is denoted $[n] : E \to E$ and sends $P \mapsto [n]P$. If $n \neq 0$, this is an isogeny because the group law is defined algebraically and over $\mathbb{F}_q$. The degree is $\deg[n] = n^2$.

- Multiplication by 2 is a degree-4 map:

$$[2] : (x, y) \mapsto \left(\frac{x^4 - 2ax^2 - 8bx + a^2}{4(x^3 + ax + b)}, y \cdot g(x) \right),$$

  omitting the description of $g(x)$ for brevity. Looking at the denominators, we can read off the kernel $\ker([2]) = E[2] = \{(x, y) : x^3+ax+b = 0\} \cup \{\infty_E\}$.

- Multiplication by 3 is a degree-9 map (lot denotes lower-order terms):

$$[3] : (x, y) \mapsto \left(\frac{x^9 + \text{lot}}{(3x^4 + 6ax^2 + bx - a^2)^2}, y \cdot g(x) \right).$$

Definition 2.2.6 (n-torsion)**.** The kernel of the multiplication by n map is called the *n-torsion* and is denoted as $E[n]$. We further set $E[n^\infty] = \cup_{k \in \mathbb{N}_{>0}} E[n^k]$.

The points of $E[n]$ may have coordinates in a large extension of $\mathbb{F}_q$; the subgroup of rational n-torsion is denoted by $E(\mathbb{F}_q)[n] = \{P \in E(\mathbb{F}_q) : [n]P = \infty_E\}$.

For $E/\mathbb{F}_q$, the structure of $E[n]$ is well-understood using $E[ab] \cong E[a] \times E[b]$ if $(a,b) = 1$, and using [Sil09, III., Cor. 6.4.]:

$$E[n] \cong \begin{cases} \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}, & \text{if } p \nmid n; \\ \mathbb{Z}/p^e\mathbb{Z}, & \text{if } n = p^e \text{ and } p \nmid t; \\ 1 & \text{if } n = p^e \text{ and } p \mid t. \end{cases}$$

From this, we can further deduce:

Corollary 2.2.7. *For any q and any $E/\mathbb{F}_q$, there exist integers $d_1 \mid d_2$ s.t.*

$$E(\mathbb{F}_q) \cong \mathbb{Z}/d_1 \times \mathbb{Z}/d_2.$$

Proof. For any n, we have $E(\mathbb{F}_q)[n] \subset E[n]$. Now take $n = \#E(\mathbb{F}_q)$. □

We will re-derive this statement – and more – as a consequence of Lenstra's theorem on endomorphism rings Theorem 2.3.6: for supersingular elliptic curves in Example 2.3.7 and for ordinary curves in Section 2.5.3.

Finally, we set

$$E(\mathbb{F}_q)[n^\infty] = E(\mathbb{F}_q) \cap E[n^\infty]. \tag{2.7}$$

As we will see in Section 2.5, the structure of the torsion groups $E(\mathbb{F}_q)[n^\infty]$ is very tightly bound to the properties of n-isogenies of E.

2.2.2 Dual isogenies

Proposition 2.2.8 ([Sil09, III.6, Theorem 6.1])**.** *For any isogeny $\varphi : E \to E'$, there exist an isogeny $\hat{\varphi} : E' \to E$ with*

$$\hat{\varphi} \circ \varphi = [\deg \varphi].$$

Definition 2.2.9. The isogeny $\hat{\varphi}$ from Proposition 2.2.8 is called the *dual isogeny*.

Proof. In the separable case, the following (almost identical) proofs of Proposition 2.2.8 are useful. Since $\#\ker\varphi = \deg\varphi$, we have $\ker\varphi \subset E[\deg\varphi]$, and:

- the isogeny $[\deg\varphi]$ factors through the isogeny φ, from which we immediately obtain that $\hat{\varphi}$ is again $\mathbb{F}_q$-rational;
- if $E[\deg\varphi] = \langle P, Q\rangle$ is generated by two points P, Q, precomposing the isogeny $\psi : E' \to E''$ with kernel $\langle\varphi(P), \varphi(Q)\rangle$ with φ we obtain an isogeny with kernel $E[\deg\varphi]$, and therefore there exists an isogeny $\rho : E'' \to E$ such that $\rho \circ \psi \circ \varphi = [\deg\varphi]$, and we can set $\hat{\varphi} = \rho \circ \psi$; from which we obtain an explicit description of the kernel of a dual isogeny. □

Note that the dual isogeny has the same degree since $\deg([\deg\varphi]) = (\deg\varphi)^2$.

Definition 2.2.10. Let $N \in \mathbb{Z}$ be coprime to p. An isogeny $\varphi : E \to E'$ is called:

1. *cyclic*, if $\ker\varphi$ is a cyclic subgroup of E;
2. an *N-isogeny*, if it is cyclic of degree N.

Lemma 2.2.11. *The dual $\hat{\varphi}$ of an N-isogeny φ is again an N-isogeny.*

Proof. As the dual isogeny has the same degree, we only need to check that the kernel of $\hat{\varphi}$ is cyclic. As φ is cyclic, the kernel of $\hat{\varphi}$ is generated by some $P \in E[N]$. Let Q be a point on E such that $\langle P, Q\rangle = E[N]$. The dual isogeny $\hat{\varphi}$ has kernel $\ker\hat{\varphi} = \langle\varphi(P), \varphi(Q)\rangle = \langle\varphi(Q)\rangle$ as $\varphi(P) = \infty_{E'}$, and the claim follows. □

If ℓ is a prime, any isogeny of degree ℓ is cyclic.

2.2.3 Equivalent isogenies

Finally, we focus on a small discrepancy which makes the definitions of all isogeny graphs rather complicated. Being isogenous is a symmetric relation thanks to the existence of dual isogenies, so we would like to identify isogenies with their duals. At the same time, we wish to view isogenies as objects defined by their kernels, and *post-composing* with an automorphism of the target curve clearly does not change the kernel of the isogeny (we return to this in Section 2.6). But combining these two identifications causes a minor headache for curves with extra automorphisms.

Definition 2.2.12 (Equivalent isogenies)**.** Two separable isogenies $\varphi : E \to E'$ and $\psi : E \to E''$ with $E' \cong E''$ are *equivalent* if $\varphi = \rho \circ \psi$ for some $\rho : E'' \to E'$ of degree 1. We write $\varphi \sim \psi$.

Equivalent isogenies have the same kernel, and the target curves are isomorphic. But looking at the identification of isogenies with their duals, the symmetric relation is broken for elliptic curves with j-invariants $j = 0$ and $j = 1728$:

Example 2.2.13 (Automorphisms.)**.** If we precompose an isogeny $\varphi : E \to E'$ with an automorphism ρ of E, then we obtain an equivalent dual isogeny:

$$\widehat{\varphi \circ \rho} = \widehat{\rho} \circ \widehat{\varphi} \sim \widehat{\varphi}$$

Elliptic curves E with $j(E) \neq 0, 1728$ satisfy $\mathrm{Aut}(E) = \{\pm 1\}$. Precomposing with either of these automorphisms does not change the kernel of an isogeny, and so $\varphi \sim \varphi \circ \rho$ under Definition 2.2.12. Therefore, the mapping identifying an isogeny with its dual is one-to-one. However:

1. The elliptic curves with $j(E) = 1728$ admit an automorphism of order 4. In the model $E_{1728}/\mathbb{F}_q : y^2 = x^3 + x$ and with $i = \sqrt{-1} \in \overline{\mathbb{F}}_q$, such an automorphism is given as $\iota : (x, y) \mapsto (-x, iy)$.

   Therefore, if $i \in \mathbb{F}_q$, then there are *twice as many* inequivalent isogenies from E_{1728} than there are inequivalent isogenies with E_{1728} as target. This happens whenever $\mathbb{F}_q \supset \mathbb{F}_{p^2}$ or whenever $p \equiv 1 \bmod 4$.

2. The elliptic curves with $j(E) = 0$ admit an automorphism of order 6. In the model $E/\mathbb{F}_q : y^2 = x^3 + 1$ and with $\zeta_6 \in \overline{\mathbb{F}}_q$ satisfying $\zeta_6^3 = -1$ a primitive 6-th root of unity, such an automorphism is given as $\omega : (x, y) \mapsto (x, \zeta_6 y)$.

   Therefore, if $\zeta \in \mathbb{F}_q$, then there are *three times as many* inequivalent isogenies from E than there are inequivalent isogenies with E as target. This happens whenever $\mathbb{F}_q \supset \mathbb{F}_{p^2}$ or whenever $p \equiv 1 \bmod 3$.

Remark 2.2.14 (Identifying edges in isogeny graphs.)**.** In the following sections, we will define many different isogeny graphs, and all will require us to choose some identification of equivalent isogenies. To agree with the definition via roots of modular polynomials in Section 2.6.1, we will use Definition 2.2.12.

2.3 Endomorphism rings

Definition 2.3.1. An *endomorphism* of E is an isogeny $\varphi : E \to E$ or the zero map $[0] : E \to E$.

The zero map $[0]$ is not an isogeny (because it is a constant morphism) but we promote the zero map to an endomorphism so that the set of all endomorphisms of E has a ring structure. The addition on endomorphisms is given pointwise and multiplication is the composition of isogenies. Because we insisted that an isogeny needs to be defined over the field of definition of E, that is, over $\mathbb{F}_q$, our endomorphisms form the *rational endomorphism ring*

$$\mathrm{End}_{\mathbb{F}_q}(E) = \{\text{isogenies } \varphi : E \to E \text{ over } \mathbb{F}_q\} \cup \{0\}. \tag{2.8}$$

Studying the ring $\mathrm{End}_{\mathbb{F}_q}(E)$ is natural in isogeny-based cryptography, but E may admit further endomorphisms when considered as a curve over some extension field. We will see an example of this in Example 2.3.8. Considering E as an elliptic curve $E/\overline{\mathbb{F}}_q$, its endomorphism ring is denoted by $\mathrm{End}(E)$ and will in general not play a role in our discussion.

Beyond the scalar multiplication maps, one endomorphism which is always present is the Frobenius endomorphism.

Definition 2.3.2. The *Frobenius endomorphism* for $E/\mathbb{F}_q$ is the isogeny

$$\begin{aligned} \pi_q : E &\longrightarrow E \\ (x, y) &\longmapsto (x^q, y^q). \end{aligned}$$

The Frobenius endomorphism depends on the field of definition of E and is an isogeny of degree $\deg \pi = q$. If q is clear form the context, we write simply π.

Proposition 2.3.3 (Rational points)**.** *A point $P \in E$ is defined over $\mathbb{F}_{q^n}$ if and only if $\pi^n(P) = P$. That is, the groups $E(\mathbb{F}_{q^n})$ are precisely the kernels of $\pi^n - 1$.*

Theorem 2.3.4 ([Sil09, V., Thm. 2.3.1.])**.** *Let $E/\mathbb{F}_q$ be an elliptic curve and write $\#E(\mathbb{F}_q) = q + 1 - t$. The Frobenius endomorphim $\pi = \pi_E$ satisfies*

$$\pi^2 - [t]\pi + [q] = 0 \quad \text{in } \mathrm{End}_{\mathbb{F}_q}(E).$$

Therefore, we can identify the Frobenius with an algebraic integer that is a root of the polynomial $x^2 - tx + q$. This explains why the value t from Definition 2.1.3 is called the trace of Frobenius: it agrees with the usual notion of trace of an algebraic number. From Theorem 2.1.2, $|t| \leq 2\sqrt{q}$, and so $t^2 - 4q \leq 0$.

Theorem 2.3.5 (Waterhouse [Wat69])**.** *Let $E/\mathbb{F}_q$ be an elliptic curve and consider its rational endomorphism ring $\mathrm{End}_{\mathbb{F}_q}(E)$. There are only two possibilities:*

1. *If $t^2 - 4q < 0$ then $\mathbb{Q}(\pi)$ is an imaginary quadratic field and*

$$\mathrm{End}_{\mathbb{F}_q}(E) \hookrightarrow \mathbb{Q}(\pi) = \mathbb{Q}(\sqrt{t^2 - 4q})$$

   as an order $\mathcal{O}$ containing $\mathbb{Z}[\pi]$.

2. *If $t^2 - 4q = 0$ then $\pi = \pm\sqrt{q} = \pm p^{n/2}$ and*

$$\mathrm{End}_{\mathbb{F}_q}(E) \hookrightarrow B_{p,\infty}$$

   as a maximal order $\mathcal{O}$ in a quaternion algebra $B_{p,\infty}$ ramified at p and ∞.

Elliptic curves satisfying $\mathrm{End}(E) = \mathcal{O}$ for $\mathcal{O}$ an order in an imaginary quadratic field are said to have complex multiplication (CM) by $\mathcal{O}$.

There is a very tight connection between the group structure of $E(\mathbb{F}_q)$ and the endomorphism ring of $E/\mathbb{F}_q$.

Theorem 2.3.6 (Lenstra [Len96, Theorem 1])**.** *Let $E/\mathbb{F}_q$ be an elliptic curve, let $\pi = \pi_E$ be the Frobenius of E, and let $\mathcal{O} = \mathrm{End}_{\mathbb{F}_q}(E)$. Then*

1. *If $\pi \notin \mathbb{Z}$, then $E(\mathbb{F}_{q^n}) \cong \mathcal{O}/(\pi^n - 1)$.*

2. *If $\pi \in \mathbb{Z}$, then $E(\mathbb{F}_{q^n}) \cong \mathbb{Z}/(\pi^n - 1) \oplus \mathbb{Z}/(\pi^n - 1)$.*

Example 2.3.7 (Supersingular curves over $\mathbb{F}_p$)**.** Let $E/\mathbb{F}_p$ be supersingular, and let t be its trace. The only t with $|t| \leq 2\sqrt{p}$ such that $p \mid t$ is $t = 0$ (since we assume $p \geq 5$). Therefore, we are in the first case of Theorem 2.3.5.

We can identify the Frobenius with $\pi = \sqrt{-p}$ and so the endomorphism ring is either $\mathbb{Z}[\sqrt{-p}]$ or $\mathbb{Z}[\frac{1+\sqrt{-p}}{2}]$, as no other orders contain $\mathbb{Z}[\sqrt{-p}]$.

The group of rational points on E is either cyclic $E(\mathbb{F}_p) \cong \mathbb{Z}/(p+1)\mathbb{Z}$ or cyclic except for the 2-torsion: $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/(\frac{p+1}{2})\mathbb{Z}$. One can distinguish between the two, following [DG16]: if $\frac{\pi-1}{2} \in \mathrm{End}_{\mathbb{F}_p}$, then $[2]$ divides $\pi - 1$ in the endomorphism ring, which happens if and only if $E[2] \subset E(\mathbb{F}_p)$. A more general case will be discussed in Section 2.5.3.

Further, we consider the base change $E/\mathbb{F}_{p^2}$ and denote its new trace t_2. From Equation (2.3), we see that $t_2 = 0^2 - 2p = -2p$, so $t_2^2 - 4q = 0$, and we are in the second case of Theorem 2.3.5. We note that Frobenius then acts as a root of $x^2 + 2px + p^2 = (x+p)^2$, that is, as the scalar $\pi = [-p]$. Combining with Theorem 2.3.6, we see that

$$E(\mathbb{F}_{p^2}) \cong \mathbb{Z}/(-p-1) \times \mathbb{Z}/(-p-1) = \mathbb{Z}/(p+1) \times \mathbb{Z}/(p+1) \tag{2.9}$$

Example 2.3.8. We give examples for the various cases in Theorem 2.3.5.

1. The elliptic curve $E/\mathbb{F}_{31}$ given by $E : y^2 = x^3 + x + 4$: by counting points (in our favorite computer algebra system), we deduce

   $$\#E(\mathbb{F}_{31}) = 26 = 1 - 6 + 31,$$

   so $t = 6$ and $t^2 - 4q = -88 \neq 0$. The Frobenius satisfies $\pi^2 - 6\pi + 31 = 0$, so $\pi = 3 \pm \sqrt{-22}$. Therefore, $\mathrm{End}_{\mathbb{F}_{31}}(E)$ is an order in $\mathbb{Q}(\pi) = \mathbb{Q}(\sqrt{-22})$ containing $\mathbb{Z}[\pi] = \mathbb{Z}[\sqrt{-22}]$. But $\mathbb{Z}[\sqrt{-22}]$ is already maximal, so

   $$\mathrm{End}_{\mathbb{F}_{31}}(E) = \mathbb{Z}[\sqrt{-22}].$$

2. The elliptic curve $E/\mathbb{F}_{31}$ given by $E : y^2 = x^3 - x$: counting points gives

   $$\#E(\mathbb{F}_{31}) = 32 = 1 + 0 + 31$$

   and so $t = 0$ and $t^2 - 4q \neq 0$. So $\mathrm{End}_{\mathbb{F}_{31}}(E)$ is an order in $\mathbb{Q}(\pi) = \mathbb{Q}(\sqrt{-31})$ containing $\mathbb{Z}[\sqrt{-31}]$.

   Because $E[2] = \{(x,y) : x^3 - x = 0\} \cup \{\infty_E\}$, we see that all of $E[2]$ is defined over $\mathbb{F}_{31}$, and because of factorization of isogenies, $[2] \mid \pi - 1$. Therefore, there exist $\omega \in \mathrm{End}_{\mathbb{F}_{31}}(E)$ with $\omega = \frac{\pi-1}{2}$, corresponding to the algebraic element $\frac{-1+\sqrt{-31}}{2}$. As the order $\mathbb{Z}\left[\frac{-1+\sqrt{-31}}{2}\right]$ is already maximal,

   $$\mathrm{End}_{\mathbb{F}_{31}}(E) \cong \mathbb{Z}\left[\frac{1+\sqrt{-31}}{2}\right].$$

3. The elliptic curve $E/\mathbb{F}_{31^2}$ given by $E : y^2 = x^3 - x$: counting points we get

   $$\#E(\mathbb{F}_{31^2}) = 1024,$$

and so $t = -62 = -2 \cdot 31 = -2 \cdot \sqrt{q}$, and $t^2 - 4q = 0$. From Theorem 2.3.5, we see that $\mathrm{End}_{\mathbb{F}_{31^2}}(E)$ is a maximal order in the quaternion algebra $B_{31,\infty}$. One can show that it is isomorphic to the maximal order

$$\mathrm{End}_{\mathbb{F}_{31^2}}(E) \cong \mathbb{Z} + \mathbb{Z}i + \mathbb{Z}\frac{1+j}{2} + \mathbb{Z}\frac{i+ij}{2},$$

with $i^2 = -1$ and $j^2 = -31$ and $ij = -ji$.

2.4 Complex multiplication theory

Assume now that we are in the first case of Theorem 2.3.5: let E be an elliptic curve over $\mathbb{F}_q$ with $q+1-t$ points and assume that $t^2 - 4q \neq 0$, so $\mathrm{End}_{\mathbb{F}_q}(E) = \mathcal{O}$ is an order in an imaginary quadratic field $\mathbb{Q}(\pi)$ and $\mathcal{O}$ contains $\mathbb{Z}[\pi]$. This assumption covers all ordinary elliptic curves, and all supersingular elliptic curves over $\mathbb{F}_{p^k}$ with k odd. For the latter, the most interesting case is that of $E/\mathbb{F}_p$.

2.4.1 The group action

Isogenous curves have the same number of points (Theorem 2.2.3), hence the same trace, but not necessarily the same endomorphism ring. Isogenies that preserve the endomorphism ring come from invertible ideals in $\mathcal{O}$. Proofs for the claims in this section can be found in [Koh96] or [Wat69].

For any (nonzero) ideal $\mathfrak{a} \subset \mathcal{O}$ we can define a finite subgroup

$$E[\mathfrak{a}] = \cap_{\alpha \in \mathfrak{a}} \ker \alpha.$$

Definition 2.4.1. Let $\mathfrak{a}$ be an ideal of $\mathcal{O}$. The *isogeny corresponding to* $\mathfrak{a}$, denoted by $\varphi_\mathfrak{a} : E \to E/E[\mathfrak{a}]$, is the separable isogeny with $\ker(\varphi_\mathfrak{a}) = E[\mathfrak{a}]$.

If we do not impose the condition that $\varphi_\mathfrak{a}$ be separable, the isogeny with kernel $E[\mathfrak{a}]$ is only defined up to post-composition with purely inseparable isogenies.

Example 2.4.2. Take a prime ℓ and $\mathfrak{l} = (\ell, \pi - 1) \subset \mathcal{O}$. We want to identify the isogeny corresponding to $\mathfrak{l}$. So we need to intersect

- $\ker[\ell] = E[\ell]$ the subgroup of ℓ-torsion points,
- and $\ker(\pi - 1) = \{P : \pi(P) = P\} = \{(x, y) : x^q = x \text{ and } y^q = y\} = E(\mathbb{F}_q)$ the group of rational points.

So $E[\mathfrak{l}] = E[\ell] \cap E(\mathbb{F}_q) = E(\mathbb{F}_q)[\ell]$, that is, the $\mathbb{F}_q$-points in the ℓ-torsion. As abelian groups, $E[\ell] \cong \mathbb{Z}/\ell\mathbb{Z} \times \mathbb{Z}/\ell\mathbb{Z}$. So the action of $\mathfrak{l} = (\ell, \pi - 1)$ is given as:

1. If $E(\mathbb{F}_q)[\ell] = E[\ell]$, then $\varphi_\mathfrak{l}$ is multiplication by ℓ,

2. if $E(\mathbb{F}_q)[\ell] = \langle P \rangle$, then $\varphi_{\mathfrak{l}}$ is the ℓ-isogeny with kernel generated by any rational ℓ-torsion point P,

3. if $E(\mathbb{F}_q)[\ell] = \{\infty_E\}$, then $\varphi_{\mathfrak{l}}$ is the identity.

For an invertible ideal $\mathfrak{a}$ whose norm is not divisible by the characteristic p, the degree of the isogeny is $\deg \varphi_{\mathfrak{a}} = \mathrm{N}(\mathfrak{a})$.

Proposition 2.4.3. *If $\mathfrak{a} \subset \mathcal{O}$ is an invertible ideal, then $E/E[\mathfrak{a}]$ has the same endomorphism ring $\mathcal{O}$ and trace t as E.*

If $\mathfrak{a}$ and $\mathfrak{b}$ are in the same class in $\mathrm{Cl}(\mathcal{O})$, then

$$E/E[\mathfrak{a}] \cong E/E[\mathfrak{b}] \quad \text{over } \mathbb{F}_q.$$

Denote

$$\mathcal{Ell}_q(\mathcal{O}, t) = \{ \text{elliptic curves } E/\mathbb{F}_q : \mathrm{End}_{\mathbb{F}_q}(E) \cong \mathcal{O} \text{ and } \mathrm{tr}(\pi) = t \} / \cong_{\mathbb{F}_q}.$$

Away from j-invariants 0 and 1728, the curves in the set $\mathcal{Ell}_q(\mathcal{O}, t)$ for $t \neq 0$ can be represented by j-invariants: the quadratic twist $\tilde{E}$ has trace $\mathrm{tr}(\tilde{E}) = -\mathrm{tr}(E)$, so we have $\tilde{E} \in \mathcal{Ell}_q(\mathcal{O}, -t)$. For $j = 0$ or $j = 1728$, there may be non-isomorphic twists with the same trace. In the supersingular case over $\mathbb{F}_p$, the twist of a supersingular elliptic curve is again supersingular, so we represent $\mathcal{Ell}_p(\mathcal{O}, 0)$ by specific models of curves, for instance Montgomery models (Definition 4.2.5).

Theorem 2.4.4 (The CM group action)**.** *Let $\mathcal{O}$ be an imaginary quadratic order and let p be a prime that is not inert in $\mathcal{O}$. Assume further that t is such that $\mathcal{Ell}_q(\mathcal{O}, t)$ is non-empty.*

For any $E, E' \in \mathcal{Ell}_q(\mathcal{O}, t)$ there exists a unique class $[\mathfrak{a}] \in \mathrm{Cl}(\mathcal{O})$ such that

$$E' = [\mathfrak{a}] \star E.$$

The group $\mathrm{Cl}(\mathcal{O})$ acts on $\mathcal{Ell}_q(\mathcal{O}, t)$ freely and transitively by $([\mathfrak{a}], E) \mapsto [\mathfrak{a}] \star E$.

This theorem is essentially due to Deuring [Deu41], our exposition follows Waterhouse [Wat69, Thm. 4.5]. See also [Sch] for a discussion of the results. The restriction to p not being inert comes from [Sch, Thm. 4.5]; it is a rare care that only happens for supersingular curves for small p. In CSIDH (Chapter 4) the order we are most interested in is $\mathbb{Z}[\sqrt{-p}]$, in which p is ramified.

2.4.2 CM graph

From now on, assume that the action of $\mathrm{Cl}(\mathcal{O})$ on $\mathcal{Ell}_q(\mathcal{O}, t)$ is free and transitive. Such actions are also called *regular*.

Suppose that $\ell \nmid t^2 - 4q$ and factor $x^2 - tx + q = (x - \lambda)(x - \mu) \bmod \ell$, which means that the two prime ideals above ℓ in $\mathcal{O}$ are $(\ell, \pi - \lambda)$ and $(\ell, \pi - \mu)$.

Now examine the repeated action of the ideal $\mathfrak{l} = (\ell, \pi - \lambda)$ on $\mathcal{Ell}_q(\mathcal{O}, t)$. Since the ideal is split and has norm ℓ, the action corresponds to a sequence of ℓ-isogenies. If n is the order of $[\mathfrak{l}]$ in $\mathrm{Cl}(\mathcal{O})$, then $[\mathfrak{l}^n] = [\mathcal{O}]$ is principal and

$$E \xrightarrow{\mathfrak{l}} [\mathfrak{l}] \star E \xrightarrow{\mathfrak{l}} [\mathfrak{l}] \star ([\mathfrak{l}] \star E) = [\mathfrak{l}^2] \star E \xrightarrow{\mathfrak{l}} \ldots \xrightarrow{\mathfrak{l}} [\mathfrak{l}^n] \star E = E,$$

so the repeated action by $\mathfrak{l}$ cycles back to E after n steps.

Remark 2.4.5. Because $[\mathfrak{l} \cdot \bar{\mathfrak{l}}] = [(\ell)]$ is principal, acting by $[\bar{\mathfrak{l}}]$ is the same as traversing the cycle in the "opposite" direction.

From the main theorem of CM (Theorem 2.4.4) we immediately get:

Corollary 2.4.6. *The isogeny graph obtained by taking the curves $\mathcal{Ell}_q(\mathcal{O}, t)$ as vertices and edges if and only if $E_2 = [\mathfrak{l}] \star E_1$, is a union of cycles. The length of each cycle is the order of $\mathfrak{l}$ in $\mathrm{Cl}(\mathcal{O})$.*

Note that we can make the graph undirected thanks to Remark 2.4.5. We also consider a similar graph for several split ideals $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ lying above pairwise different primes $\ell_1, \ldots, \ell_n$.

Definition 2.4.7. Fix the field $\mathbb{F}_q$, trace t and order $\mathcal{O}$ and split ideals $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ lying above pairwise different primes $\ell_1, \ldots, \ell_n$. Define the following graph $\mathcal{G}$:

- the set of vertices is $\mathcal{Ell}_q(\mathcal{O}, t)$;
- there is an undirected edge between $[E_1]$ and $[E_2]$ with label ℓ_i if and only if $E_2 = [\mathfrak{l}_i] \star E_1$.

Remark 2.4.8 (Expander graphs.)**.** The graph $\mathcal{G}$ will be the conceptual basis for the CSIDH key exchange [CLM+18], which we will study in Chapter 4. Under the Generalized Riemann Hypothesis, the graph $\mathcal{G}$ defined in Definition 2.4.7 is an *expander graph* [JMV09]: roughly speaking, random walks of relatively short length have endpoints that are close to the uniform distribution on the vertices.

We reformulate this property in a simplified version for the action by ideals: If the ideals $\mathfrak{l}_i$ are chosen as all prime ideals in $\mathcal{O}$ lying above primes $\ell_i < \log(|\Delta_{\mathcal{O}}|)^c$ for some c, then random walks in the graph $\mathcal{G}$ of length $t = \tilde{O}(\log|\Delta|)$ yield nearly uniformly random elements in $\mathcal{Ell}_q(\mathcal{O}, t)$.

2.5 Isogeny volcanoes

This section is partially based on the background section of [CSV20] and [Sut13b].

The kernel of an ℓ-isogeny is a subgroup of size ℓ. However, we have

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

and so there are $\ell+1$ different subgroups of size ℓ in $E[\ell]$, each of them defining an ℓ-isogeny over some field. So, there are up to $\ell+1$ isogenies of degree ℓ defined over $\mathbb{F}_q$. We saw in Section 2.4.2 that at most 2 of them correspond to invertible ideals in $\mathrm{End}_{\mathbb{F}_q}(E)$. In this section, we study the remaining isogenies. We will always mean inequivalent isogenies in the sense of Section 2.2.3.

2.5.1 Isogenies and conductors

Consider an isogeny $\varphi : E \to E'$ of degree ℓ. Isogenous elliptic curves have the same trace (by Theorem 2.2.3), and so π_E and π'_E are both roots of the same polynomial $x^2 - tx + q$. Write $\mathcal{O} = \mathrm{End}_{\mathbb{F}_q}(E)$ and $\mathcal{O}' = \mathrm{End}_{\mathbb{F}_q}(E')$. Then they can both be identified with orders in the same quadratic field $\mathbb{Q}(\pi) = \mathbb{Q}\sqrt{t^2-4q})$. Both orders contain $\mathbb{Z}[\pi]$.

The isogeny φ can be used to explain how the two orders $\mathcal{O}$ and $\mathcal{O}'$ relate to each other. But first, we need a bit more information on imaginary quadratic orders, for which the reader is encouraged to consult the exposition in Kohel's thesis [Koh96, Chapter 4].

All orders $\mathcal{O}$ in an imaginary quadratic field K are suborders of the maximal order $\mathcal{O}_K$ with discriminant Δ_K. For any order $\mathcal{O}$, there exists a positive integer $f(\mathcal{O}) \in \mathbb{N}$, called the *conductor of* $\mathcal{O}$, such that $\Delta_{\mathcal{O}} = f^2\Delta_K$ and

$$\mathcal{O} = \mathbb{Z} + f\mathcal{O}_K.$$

From this description it is easy to see that, $\mathcal{O} \subset \mathcal{O}'$ if and only if $f(\mathcal{O}') \mid f(\mathcal{O})$. Therefore, the *lattice* of suborders of $\mathcal{O}_K$ (with lattice assuming its "other" standard meaning, a partially ordered set in which every pair of elements has a unique supremum) is anti-isomorphic to the lattice of positive integers, with divisibility.

Theorem 2.5.1 ([Koh96, Prop. 21])**.** *Let $E, E'/\mathbb{F}_q$ be elliptic curves. Suppose there exists an ℓ-isogeny $\varphi : E \to E'$, and denote the $\mathbb{F}_q$-rational endomorphism rings of E and E' by $\mathcal{O}$ and $\mathcal{O}'$, respectively. Then*

1. *either $\mathcal{O} = \mathcal{O}'$, in which case φ is called horizontal;*
2. *or $[\mathcal{O} : \mathcal{O}'] = \ell$, in which case φ is called descending;*
3. *or $[\mathcal{O}' : \mathcal{O}] = \ell$, in which case φ is called ascending.*

Note that a dual of an ascending isogeny is a descending isogeny, and vice versa. One can only have an ascending isogeny if $\ell \mid f(\mathcal{O})$, that is, if the prime ideal above ℓ is *singular* (that is, not invertible).

Proof. For any $\alpha' \in \mathcal{O}'$, the isogeny $\hat{\varphi} \circ \alpha_1 \circ \varphi$ is an endomorphism of E, so there is a homomorphism of rings $\iota_\varphi : \mathcal{O}' \to \mathcal{O} \otimes \mathbb{Q}$ as:

$$\alpha' \in \mathcal{O}' \mapsto \frac{1}{\deg \varphi} \cdot \hat{\varphi} \circ \alpha \circ \varphi \in \mathcal{O} \otimes \mathbb{Q} \tag{2.10}$$

By construction, since $\deg \varphi = \ell$, we have $\ell\iota(\mathcal{O}') \subset \mathcal{O}$. We can identify $\iota(\mathcal{O}')$ with its image in $\mathcal{O} \otimes \mathbb{Q} \cong \mathbb{Q}(\sqrt{t^2 - 4q})$; from now on we suppress writing ι. The situation is symmetric (replacing φ with $\hat{\varphi}$), so we obtain $\ell\mathcal{O} \subset \mathcal{O}'$, which implies

$$\mathbb{Z} + \ell^2\mathcal{O} \subset \mathbb{Z} + \ell\mathcal{O}' \subset \mathcal{O}.$$

Finally, we compare the conductors we get $f(\mathcal{O}) \mid \ell f(\mathcal{O}')$ and $f(\mathcal{O}') \mid \ell f(\mathcal{O})$. □

Remark 2.5.2 (Horizontal isogenies)**.** Horizontal isogenies are precisely those given by the class group action ([Koh96, Prop. 22], who cites Deuring [Deu41]).

2.5.2 Isogeny volcanoes

Theorem 2.5.1 shows that isogenies stratify elliptic curves, based on their endomorphism ring. So we consider all the elliptic curves with the same trace

$$\mathcal{E}\ell\ell(t) = \{E/\mathbb{F}_q : \mathrm{tr}_E = t\}/\cong_{\mathbb{F}_q} \tag{2.11}$$

$$= \bigcup_{\mathcal{O} \supset \mathbb{Z}[\pi]} \mathcal{E}\ell\ell_q(\mathcal{O}, t) \tag{2.12}$$

Definition 2.5.3 (ℓ-isogeny "ordinary" graph)**.** Let t be an integer, let $\mathbb{F}_q$ be a finite field, and suppose that $\mathcal{E}\ell\ell(t)$ is non-empty. We define a graph $G_{q,\ell}(t)$:

1. the vertices are given by curves in $\mathcal{E}\ell\ell(t)$,
2. there is a (directed) edge between vertices represented by curves E and E' for every ℓ-isogeny between E and E'.

We can make the graph undirected by identifying isogenies with their duals everywhere except at vertices with j-invariants 0 and 1728, see the discussion in Section 2.2.3. In the rest of the section, we restrict to the components not containing these curves, and remark on the general case in Remark 2.5.6. Denote by $\mathrm{val}_\ell(n)$ the ℓ-adic valuation of $n \in \mathbb{Z}$, that is, the largest power of ℓ dividing n.

Theorem 2.5.4 ([Koh96, Prop. 23])**.** *Let V be any connected component of $G_{q,\ell}(t)$ that does not contain curves with j-invariant $= 1728$ or $j = 0$. Then V is an* isogeny volcano*: there exists an integer h such that we can partition V into disjoint sets $V = V_0 \sqcup V_1 \sqcup \cdots \sqcup V_h$ such that*

- *the subgraph induced by vertices in V_h is a cycle, and is called the* surface*,*
- *the vertices in V_0 are called the* floor*,*
- *isogenies from surface to floor are descending,*
- *isogenies from floor to surface are ascending,*

- *the subgraph of V_i for $i \neq h$ has no edges,*
- *if $i < h$, every $E_i \in V_i$ has exactly one neighbor $E_{i+1} \in V_{i+1}$,*
- *for $i \neq 0$: every $E_i \in V_i$ has $\ell + 1$ neighbors.*

Moreover, the level of the volcano determines the endomorphism ring:

- *the elliptic curves on level i all have the same endomorphism ring $\mathcal{O}_i$, with discriminant $\Delta_{\mathcal{O}_i} = \ell^{2(h-i)}\Delta_{\mathcal{O}_h}$,*
- *the endomorphism ring $\mathcal{O}_h$ of the elliptic curves on the surface V_h is locally maximal at ℓ: if ℓ is odd then $\ell^2 \nmid \Delta_{\mathcal{O}_h}$, while if $\ell = 2$ and $4 \mid \Delta_{\mathcal{O}_h}$ then $\Delta_{\mathcal{O}_h}/4 \equiv 2, 3 \bmod 4$,*
- *the endomorphism ring $\mathcal{O}_0$ of the elliptic curves on the floor V_0 satisfies $\mathrm{val}_\ell(\Delta_{\mathcal{O}_0}) = \mathrm{val}_\ell(t^2 - 4q)$.*

In particular, if ℓ is odd then $h = \lfloor \frac{1}{2}\mathrm{val}_\ell(t^2 - 4q) \rfloor$, while if $\ell = 2$ then h may be 1 less than this value. Therefore, for a tall volcano, a large power of ℓ needs to divide the discriminant Δ_π. The set V_h is called the *surface* (or the *rim*, or the *crater*), the set V_0 is called the *floor*. Note that some authors flip the labelling so that V_0 is the surface and talk about the depth of the volcano instead.

Note that the only horizontal isogenies are permitted at the surface of the isogeny volcano. Indeed, after any descending isogeny, the primes above ℓ are no longer invertible, and we cannot have horizontal isogenies by Remark 2.5.2.

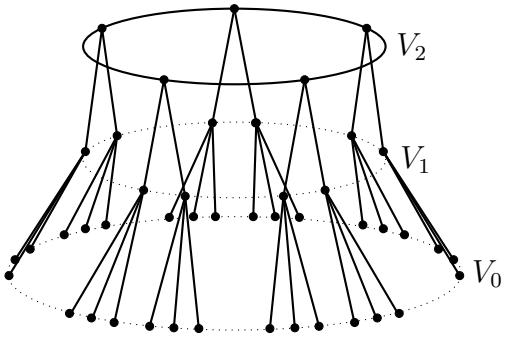


Figure 2.3: A 3-volcano from [CSV20]: height $h = 2$, together with its levels. This corresponds to the case where the prime 3 splits in $\mathcal{O}_h$, into two norm 3 prime ideals whose ideal-classes (which are each other's inverses) have order 5.

Example 2.5.5 (Supersingular volcanoes)**.** The supersingular volcanoes over $\mathbb{F}_p$ were first studied by Delfs and Galbraith [DG16]. Such volcanoes are rather modest. We always have $\Delta_{\mathbb{Z}[\pi]} = -p$ or $-4p$. So, for $\ell \neq 2$ we obtain disjoint

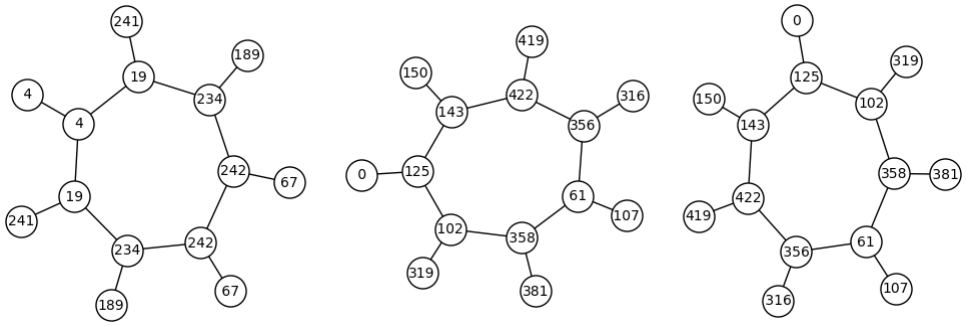


Figure 2.4: Aerial view of the volcanoes of 2-isogenies of $\mathcal{G}_2(\mathbb{F}_{431})$.

cycles as in Section 2.4.2, for $\ell = 2$ we obtain volcanoes of level at most 2. We denote these graphs $\mathcal{G}_\ell(\mathbb{F}_p)$, and will continue studying them in Chapter 3.

In Example 2.5.5, the curves on the surface admit 3 different 2-isogenies defined over $\mathbb{F}_p$. Since a 2-isogeny is rational if and only if its kernel point is rational, the curves on the surface have full 2-torsion, and the curves on the floor only one rational 2-torsion point. From Example 2.3.7, we see that they also have different endomorphism rings: the isogenies from the surface are *descending*.

Remark 2.5.6. If $j = 0$ or $j = 1728$ do appear in V, then Theorem 2.5.4 remains "sufficiently valid" for our purposes; the only difference is that $G_{q,m}(t)$ may become directed, see Section 2.2.3. The endomorphism rings of the curves with j-invariant 0 or 1728 have trivial class groups, so this remark only affects suborders of (certain) number fields having class number 1. Such suborders are usually not considered in isogeny-based cryptography, although they make an appearance in the recent OSIDH protocol due to Colò and Kohel [CK20] and the signature scheme SCALLOP [DFK+23].

2.5.3 Level structure

The last piece of information we need is how the levels of the volcano impact the level structure of the elliptic curves: we will study the torsion groups $E_i(\mathbb{F}_q)[\ell^\infty]$ for any $E_i \in V_i$. All elliptic curves $E \in \mathcal{E}\ell\ell(t)$ satisfy $\#E(\mathbb{F}_q) = 1 + q - t$, so setting $v = \mathrm{val}_\ell(1 - t + q)$, we see that

$$E_i(\mathbb{F}_q)[\ell^\infty] = \mathbb{Z}/\ell_1^v\mathbb{Z} \times \mathbb{Z}/\ell^{v_2}\mathbb{Z} \quad \text{with } v_1 + v_2 = v.$$

First, let us note some general facts. We know that as groups

$$E(\mathbb{F}_q) \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$$

with $n_1 \mid n_2$ (see Corollary 2.2.7). Since the product $n_1 n_2 = 1 - t + q$ is fixed, we deduce that we only need to determine the largest n_1 such that $E[n_1] \subset E(\mathbb{F}_q)$.

One restriction comes from the Weil pairing (for details, see [Sil09, III.8]): if $E[n] \subset E(\mathbb{F}_q)$ then $\mathbb{F}_q$ contains a non-trivial n-th root of unity and so $n \mid q - 1$.

Curves on the floor. First, we examine the curves E with endomorphism ring $\mathcal{O} = \mathbb{Z}[\pi]$. Since this is the smallest order allowed, they are clearly all on the floor of their respective volcanoes as no descending isogenies are possible.

Looking at the possible shape of the volcanoes, and noting that at most 2 isogenies can be non-descending, we deduce immediately that for odd ℓ, all curves on the floor of any volcano have cyclic torsion

$$E_0(\mathbb{F}_q)[\ell^\infty] = \mathbb{Z}/\ell^v\mathbb{Z}.$$

An alternative reasoning is by using Theorem 2.3.6: we have the map

$$\begin{aligned} \mathbb{Z}[\pi] &\to \mathbb{Z}/(1-t+q)\mathbb{Z} \\ \pi &\mapsto 1 \end{aligned}$$

which is a well-defined homomorphism of rings and factors through $\mathbb{Z}[\pi]/(\pi - 1)$ since $\pi^2 - t\pi + q \mapsto 1 - t + q = 0$ (remember that $\mathbb{Z}[\pi] \cong \mathbb{Z}[x]/(x^2 - tx + q)$). Comparing the sizes we get that the group of rational points

$$E(\mathbb{F}_q) \cong \mathbb{Z}[\pi]/(\pi - 1) \cong \mathbb{Z}/(1 - t + q)\mathbb{Z}$$

is cyclic of size $1 - t + q$. Therefore, $E(\mathbb{F}_q)[\ell^\infty] = (\mathbb{Z}/(1 - t + q)\mathbb{Z})_\ell = \mathbb{Z}/\ell^v\mathbb{Z}$.

Other levels. For other levels of the volcano, a similar direct homomorphism of rings quickly becomes rather daunting.

Instead, we again use the trick of factoring isogenies from Equation (2.6) (in the same way as in Example 2.3.7): for any n, we have $E[n] \subset E(\mathbb{F}_q)$ if and only if $\frac{\pi-1}{n} \in \mathrm{End}(E)$. The following description is mentioned in [Koh96], we follow the proof due to Wittmann [Wit01].

Write $\mathcal{O}_K = \mathbb{Z}[\omega]$ for some generator ω. Since $\mathbb{Z}[\pi] \subset \mathcal{O}_K$ with conductor f then $\mathbb{Z}[\pi] \subset \mathbb{Z} + f\mathcal{O}_K$, and so we can write $\pi = a + f\omega$ for some integer a.

The order of conductor g is the order $\mathcal{O} = \mathbb{Z} + g\mathcal{O}_K = \mathbb{Z}[g\omega]$. Rewriting

$$\frac{\pi - 1}{n} = \frac{a - 1 + f\omega}{n} = \frac{a-1}{n} + \frac{f}{n}\omega = \frac{a-1}{n} + \frac{f/g}{n}(g\omega)$$

we see that $\frac{\pi-1}{n} \in \mathcal{O}_g$ if and only if $n \mid a-1$ and $n \mid f/g$. The largest such integer is clearly $n_1 = \gcd(a - 1, f/g)$.

We conclude that for the elliptic curve with $\mathrm{End}(E) = \mathcal{O}$ of conductor $g \mid f$, the group structure is $E(\mathbb{F}_q) \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$ with $n_1 \mid n_2$ and:

$$n_1 = \gcd(a - 1, f/g), \tag{2.13}$$
$$n_1 \mid q - 1. \tag{2.14}$$

Notice that a only depends on the choice of ω, and is therefore independent of the level i. However, from Theorem 2.5.4, we see that curves on the same level of the same volcano have the same conductor, and hence the same torsion groups.

Further, Equation (2.13) implies that as we go up the volcano, we are allowing higher powers of ℓ in n_1, because the ratio of conductors f/g increases. We explain this further in Example 2.5.7; the general case is treated in Theorem 5.4.1 and is due to [Len96], [MMS+06, Cor. 1] for $m = 2$ and [MST+07, Thm. 3] for m odd.

Example 2.5.7. Assume that the curves on the floor have $\mathrm{End}(E_0) = \mathbb{Z}[\pi]$ and conductor f, assume that ℓ is odd, and that $v = \mathrm{val}_\ell(1-t+q)$ is even. On the floor, the ℓ^∞-torsion is cyclic. Then, one step up from the floor, the quotient $f/g = \ell$ is divisible by ℓ, so if $\ell \mid a-1$, we get full ℓ-torsion. Continuing up the volcano, the torsion continues to "balance" itself, see Example 2.5.7.

level	conductor	group structure	conditions
floor, $i = 0$	f	$\mathbb{Z}/\ell^v\mathbb{Z}$	
$i = 1$	f/ℓ	$\mathbb{Z}/\ell^{v-1}\mathbb{Z} \times \mathbb{Z}/\ell\mathbb{Z}$	
$i = 2$	f/ℓ^2	$\mathbb{Z}/\ell^{v-2}\mathbb{Z} \times \mathbb{Z}/\ell^2\mathbb{Z}$	
	$\vdots$		
i	f/ℓ^i	$\mathbb{Z}/\ell^{v-i}\mathbb{Z} \times \mathbb{Z}/\ell^i\mathbb{Z}$	$i \le v/2$
		$\mathbb{Z}/\ell^{v/2}\mathbb{Z} \times \mathbb{Z}/\ell^{v/2}\mathbb{Z}$	$i \ge v/2$
	$\vdots$		
surface, $i = h$	$f = 1$	$\mathbb{Z}/\ell^{v/2}\mathbb{Z} \times \mathbb{Z}/\ell^{v/2}\mathbb{Z}$	

Table 2.1: Group structure of elliptic curves on a volcano assuming curves in V_0 have endomorphism ring $\mathbb{Z}[\pi]$, the prime ℓ is odd and $v = \mathrm{val}_\ell(1-t+q)$ is even.

Finally, we note that the constant a determining the torsion is not a mysterious quantity. From [Koh96], we see that we can choose $a = t/2$ if $\Delta_K \equiv 0 \bmod 4$ and $a = \frac{t-f}{2}$ otherwise. Assume now that $a = t/2$ and ℓ is odd and remember that $v = \mathrm{val}_\ell(1-t+q)$ and $\ell^{2h} \mid f^2 \mid \Delta_\pi = t^2 - 4q$. We compute directly

$$(a-1)^2 = \frac{t^2-4t+4}{2} = \frac{t^2-4q}{2} + \frac{4(1-t+q)}{2}$$

and so the ℓ-adic valuation of $a-1$ connects the valuation of the number of points and the height of the volcano since $\mathrm{val}_\ell(a-1) \ge \frac{1}{2}\min\{2h + \mathrm{val}_\ell(\Delta_K), v\}$.

Remark 2.5.8 (Orientations)**.** Còlo and Kohel [CK20] introduce the volcano structure for supersingular elliptic curves over $\mathbb{F}_{p^2}$. For such curves, the endomorphism ring $\mathrm{End}(E)$ is a maximal order in a quaternion algebra $B_{p,\infty}$. As any element in $\mathrm{End}(E)$ is quadratic, $\mathrm{End}(E)$ contains *many* imaginary quadratic suborders $\mathcal{O} \subset \mathrm{End}(E)$. Moreover, any imaginary quadratic order $\mathcal{O}$ can be embedded into $\mathrm{End}(E)$ in multiple ways.

For an imaginary quadratic order $\mathcal{O}$ with field of fractions K, a field embedding $\iota : K \to B_{p,\infty}$ such that $\iota(\mathcal{O}) \subset \mathrm{End}(E)$ is called an *orientation* of E by $\mathcal{O}$.

An orientation ι is called *primitive* if $\iota(\mathcal{O}) = \iota(K) \cap \mathrm{End}(E)$. Suppose (E, ι) and (E', ι') are oriented by the same quadratic order $\mathcal{O}$. An isogeny $\varphi : E \to E'$ induces an orientation $\varphi_*(\iota)$ on E' via $\alpha \mapsto \frac{1}{\ell}\varphi \circ \iota(\alpha) \circ \hat{\varphi}$. We say that the isogeny φ is *oriented* if $\iota' = \varphi_*(\iota)$.

Onuki [Onu21] showed that under some mild conditions, the ideal class group of $\mathcal{O}$ acts on the set of elliptic curves primitively oriented by $\mathcal{O}$ (up to isomorphisms and Galois conjugacy), and that this action is free and transitive.

Moreover, if we consider oriented isogenies, the suborders of $\mathcal{O}$ allow for a similar volcano structure, but the volcanoes are infinite in this case.

2.6 Supersingular isogeny graphs

In this section, we restrict ourselves to the second case of Theorem 2.3.5: supersingular elliptic curves for which the endomorphism ring is a maximal order in a quaternion algebra. We will restrict ourselves to $\mathbb{F}_q = \mathbb{F}_{p^2}$: all supersingular elliptic curves admit a model over $\mathbb{F}_{p^2}$ (c.f. Remark 2.1.7).

We will not use the same tools as those used to study CM theory (Section 2.4.1, focusing on ideals in the endomorphism ring) or volcanoes (Section 2.5, focusing on the conductors, or torsion structure). We will study supersingular isogeny graphs from the viewpoint of modular polynomials.

This section is partially based on the background sections of [ACL+23].

2.6.1 Isogeny graphs

Defining modular polynomials is outside of the scope of this thesis; instead, we will use the fact that they relate isogenies and j-invariants:

Theorem 2.6.1 (Modular polynomials)**.** *For any integer N, there exists a polynomial $\Phi_N(X, Y) \in \mathbb{Z}[X, Y]$ with the following property:*

For E, E' elliptic curves, there exists a cyclic N-isogeny $\varphi : E \to E'$ if and only if $\Phi_N(j(E), j(E')) = 0$.

Clearly, Theorem 2.6.1 does not determine modular polynomials uniquely. For a proper definition see [Sil94, II., 6.3]. The polynomial Φ_N from this definition is called the N-th modular polynomial. The miracle of the modular polynomials is the subtle detail that the same modular polynomial Φ_N governs the isogenies of all elliptic curves, irrespective of the defining field.

The modular polynomial Φ_N is symmetric in X and Y. If $N = \ell$ is prime, the degree of Φ_ℓ in X and Y is $\ell+1$. The modular polynomial Φ_ℓ can be constructed in time $\tilde{O}(\ell^3)$ [Sut13a] and have been computed for $N > 60000$ in [BOS16, BLS11]; for $N < 1000$ they can be accessed as an online database [Sut].

The coefficients of modular polynomials grow for prime ℓ as $2^{\tilde{O}(\ell)}$, already the largest coefficient of $\Phi_2(X,Y)$ requires 48 bits to store:

$$\begin{aligned}\Phi_2(X,Y) = &- X^2Y^2 + X^3 + 1488X^2Y + 1488XY^2 + Y^3 - 162000X^2 \\ &+ 40773375XY - 162000Y^2 + 8748000000X + 8748000000Y \\ &- 157464000000000 \end{aligned} \tag{2.15}$$

For $\ell = 97$, the largest coefficients requires 5476 bits. The largest coefficient in database [Sut], for $\ell = 997$, requires 76672 bits to write down [BS, Table 1].

Remark 2.6.2 (Multiple isogenies)**.** The modular polynomial counts isogenies in the following way: for any elliptic curve E with $j = j(E)$, the roots of the polynomial $\Phi_N(j,Y) = \prod_{i=0}^{N}(Y - j_i)$, counted with multiplicity, correspond to inequivalent (cf. Section 2.2.3) cyclic N-isogenies from E to E_i with $j(E_i) = j_i$, counted with multiplicity.

Note that for $j(E)$ a supersingular j-invariant, the polynomial $\Phi_N(j(E),Y)$ splits completely over $\mathbb{F}_{p^2}$.

Now we are ready for the main definition of this section:

Definition 2.6.3 (Supersingular ℓ-isogeny graph)**.** Let ℓ be a prime, and let V be the set of supersingular j-invariants in $\mathbb{F}_{p^2}$. The graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ has vertex set V, and there is a directed edge from vertex j to j' for every root j' of the modular polynomial $\Phi_\ell(j,Y)$, considered with multiplicity.

As before, we can make this graph undirected at all vertices away from the j-invariants $j = 0, 1728$. These j-invariants are only supersingular if $p \equiv 2 \bmod 3$ and $p \equiv 3 \bmod 4$, respectively. For simplicity, we assume that we are in this undirected case $p \equiv 1 \bmod 12$.

Definition 2.6.4 (Adjacency matrix)**.** For a (possibly directed) graph G with vertices $V = \{v_1, \ldots, v_n\}$, the *adjacency matrix* A is the matrix $A = (a_{ij})$ indexed by $i, j = 1, \ldots, n$ with $a_{i,j} =$ number of edges from v_i to v_j.

Many properties of the graph are easily read off from the adjacency matrix. If G is undirected, the adjacency matrix is symmetric. The graph G has no loops if and only if $\mathrm{tr}(A) = 0$. If G is k-regular then k is the largest eigenvalue of A; the multiplicity of k as an eigenvalue is the number of connected components of G.

Formally, a path of length n in a graph G is a sequence of vertices and edges

$$(v_0, e_1, v_1, \ldots, e_n, v_n)$$

where $v_i \in V$ for $i = 0, \ldots, n$ are vertices, $e_j \in E$ for $j = 1, \ldots, n$ are edges, and every edge e_i is an edge between vertices v_i and v_{i+1}. Such a path is called a path between v_0 and v_n of length n; for simplicity, we will mostly consider paths as lists of vertices $(v_0, \ldots, v_n)$.

We call a path non-backtracking if for any $1 \leq j < n$ we have $e_i \neq e_{i+1}$, that is, if we do not immediately use the same edge to return to the previous vertex. In the isogeny world, walking back on an edge corresponds to taking a dual isogeny, so the non-backtracking walks correspond to cyclic isogenies.

Proposition 2.6.5 (Walks in G)**.** *Let G be a k-regular graph with adjacency matrix A. Consider the matrix $A^r = (\alpha_{i,j}^r)$. Then $\alpha_{i,j}^r$ is equal to the number of paths of length r from vertex v_i to vertex v_j.*

The number of non-backtracking paths from v_i to v_j of length r is given by the entries $\beta_{i,j}^r$ in $B_r = (\beta_{i,j}^r)$ defined recursively by $B_1 = 1$ and $B_2 = A^2 - k\boldsymbol{I}$ and

$$B_{r+1} = B_1 B_r - (k-1)B_{r-1}.$$

Proof. In the context of isogenies, see for instance [Ste22]. □

Specializing with $k = \ell + 1$, we recognize the same recurrence relation for the trace formula from Equation (2.3). This is not a coincidence but discussing the ubiquity of *Hecke operators* in number theory is out of the scope of this thesis.

Example 2.6.6. The author admits to being guilty of the following transgression of common sense: when showing examples, one might plot for instance the graph $\mathcal{G}_2(\mathbb{F}_{661})$ like Figure 2.5.

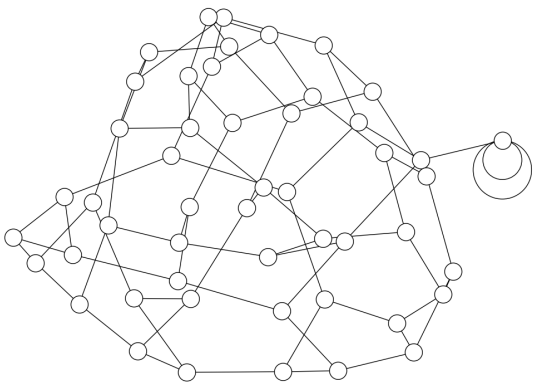


Figure 2.5: Supersingular isogeny graph $\mathcal{G}_2(\mathbb{F}_{661})$.

Since $601 \equiv 1 \pmod{12}$, the graph is 3-regular at all vertices, which is possible to verify with some effort, but very little else can be seen in this representation. We use *pixel graphs* instead (Figure 2.6): in the adjacency matrix A, each entry $a_{i,j}$ corresponds to a pixel at position (i, j), and this pixel is black if $a_{i,j} > 0$.

The matrices B_r give the number of non-backtracking paths of length t between the i-th and j-th vertex in G. Instead of plotting the graph for matrices B_r, we plot them for $\tilde{B}_r = \sum_{\rho=1}^{r} B_\rho$, which count the number of non-backtracking paths of length at most r. An example of such graphs for $\mathcal{G}_2(\mathbb{F}_{601})$ is in Figure 2.7.

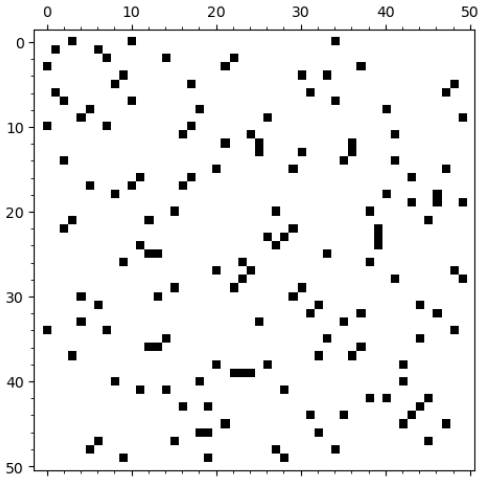


Figure 2.6: Pixel graph for the matrix A.

The pixel graph for $\tilde{B}_6$ is almost all black; all the pixel graph $\tilde{B}_r$ for $r \geq 7$ are full black (and so are for B_r with $r \geq 9$). The diameter (the maximum of the lengths of the shortest paths among any two vertices) of $\mathcal{G}_2(\mathbb{F}_{661})$ is $\delta = 7$. The example shows that the graphs have fairly short diameters; using different values $\alpha_{i,j}$ to show the number of paths between vertices v_i and v_j could be used to visualize the rapid mixing property (cf. Remark 2.6.8.

In Example 2.6.6, we saw that the graph $\mathcal{G}_2(\mathbb{F}_{661})$ has small diameter $\delta = 7$, compared to the number of vertices $\#V = \lfloor \frac{p-1}{12} \rfloor = 50$. This is a property shared by all isogeny graphs and will be one of the topics we study in Chapter 3.

We conclude the section by stating some of the properties of the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. For simplicity, we only state them for $p \equiv 1 \bmod 12$; however, most of the claims can be extended to the cases $p \not\equiv 1 \bmod 12$ ([BCC+23, Sec. 3.3].)

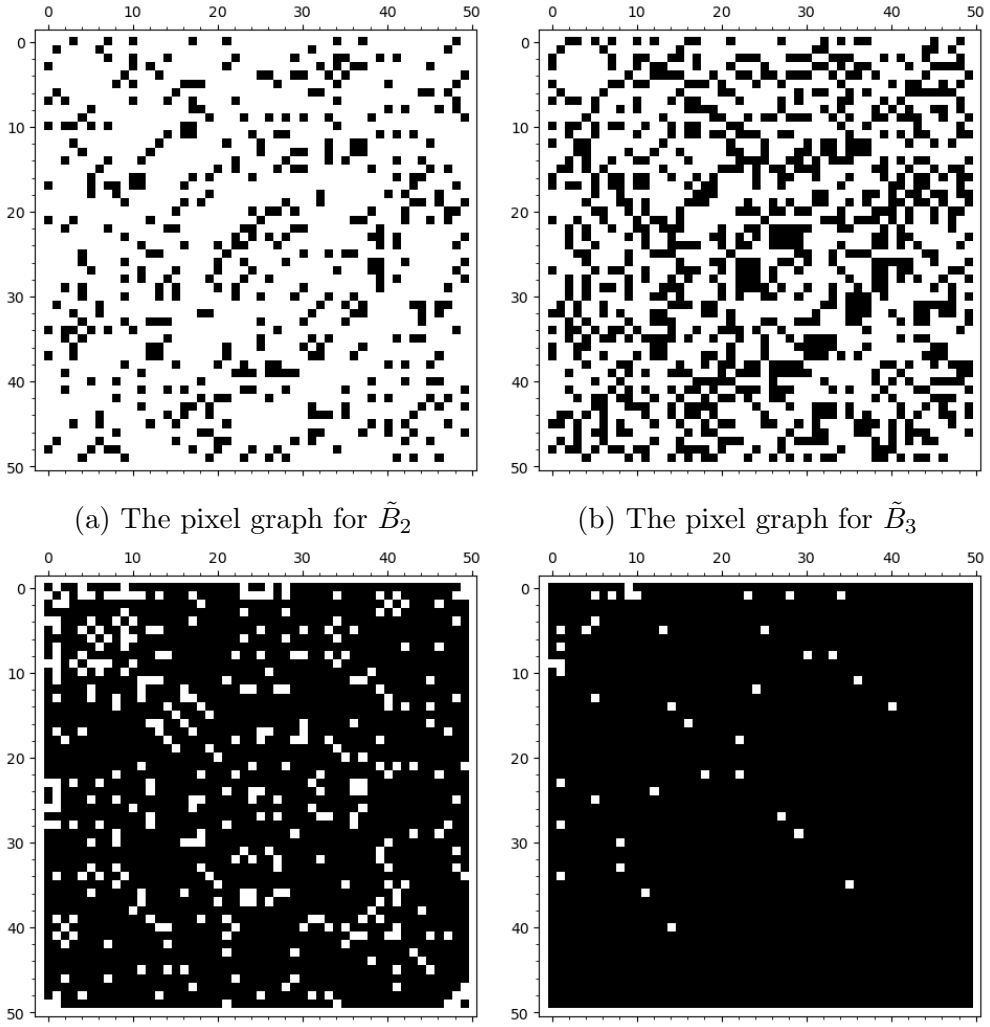
Proposition 2.6.7 (Properties of the isogeny graphs)**.** *Let $p \equiv 1 \bmod 12$, and let $\ell \neq p$ be a prime. Consider the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$:*

1. *The graph is a $(\ell+1)$-regular undirected graph and has $\lfloor \frac{p-1}{12} \rfloor$ vertices.*
2. *The graph is connected.*
3. *The* diameter of the graph

$$\delta = \min_{j_1 \neq j_2} \{\textit{length of the shortest path between } j_1, j_2\}$$

   satisfies $\delta = O(\log p)$, where the constant is independent of ℓ.

4. *The graph is Ramanujan: the second largest eigenvalue λ of its adjacency matrix A is bounded by $|\lambda| \leq 2\sqrt{\ell}$.*



(a) The pixel graph for $\tilde{B}_2$

(b) The pixel graph for $\tilde{B}_3$

(c) The pixel graph for $\tilde{B}_5$

(d) The pixel graph for $\tilde{B}_6$

Figure 2.7: Pixel graphs for $\mathcal{G}_2(\mathbb{F}_{601})$ and the matrices $\tilde{B}_2, \tilde{B}_3, \tilde{B}_5, \tilde{B}_6$.

Proof. The number of supersingular j-invariants can be computed as in [Sil09, Section V.4]. For connectedness, note that an even stronger claim is true: for any two elliptic curves E_1 and E_2 and a large enough n with $(p, n) = 1$, there exists an isogeny $E_1 \to E_2$ of degree n (see [Koh96, Cor. 77]). Setting $n = \ell^e$ for e large enough gives the claim. The claim about the diameter is [Koh96, Thm. 79], the Ramanujan property follows from [Mes86]. □

Remark 2.6.8 (Expander graphs)**.** Ramanujan graphs are expander graphs (cf. Remark 2.4.8). Supersingular isogeny graphs are therefore another example of graphs with *rapid mixing property*: the endpoints of random walks of sufficient length (about $\log_\ell p$) are close to being uniformly distributed in the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$.

This claim holds even for $p \not\equiv 1 \bmod 12$; the stationary distribution of the

random walk is the distribution in which all the elliptic curves are further weighed by the size of their automorphism groups $1/\#\operatorname{Aut}(E)$, see [BCC+23, Sec. 3.3].

2.6.2 Isogeny graphs in isogeny-based cryptography

The graphs $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ are examples of expander graps. In cryptography, they are typically used as follows: the public parameters are a prime p and a starting curve E_0. The secret is a description of a path in an isogeny graph, and the public information is the endpoint of this path.

We note that there are isogeny schemes that do not use the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$: for instance group-action based protocols like CSIDH [CLM+18]. These will be discussed in depth in Chapter 4.

CGL hash function. The first protocol to use the graphs $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ was the CGL hash function [CLG09]. For concreteness, take $\ell = 2$, then hashing a binary string x corresponds to choosing a path in the 3-regular graph $\mathcal{G}_2(\mathbb{F}_{p^2})$: beyond the initial step in the path, at any step i there are two isogenies that give a non-backtracking path, and one can deterministically select one of these depending on the i-th bit.

Pre-image resistance of the CGL hash function is equivalent to the path-finding problem: given an elliptic curve E_A, find a path from E_0 to E_A (possibly of given length). Collision registance is related to finding cycles in the graph $\mathcal{G}_2(\mathbb{F}_{p^2})$. While finding cycles in isogeny graphs is a hard problem, it is possible in specific instances, so the basic version of the CGL hash function is insecure [EHL+a].

SIDH. Until 2022, the fastest isogeny key-exchange scheme was SIDH [JD11], in particular in its optimized instantiation SIKE [SIK17]. In the language of isogeny graphs, Anouk and Bas would agree on a prime p of the shape $p = 2^a 3^b - 1$, and a starting supersingular curve E_0. Anouk would then take a random point R_A of order 2^a in $E_0(\mathbb{F}_{p^2})$, and compute the 2^a-isogeny $\varphi_A : E_0 \to E_A = E/\langle R_A \rangle$ as a sequence of a-many 2-isogenies. Bas would symmetrically compute an 3^b-isogeny described by some point R_B.

However, to allow the continuation of the Diffie–Hellman like key exchange, Bas needed to be able to compute the isogeny with kernel $\varphi_A(R_B)$ from E_A. This was achieved by Anouk revealing the images of a generating set P_3, Q_3 for $E_0[3^b]$. The belief was that this did not leak too much information about Anouk's isogeny.

This belief was proven wrong in 2022 by a sudden break by Castryck and Decru [CD23], followed independently by Maine and Martindale [MM22], and built upon by Robert [Rob23]. All these attacks use the images of the torsion points under the secret isogeny to reconstruct the whole isogeny.

The break has also lead to the new framework of evaluating large-degree isogenies [Rob22a] via embedding them into isogenies of higher-dimensional abelian varieties, and other applications [Rob22b].

SQISign Among the most recent schemes in isogeny-based cryptography is the signature scheme SQISign [DKL+20]. In this scheme, isogenies are selected by considering suitable ideals in the endomorphism ring $\mathrm{End}(E)$ inside the quaternion algebra $B_{p,\infty}$. Discussing this scheme is outside of scope for this thesis; we note that its security is not affected by the insecurity of SIDH.

Parameters and choices. In cryptography, rather than working with j-invariants, we prefer working with particular models of curves. For supersingular elliptic curves over $\mathbb{F}_{p^2}$, the only possibilities for the trace are $t = 0, \pm p, \pm 2p$.

Waterhouse [Wat69] shows that trace $t = \pm 2p$ is equivalent to all the endomorphisms of the elliptic curves being defined over $\mathbb{F}_{p^2}$. The trace $t = 0$ (resp. traces $t = \pm p$) correspond to quartic (resp. sextic) twists of the elliptic curves with j-invariant $j = 1728$ (resp. $j = 0$), and the resulting isogeny graphs have been described completely in [AAM19, Sec. 4]. We will focus on the case $t = \pm 2p$.

Since the difference in sign of the trace corresponds to twisting the curves, we will further only focus on $t = -2p$. In this case, Frobenius is a root of the polynomial $x^2 + 2px + p^2$, and hence acts as a scalar $\pi = [-p]$.

As we saw in Example 2.3.7, base-changing a supersingular curve $E/\mathbb{F}_p$ to a curve $E/\mathbb{F}_{p^2}$ produces an elliptic curve with trace $t = -2p$, without having to change the model. For instance, protocols which require a starting curve can make use of the curve $E_0 : y^2 = x^3 + x$ with j-invariant $j(E) = 1728$, which is supersingular for $p \equiv 3 \bmod 4$. This is one of the main reasons why we prefer to start from trace $-2p$ and not the analogous (twist component) with trace $t = 2p$. The graphs corresponding to the traces $t = \mp 2p$ were studied in depth in [AAM19].

Assuming the trace is $t = -2p$, Frobenius acts as the scalar $[-p]$, and all isogenies are necessarily defined over $\mathbb{F}_{p^2}$. However, from Lenstra's Theorem 2.3.6, we see that the group structure is $E(\mathbb{F}_{p^2}) \cong \frac{\mathbb{Z}}{(p+1)\mathbb{Z}} \times \frac{\mathbb{Z}}{(p+1)\mathbb{Z}}$; and so for any n-torsion we get: the group $E(\mathbb{F}_{p^2})[n]$ is either trivial or isomorphic to $\mathbb{Z}/n \times \mathbb{Z}/n$.

In isogeny-based cryptography, we like to put ourselves in the second case: we choose $n \mid p+1$, so that all the n-torsion points are already defined over $\mathbb{F}_{p^2}$. For instance, SIDH [JD11, DJP14] uses $p = 2^a 3^b \cdot f - 1$ for some large integers a, b and a small integer f, typically $f = 1$. The protocol uses 2^a- and 3^b-isogenies; their kernels can be represented by one point in $\mathbb{F}_{p^2}$ and they can be evaluated efficiently as a-many (resp. b-many) 2-isogenies (resp. 3-isogenies).

Remark 2.6.9 (Working with twists)**.** Costello [Cos20] noticed that we can in fact work in both the components with trace $t = \mp 2p$, the twist component allowing us to compute isogenies from twists $\tilde{E}$. The twist has $\#\tilde{E}(\mathbb{F}_{p^2}) = (p-1)^2$ points, and so we can compute isogenies of degrees $\mid p-1$ as rational isogenies, which is significantly cheaper than passing to extension fields and finding their kernels there. Since then, isogeny protocols using supersingular isogeny graphs

over $\mathbb{F}_{p^2}$ have taken advantage of primes for which $p^2 - 1$ has suitable smooth factors, see for instance [DKL+20].

Chapter 3

Adventures in Supersingularland

This chapter is based on the paper *Adventures in Supersingularland* [ACL+23], which is joint work with Sarah Arpin, Catalina Camacho-Navarro, Kristin Lauter, Joelle Lim, Kristina Nelson, and Travis Scholl and was published in Experimental Mathematics.

The introduction is new, and the background sections are omitted as they have largely been discussed in Section 2.6. The rest of the paper is edited for style and typographical adjustments. The definition of the "mirror involution" has been moved into Section 3.4. The section on related work has been expanded with a short discussion of subsequent work on some of the problems studied in this chapter, and is presented in Section 3.8.

3.1 Introduction

In this chapter, and in the paper on which this chapter is based [ACL+23], the main object of interest is the supersingular isogeny graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ defined in Definition 2.6.3. The vertices of this graph are the j-invariants of supersingular elliptic curves, and the edges between j-invariants j_1, j_2 describe the vanishing of $\Phi_\ell(j_1, j_2)$; counted with multiplicity, they correspond to the isogenies between any two elliptic curves with j-invariants j_1 and j_2.

Some of the "global" properties of the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ were given in Proposition 2.6.7. However, in this chapter, we are interested in studying the properties that make the graph – or some of its vertices – special, in a broad sense.

We start with a warm-up section Section 3.2 on the j-invariants that admit the simplest irregularities: where the graph cannot be made undirected ($j = 0, 1728$), which vertices admit loops, and which vertices admit double edges.

We encounter a more interesting subset of vertices of $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ in Section 3.3: the set of $\mathbb{F}_p$-rational vertices. We call the induced subgraph by these vertices the *spine* $\mathcal{S}$. The motivation behind studying the spine $\mathcal{S}$ is manifold: an attack by Delfs and Galbraith [DG16] on the path-finding problem in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ uses *navigating* to this set to find isogenies between supersingular elliptic curves (navigating means finding a path in the graph). Further, many isogeny-based protocols use an elliptic curve in $\mathbb{F}_p$ as a starting curve in the protocol.

We relate the spine $\mathcal{S}$ to the $\mathbb{F}_p$-rational graph $\mathcal{G}_\ell(\mathbb{F}_p)$ from Example 2.5.5. First studied in [DG16], we know that for ℓ odd, this graph is a union of cycles; for $\ell = 2$ it is a union of volcanoes. This structure describes isogenies defined over $\mathbb{F}_p$, and thus cannot disappear when we identify quadratic twists and pass to $\mathcal{S} \subset \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. The main results Section 3.3 are that $\mathcal{S}$ is formed from $\mathcal{G}_\ell(\mathbb{F}_p)$ via three possible operations: *stacking* of components with the same j-invariants, *folding* a component in half, and *attaching* components with *double* edges. This is shown in Theorem 3.3.16 for $\ell > 2$ and in Theorem 3.3.27 for $\ell = 2$. We conclude that $\mathcal{S}$ is far from a random subgraph of $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ because of the inherited volcanic structure, but this structure is remarkably preserved in the $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$.

Another aspect making $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ special is the following *mirror involution*: the Frobenius map $j \mapsto j^p$ swaps elliptic curves with $\mathbb{F}_{p^2}$-conjugate j-invariants, and fixes the spine $\mathcal{S}$. In Section 3.4, we are in particular interested in paths in the graph fixed by the mirror symmetry; and also study the shortest such paths: pairs of conjugate vertices that are ℓ-isogenous (Section 3.5).

We also experiment with isogeny graphs for small p and compute their diameters in Section 3.6. Throughout the experimental Sections 3.4 to 3.6, we notice different behavior depending on p mod 12.

Finally we include a discussion of related work in Section 3.8, including an analogous section from [ACL+23].

The code and data acompanying this article can be accessed via
https://github.com/krstnmnlsn/Adventures-in-Supersingularland-Data.

Notation and background. Assume that p is a prime $p \geq 5$ and ℓ is a prime. We allow $\ell = 2$, however, the cases $\ell = 2$ and ℓ odd are mostly treated separately throughout the chapter. We assume $\ell \neq p$.

In this chapter only, we will denote by E_j an elliptic curve with j-invariant j. We label the vertices of $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ (Definition 2.6.3) with j-invariants, but sometimes it is more convenient to talk about elliptic curves. By definition, every edge $[j, j']$ corresponds to an ℓ-isogeny $E_j \to E_{j'}$ for any fixed choice of E_j and $E_{j'}$.

3.2 Special j-invariants

First, we discuss j-invariants with special properties: j-invariants with extra automorphisms and then on j-invariants that are contained in very short cycles in $\mathcal{G}_\ell(\mathbb{F}_p)$: self-loops and cycles of length 2 (double edges).

Automorphisms. First, $j = 1728$ and $j = 0$ correspond to curves with extra automorphisms, which make the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ a priori directed since the in- and out-degrees of the vertices are different (for details, see Section 2.2.3).

Loops. Loops in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ are easily read off from the factorization of $\Phi_\ell(X, X)$: a vertex represented by a j-invariant j admits a loop if and only if $\Phi_\ell(j, j) = 0$. For $\ell = 2$, factoring over $\mathbb{Z}$ the polynomial $\Phi_2(X, Y)$ from (2.15) we get:

$$\Phi_2(X, X) = -(X + 3375)^2(X - 1728)(X - 8000). \tag{3.1}$$

Therefore, loops in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ can happen at $j = -3375$ (two loops), $j = 1728$ (one loop, see Example 3.3.5) and $j = 8000$ (one loop). These j-invariants need not be supersingular for a particular prime p.

Double edges. The following lemma describes double edges in the ℓ-isogeny graph (and note that these edges may in fact exist with higher multiplicity in the graph). This lemma applies *mutatis mutandis* to ordinary curves, replacing $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ by the ℓ-isogeny graph of ordinary elliptic curves.

Lemma 3.2.1 (Double edges)**.** *If there is a double edge between j-invariants j_1, j_2 in the ℓ-isogeny graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$, then j_1 and j_2 are both roots of the polynomial*

$$\mathrm{Res}_\ell(X) := \mathrm{Res}\left(\Phi_\ell(X, Y), \frac{d}{dY}\Phi_\ell(X, Y);\ Y\right). \tag{3.2}$$

The degree of $\mathrm{Res}_\ell(X)$ *is bounded by* $2\ell \cdot (2\ell - 1)$.

Proof. Suppose that j_1 and j_2 are two vertices in the ℓ-isogeny graph connected with a double edge. As a polynomial in Y, factor $\Phi_\ell(j_1, Y) = (Y - j_2)^2 \cdot g(j_1, Y)$

with some $g(j_1, Y) \in \overline{\mathbb{F}}_p[Y]$. The derivative $\frac{d}{dY}\Phi_\ell(j_1, Y)$ with respect to Y also vanishes at $Y = j_2$. So the polynomials $\Phi_\ell(X, Y)$ and $\frac{d}{dY}\Phi_\ell(X, Y)$ share the root $X = j_1$, and so by definition j_1 is a root of the resultant $\mathrm{Res}_\ell(X)$. This argument is also true for j_2, so j_2 is also a root of $\mathrm{Res}_\ell(X)$.

The total degree of $\Phi_\ell(X, Y)$ is 2ℓ, the total degree of $\frac{d}{dY}\Phi_\ell(X, Y)$ is $2\ell - 1$. Since the resultant of two polynomials $P(X, Y)$ and $Q(X, Y)$ of total degrees d and e generically has degree $d \cdot e$, the number of j-invariants that admit a double edge is bounded from above by $2\ell \cdot (2\ell - 1)$. □

Example 3.2.2 (Double edges for $\ell = 2$)**.** We factor $\mathrm{Res}_2(X)$ over $\mathbb{Z}$:

$$\mathrm{Res}_2(X) = -2^2 \cdot X^2 \cdot (X - 1728) \cdot (X + 3375)^2 \cdot (X^2 + 191025X - 121287375)^2 \quad (3.3)$$

Therefore, any double edge in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ can only appear between j-invariants from this list: $0, 1728, -3375$ or a root of $X^2 + 191025X - 121287375$.

By factoring $\Phi_2(j, X)$ for these values, we can for instance see that $j = 0$ is the only j-invariant that admits a triple edge in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$.

Remark 3.2.3 (Short cycles and tightness of the bound)**.** The bound on the degree in Lemma 3.2.1 is not tight. For example for $\ell = 2$, the bound is 12 whereas the polynomial has degree 9. A sharper estimate could be obtained as a sum of class numbers using an approach from [CLG09], which showed that this is a congruence condition on p. A double edge in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ gives a cycle of length 2 in the same graph. A short cycle of length n corresponds to an element of norm ℓ^n in the endomorphism rings of the curves whose j-invariants form this cycle. For instance, a double edge in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ gives a non-trivial quadratic element of norm 4 in $\mathrm{End}(E_j)$. The only imaginary quadratic fields that contain an element of norm 4 are $\mathbb{Q}(\sqrt{-3}), \mathbb{Q}(\sqrt{-1}), \mathbb{Q}(\sqrt{-7})$ and $\mathbb{Q}(\sqrt{-15})$. The factors of $\mathrm{Res}_2(X)$ are precisely the Hilbert class polynomials for these fields, and the roots are supersingular j-invariants if and only if p is inert in that field. Thus the degree of $\mathrm{Res}_2(X)$ can be bounded by the sum of the class numbers of these imaginary quadratic fields. For any given ℓ, there are at most 2ℓ such imaginary quadratic fields possible, all with discriminant bounded by $4\ell^2$. One can get a sharper bound by summing class numbers of these quadratic fields.

3.3 Structure of the $\mathbb{F}_p$-subgraph: the spine $\mathcal{S}$

Definition 3.3.1 (Spine)**.** The *spine* is the subgraph $\mathcal{S}$ of $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ induced by the vertices whose j-invariants lie in $\mathbb{F}_p$ and all edges between them in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$.

Many of the edges do correspond to isogenies defined over $\mathbb{F}_p$ but there may be edges only defined over $\overline{\mathbb{F}}_p$. In Section 3.3, we determine all such possible edges, and the size, shape, and number of connected components of $\mathcal{S}$ depending on the

class number of $\mathbb{Q}(\sqrt{-p})$. We also give some experimental data on the distance of connected components of $\mathcal{S}$ in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ in Section 3.3.6.

In this section, we assume that all elliptic curves are supersingular.

3.3.1 Structure of the $\mathbb{F}_p$-Graph $\mathcal{G}_\ell(\mathbb{F}_p)$

We start the section by discussing the isogeny graph $\mathcal{G}_\ell(\mathbb{F}_p)$ due to [DG16], which we discussed in Example 2.5.5. The two vertices in $\mathcal{G}_\ell(\mathbb{F}_p)$ with j-invariant a are labelled v_a, w_a; the vertex v_a, resp. w_a lies on the connected component V, resp. W. Note that V and W are not necessarily distinct. The vertex corresponding to a in $\mathcal{S}$ is denoted by a.

For visualization, we label the vertices of the graph $\mathcal{G}_\ell(\mathbb{F}_p)$ only by j-invariants.

Remark 3.3.2. We highlight the differences between $\mathcal{S}$ and $\mathcal{G}_\ell(\mathbb{F}_p)$:

- $\mathcal{S}$ has half as many vertices as $\mathcal{G}_\ell(\mathbb{F}_p)$, since the vertices in $\mathcal{S}$ are considered up to $\overline{\mathbb{F}}_p$-isomorphism and up to $\mathbb{F}_p$-isomorphism in the graph $\mathcal{G}_\ell(\mathbb{F}_p)$.
- $\mathcal{S}$ can have edges between j-invariants previously not connected in $\mathcal{G}_\ell(\mathbb{F}_p)$. However, we will show in Lemma 3.3.12 that this is a fairly rare occurrence.

An equivalent definition of supersingularity for elliptic curves over $\mathbb{F}_p$ was pointed out by [DG16]: $E/\mathbb{F}_p$ is supersingular if and only if $\mathbb{Z}[\sqrt{-p}] \subset \mathrm{End}_{\mathbb{F}_p}(E)$. In the volcano terminology (Section 2.5), the curves on the surface of the volcano, resp. floor, are those satisfying $\mathrm{End}_{\mathbb{F}_p}(E) = \mathcal{O}_K$, resp. $\mathrm{End}_{\mathbb{F}_p}(E) = \mathbb{Z}[\sqrt{-p}]$. For $p \equiv 1 \bmod 4$, the surface and floor coincide.

Recall the notation from Theorem 2.5.1: an isogeny $\varphi : E \to E'$ is *horizontal* if $\mathrm{End}_{\mathbb{F}_p}(E) \cong \mathrm{End}_{\mathbb{F}_p}(E')$, otherwise, we call the isogeny *vertical* (that is, the isogeny is ascending or descending). The following theorem shows that the connected components of $\mathcal{G}_\ell(\mathbb{F}_p)$ are volcanoes:

Theorem 3.3.3 ([DG16, Thm. 2.7])**.** *Let $p > 3$ and $\ell \neq p$ be primes.*

1. *For $\ell > 2$, there are no vertical isogenies. If $\left(\frac{-p}{\ell}\right) = 1$, then there are two horizontal ℓ-isogenies from each vertex and beyond this no other ℓ-isogenies. Hence every connected component of $\mathcal{G}_\ell(\mathbb{F}_p)$ is a cycle.*
2. *$p \equiv 1 \pmod 4$: all curves E satisfy $\mathrm{End}_{\mathbb{F}_p}(E) = \mathbb{Z}[\sqrt{-p}]$, so there is one level in $\mathcal{G}_\ell(\mathbb{F}_p)$. For $\ell = 2$: from each vertex there is one outgoing 2-isogeny. There are $h(-4p)$ vertices on the surface (which coincides with the floor).*
3. *$p \equiv 3 \pmod 4$: there are two levels in $\mathcal{G}_\ell(\mathbb{F}_p)$: surface and floor. For $\ell = 2$:*
   (a) *If $p \equiv 7 \pmod 8$: there is exactly one vertical isogeny from any vertex on the surface to the floor, vertices on the surface admit two horizontal isogenies. There are no horizontal isogenies on the floor. There are $h(-p)$ vertices on the floor and $h(-p)$ vertices on the surface.*

(b) *If $p \equiv 3 \pmod 8$: from every vertex on the surface, there are three vertical isogenies to the floor. There are no horizontal isogenies. There are $3 \cdot h(-p)$ vertices on the floor and $h(-p)$ vertices on the surface.*

Let $K = \mathbb{Q}(\sqrt{-p})$, $\mathfrak{p}$ be a prime above $\ell = 2$ in $\mathcal{O}_K$, and $h = \#\operatorname{Cl}(\mathcal{O}_K)$ the class number of K. Let n be the order of $\mathfrak{p}$ in $\operatorname{Cl}(\mathcal{O}_K)$. The surface of any volcano in $\mathcal{G}_2(\mathbb{F}_p)$ is a cycle of precisely n vertices. There are h/n connected components (volcanoes) in $\mathcal{G}_2(\mathbb{F}_p)$, the index of $\langle\mathfrak{p}\rangle$ in $\operatorname{Cl}(\mathcal{O}_K)$.

In Figure 3.1a, we see the graph $\mathcal{G}_2(\mathbb{F}_p)$ for $p = 431$ (case 3.a of Theorem 3.3.3). The vertices are labelled by j-invariants of each curve. Each component is a volcano, with an inner ring of surface curves and the outer vertices all being curves on the floor. The class number of $\mathbb{Q}(\sqrt{-431})$ is $3 \cdot 7 = 21$ and the orders of the two primes above 2 are 7.

Lemma 3.3.4 ([DG16])**.** *Let $p > 5$ and $E/\mathbb{F}_p$ be a supersingular elliptic curve. Then $\operatorname{End}_{\mathbb{F}_p}(E) = \mathbb{Z}\left[\frac{1+\sqrt{-p}}{2}\right]$ if and only if $E[2] \subset E(\mathbb{F}_p)$.*

Note that for $p \equiv 1 \pmod 4$, the ring $\mathbb{Z}\left[\frac{1+\sqrt{-p}}{2}\right]$ is not an order of $\mathcal{O}_K$, so no supersingular elliptic curves in $\mathbb{F}_p$ have their full 2-torsion defined over $\mathbb{F}_p$.

The following corollary will be essential in our discussion in Section 3.3.5.

Example 3.3.5 (The j-invariant 1728 is both on the surface and on the floor)**.** Suppose $p \equiv 3 \pmod 4$. Consider the isogeny $\varphi : y^2 = x^3 - x \to y^2 = x^3 + 4x$

$$(x, y) \mapsto \left(\frac{x^2 + x + 2}{x + 1}, y \cdot \frac{x^2 + 2x - 1}{x^2 + 2x + 1}\right).$$

Comparing the 2-torsion, we see that φ is a vertical 2-isogeny with kernel $(0, 0)$ of non-$\mathbb{F}_p$-isomorphic elliptic curves with j-invariant 1728.

This is the only example of a vertical isogeny between twists:

Corollary 3.3.6. *Let $p > 3$ be a prime, let $E/\mathbb{F}_p$ be an elliptic curve. Assume that $j(E) \neq 1728$ and let $\tilde{E}$ be its quadratic twist. Then $\operatorname{End}_{\mathbb{F}_p}(E) \cong \operatorname{End}_{\mathbb{F}_p}(\tilde{E})$.*

Proof. Suppose $E : y^2 = x^3 + ax + b$. Then $\tilde{E} : y^2 = x^3 + d^2ax + d^3b$ is a quadratic twist of E, where d is a quadratic non-residue. The isomorphism over $\overline{\mathbb{F}}_p$ is given as $(x, y) \mapsto \left(\frac{x}{d}, \frac{y}{d\sqrt{d}}\right)$. So $E[2] \subset E(\mathbb{F}_p)$ if and only if $\tilde{E}[2] \subset \tilde{E}(\mathbb{F}_p)$. □

Notice that in Example 3.3.5, we have $d = 2\sqrt{-1} \notin \mathbb{F}_p$, so the curve $y^2 = x^3 - x$ is a quartic twist of $y^2 = x^3 + 4x$.

Corollary 3.3.7. *If $j \neq 1728$, then the two distinct vertices corresponding to the j-invariant $j \in \mathbb{F}_p$ are either both on the floor or on the surface of $\mathcal{G}_\ell(\mathbb{F}_p)$.*

Another proof of this statement can be found in the appendix of [Kan89] and is obtained by a careful examination of Hilbert polynomials of discriminant $-p$ and $-4p$, considered modulo p.

3.3.2 Passing from the graph $\mathcal{G}_\ell(\mathbb{F}_p)$ to the spine $\mathcal{S} \subset \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$

We can obtain the spine $\mathcal{S}$ from the graph $\mathcal{G}_\ell(\mathbb{F}_p)$ in the following two steps:

1. Identify the vertices with the same j-invariant: these two vertices of $\mathcal{G}_\ell(\mathbb{F}_p)$ glue to a single vertex on $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. Identify equivalent edges.

2. Add the edges from $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ between vertices in $\mathbb{F}_p$ corresponding to isogenies which are defined over $\overline{\mathbb{F}}_p \setminus \mathbb{F}_p$.

The graph $\mathcal{G}_\ell(\mathbb{F}_p)$ is not a subgraph of $\mathcal{S}$, however, distinct edges from the same vertex in $\mathcal{G}_\ell(\mathbb{F}_p)$ correspond to distinct edges in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$:

Lemma 3.3.8. *Let E be an elliptic curve with $j(E) \notin \{0, 1728\}$ defined over $\mathbb{F}_p$ (with $p \geq 5$). Suppose that there are two ℓ-isogenies from E defined over $\mathbb{F}_p$. Then they are equivalent over $\mathbb{F}_p$ if and only if they are equivalent over $\overline{\mathbb{F}}_p$.*

Proof. If the isogenies are equivalent over $\mathbb{F}_p$ via a pair of $\mathbb{F}_p$-isomorphisms, then they are equivalent over $\overline{\mathbb{F}}_p$ as well.

Let $\phi_1 : E \to E'$ and $\phi_2 : E \to E'$ be two isogenies that are equivalent over $\overline{\mathbb{F}}_p$. We show that they are equivalent over $\mathbb{F}_p$. By assumption, there exist (over $\overline{\mathbb{F}}_p$) isomorphisms $\rho : E \to E$ and $\rho' : E' \to E'$ such $\rho' \circ \phi_1 = \phi_2 \circ \rho$. We know that the kernel of the map $\rho' \circ \phi_1$ is $\ker \phi_1$. Therefore, the kernel of the map $\phi_2 \circ \rho$ is also $\ker \phi_1$. This means that $\rho(\ker(\phi_1)) = \ker \phi_2$. By the hypothesis on $j(E)$, we have $\mathrm{Aut}_{\mathbb{F}_p}(E) = \mathrm{Aut}_{\overline{\mathbb{F}}_p}(E) = \{\pm 1\}$, and since $[\pm 1] \ker \phi_1 = \ker \phi_1$, it follows that $\ker \phi_1 = \ker \phi_2$. □

Lemma 3.3.8 for $\ell = 2$ gives the following corollary.

Corollary 3.3.9. *If an elliptic curve E with j-invariant j in $\mathbb{F}_p$ has 3 neighbors with j-invariants j_1, j_2 and j_3 (not necessarily distinct), then the 3 neighbors of j in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ have j-invariants j_1, j_2 and j_3.*

Example 3.3.10. ($\ell = 2$). If the vertex v_a has neighbors v_b, v_c, v_d in $\mathcal{G}_2(\mathbb{F}_p)$ (with b, c, d not necessarily distinct), then the neighbors of a in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ are b, c, d.

As there are at most 2 neighbors of any vertex in $\mathcal{G}_\ell(\mathbb{F}_p)$ for $\ell > 2$, the above corollary does not generalize. However, if the neighbors of v_a, w_a have j-invariants b, c, d, e, then there are (not necessarily distinct) edges $[a, b], [a, c], [a, d]$ and $[a, e]$ in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. In Lemma 3.3.15, we will show that typically $\{b, c\} = \{d, e\}$ and explain how to find j-invariants a for which this property fails.

Passing from $\mathcal{G}_\ell(\mathbb{F}_p)$ to the spine $\mathcal{S}$, the following can happen (and we will show that the list of events is complete):

Definition 3.3.11. Let V and W be connected components of $\mathcal{G}_\ell(\mathbb{F}_p)$.

1. We say that the components V and W *stack* if, when we relabel the vertices v_a by the j-invariant a, they are the same graph.

2. We say that the component V *folds* if V contains vertices corresponding to both quadratic twists of every j-invariant on V. The term is meant to invoke what happens to this component when the quadratic twists are identified in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$.

3. Two connected components V and W of $\mathcal{G}_\ell(\mathbb{F}_p)$ become *attached by a new edge* in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ if there is a new edge $[a, b] \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ corresponding to an isogeny between vertices $v_a \in V$ and $w_b \in W$ that is not defined over $\mathbb{F}_p$.

4. (for $\ell > 2$) We say that components V and W *attach along the j-invariant a* if they both contain a vertex with j-invariant a and the j-invariants of neighbors of v_a in V and the neighbors of w_a in W are not equal as multisets.

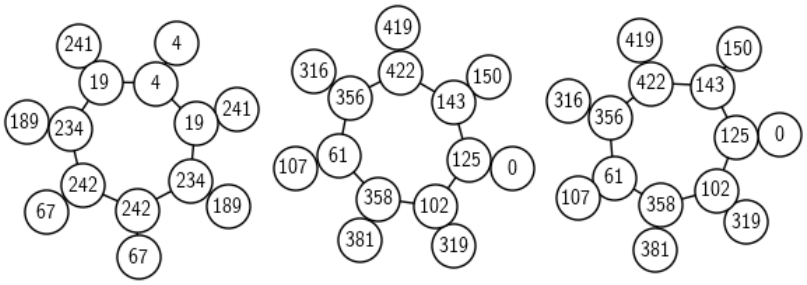
Examples of the first three of the four above are given by Figure 3.1. An example of attachment along a j-invariant can be seen in Figures 3.2 and 3.3. We will show that attachment along the j-invariant for $\ell = 2$ cannot happen in Corollary 3.3.22, however, the j-invariant 1728 is the only possible j-invariant for which the two vertices in $\mathcal{G}_2(\mathbb{F}_p)$ do not have the same neighbor set (Proposition 3.3.20), however these vertices are already connected by an edge.

We now begin analyzing the 'new' edges in $\mathcal{S}$: the edges that do not come from isogenies defined over $\mathbb{F}_p$. Let us look at Figure 3.1 again: the edges in $\mathcal{S}$ between vertices corresponding to j-invariants 150 and 189 in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ do not correspond to isogenies defined over $\mathbb{F}_p$. Also, the vertex 4 (and note $4 \equiv 1728 \pmod{431}$) on the floor of $\mathcal{G}_2(\mathbb{F}_p)$ has no edge j-invariant 19, but there is an edge $[4, 19] \in \mathcal{G}_2(\overline{\mathbb{F}}_p)$ coming from the two isogenies from vertex 4 on the surface. Lemma 3.3.8 gives us a double edge $[4, 19] \in \mathcal{G}_2(\overline{\mathbb{F}}_p)$. This not a coincidence:

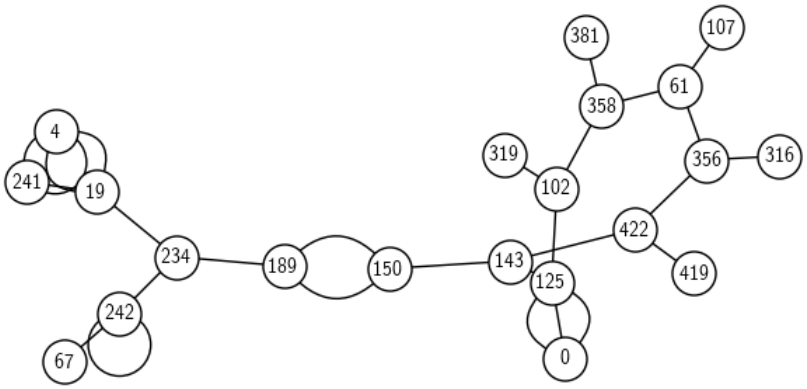
Lemma 3.3.12 (One new isogeny implies two new isogenies)**.** *Let $v_a, v_b \in \mathcal{G}_\ell(\mathbb{F}_p)$ correspond to $\mathbb{F}_p$-elliptic curves E_a and E_b with j-invariants a, b with $a \neq 1728, 0$. Assume that there is no edge $[v_a, v_b] \in \mathcal{G}_\ell(\mathbb{F}_p)$, but $[a, b] \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. Then there are two isogenies defined between $E_a \to E_b$ which are inequivalent over $\overline{\mathbb{F}}_p$ and hence a double edge $[a, b] \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$.*

Proof. We know that there is an ℓ-isogeny $\phi : E_a \to E$ to some elliptic curve E with j-invariant b. Since $j(E) = b$, then E is isomorphic to E_b over $\mathbb{F}_{p^2}$. Composing with this isomorphism, we obtain an ℓ-isogeny $\psi : E_a \to E_b$. But ψ cannot be defined over $\mathbb{F}_p$, since we assumed there was no edge $[v_a, v_b] \in \mathcal{G}_\ell(\mathbb{F}_p)$.

The kernel of ψ is not defined over $\mathbb{F}_p$ (otherwise ψ would be defined over $\mathbb{F}_p$), so the p-power Frobenius map $\mathrm{Frob} : \mathbb{F}_p \to \mathbb{F}_p$ does not preserve $\ker \psi$. There is an isogeny from E_a with kernel ψ^{Frob}. This isogeny has degree ℓ since ψ^{Frob} has order ℓ and it is not equivalent to ψ. Using Vélu's formulae [Vél71], we obtain the rational maps for defining ψ^{Frob}. In particular, the j-invariant of the target of ψ^{Frob} is necessarily $\mathrm{Frob}(b) = b$. Hence, there are two inequivalent isogenies between E_a and E_b and hence two edges $[a, b] \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. □

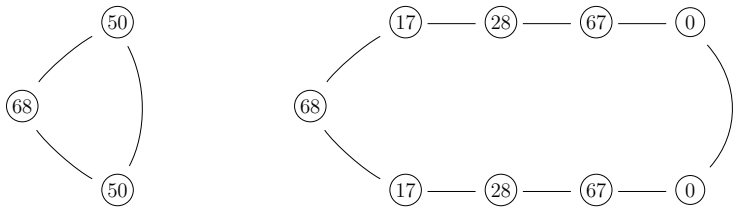


(a) The graph $\mathcal{G}_2(\mathbb{F}_p)$ for $p = 431$

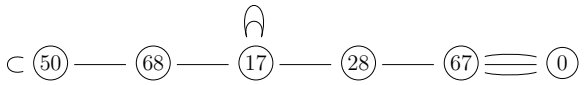


(b) The spine $\mathcal{S} \subset \mathcal{G}_2(\overline{\mathbb{F}}_p)$ for $p = 431$.

Figure 3.1: Stacking, folding and attaching by an edge for $p = 431$ and $\ell = 2$. The leftmost component of the graph $\mathcal{G}_2(\mathbb{F}_p)$ folds, the other two components stack, and the vertices 189 and 150 get attached by a double edge.

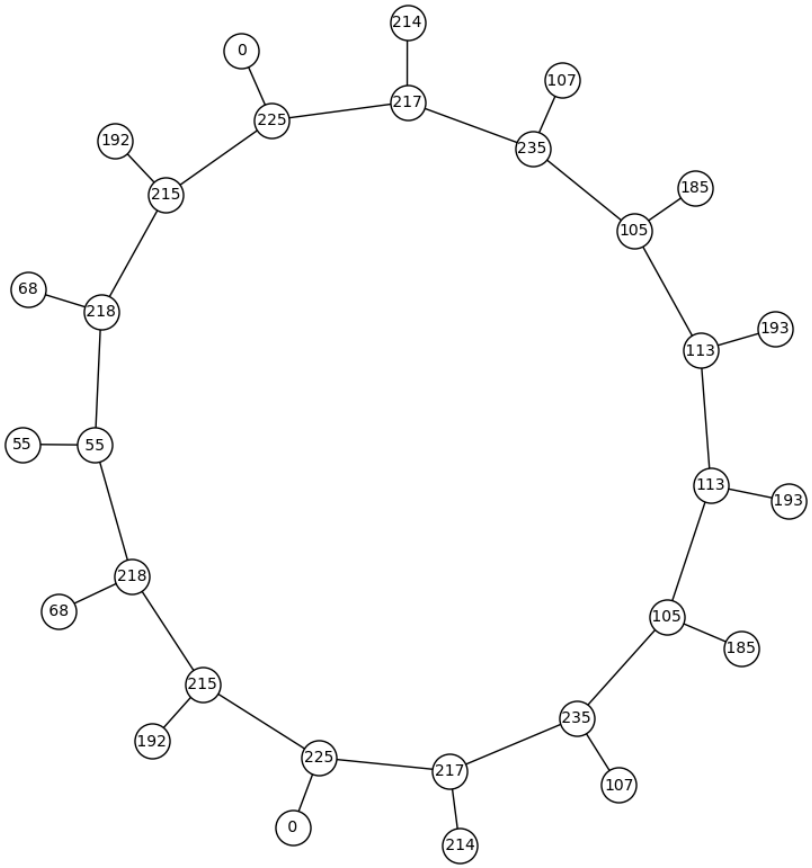


(a) The graph $\mathcal{G}_3(\mathbb{F}_p)$ for $p = 83$

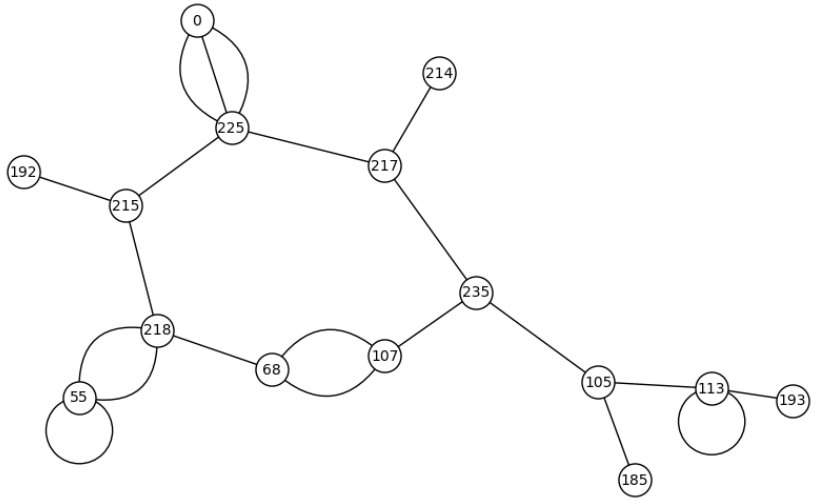


(b) The spine $\mathcal{S} \subset \mathcal{G}_3(\overline{\mathbb{F}_p})$ for $p = 83$.

Figure 3.2: Attachment along a j-invariant for $p = 83$ and $\ell = 3$. The two connected components of $\mathcal{G}_3(\mathbb{F}_p)$ are attached along $j = 68 = 1728 \bmod 83$. There are two outgoing double edges from $j = 1728$ but because of the extra automorphisms, these edges are identified in the undirected graph.



(a) The graph $\mathcal{G}_2(\mathbb{F}_p)$ for $p = 239$



(b) The spine $\mathcal{S} \subset \mathcal{G}_2(\overline{\mathbb{F}}_p)$ for $p = 239$.

Figure 3.3: Attachment by an edge that does not attach two distinct components. The component on the left folds and the vertices with j-invariants 68 and 107 are joined by a double edge. Notice the loop at 113, which is the vertex opposite of $55 \equiv 1728 \bmod p$. See Remark 3.3.17 for an explanation of this general behavior.

The corollary below explains why for both attachment by a new edge (cf. Figures 3.1 and 3.3) and along a j-invariant (cf. Figure 3.2), we see double edges.

Corollary 3.3.13. *Attachment of components by a new edge from v_a to w_b forces a double edge $[a, b]$ in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. Attachment along the j-invariant j implies a double edge from j in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$.*

Proof. In the first case, we are adding an edge between v_a and w_b that is not defined over $\mathbb{F}_p$ and we can apply Lemma 3.3.12. In the second case, assume there is a neighbor v_b of v_j such that w_b is not a neighbor of w_j. Applying Lemma 3.3.12 to the $\overline{\mathbb{F}}_p$ isogeny from w_j to v_b. □

Because we display the isogeny graphs undirected, the resulting graphs are not regular at the vertices corresponding to $j = 1728$ and $j = 0$.

3.3.3 The spine for $\ell > 2$

For this section, we consider the spine $\mathcal{S}$ for $\ell > 2$. In this case, there are no vertical isogenies, hence the graph $\mathcal{G}_\ell(\mathbb{F}_p)$ is a union of disjoint cycles: the cycles of vertices corresponding to curves either only with endomorphism ring $\mathbb{Z}\left[\frac{1+\sqrt{-p}}{2}\right]$, or only with endomorphism ring $\mathbb{Z}[\sqrt{-p}]$. The main tool for understanding the spine will be the neighbor Lemma 3.3.15, in which we show that the vertices v_a and w_a have neighbors with the same j-invariants, provided $a \neq 1728$. If the neighbors have different j-invariants, we would get attachment along a vertex and this situation we solve in Proposition 3.3.14. Using the information about the shape of the graph $\mathcal{G}_\ell(\mathbb{F}_p)$ and the neighbor lemma, we can completely describe the spine $\mathcal{S}$ in Theorem 3.3.16.

We will avoid the case when the graph $\mathcal{G}_\ell(\mathbb{F}_p)$ is just a disjoint union of vertices (i.e., when there are no isogenies defined over $\mathbb{F}_p$) by assuming ℓ is split in $\mathbb{Z}[\sqrt{-p}]$. If one assumes that $p \equiv -1 \pmod{\ell}$, then $\ell | \#E(\mathbb{F}_p) = p + 1$ and there are $\mathbb{F}_p$-rational points of order ℓ. We will also assume that the class number $\mathrm{Cl}(\mathbb{Z}[\sqrt{-p}])$ is odd, which is equivalent to $p \equiv 3 \pmod{4}$.

Proposition 3.3.14 (Attachment along a vertex)**.** *Let p and ℓ be primes satisfying $\ell^4 < \frac{1}{4}p$. Let j be a j-invariant such that attachment along a vertex happens. Then $j = 1728$, the two neighbors in $\mathcal{G}_\ell(\mathbb{F}_p)$ of v_{1728} have the same j-invariant and the neighbors of w_{1728} have the same j-invariant.*

The proof below uses the properties of the $\overline{\mathbb{F}}_p$-endomorphism ring $\mathrm{End}(E)$ as a maximal order in a quaternion algebra. For references see [Koh96, Kan89, Ibu82].

Proof. We first show that j corresponds to a curve with automorphism group larger than $\{\pm 1\}$, so $j = 1728$ or 0. Then we show that $j = 0$ cannot happen.

Let E be a supersingular elliptic curve over $\mathbb{F}_{p^2}$, then $\mathrm{End}(E)$ is a maximal order in a quaternion algebra (cf. Theorem 2.3.5). Every element $\alpha \in \mathrm{End}(E) \setminus \mathbb{Q}$

generates a quadratic number field $\mathbb{Q}(\alpha)$. From [Kan89, Theorem 2'], if O, O' are maximal quadratic orders lying in different imaginary quadratic fields that embed to $\mathrm{End}(E)$ then $\operatorname{disc} O \cdot \operatorname{disc} O' \geq 4p$.

Suppose now that we have attachment at a vertex j. Let v_a, v_b be the neighbors of v_j and let w_c, w_d be the neighbors of w_j, with $a \notin \{c, d\}$. Then, by Corollary 3.3.13, there is a double edge from $[a, j] \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ and, by symmetry, a double edge from either c or d. Assume it is from c. Any double edge gives a non-trivial element of norm ℓ^2 in $\mathrm{End}(E)$. Let α denote the cycle arising from the double edge at a, and γ the cycle arising from the double edge at c.

Suppose that α and γ generate different quadratic number fields. Then we compute that $\ell^2 = \operatorname{nrd}(\alpha) \geq \frac{1}{4} \operatorname{disc} \mathbb{Z}[\alpha] \geq \frac{1}{4} \operatorname{disc}(\mathbb{Q}(\alpha))$ and we obtain a similar statement for γ. Using Kaneko's theorem we deduce that

$$\ell^2 \cdot \ell^2 = \operatorname{nrd} \alpha \cdot \operatorname{nrd} \beta \geq \frac{1}{16} \operatorname{disc}(\mathbb{Q}(\alpha)) \cdot \operatorname{disc}(\mathbb{Q}(\gamma)) \geq \frac{1}{4} p,$$

which is impossible for $\ell^4 < \frac{1}{4}p$. So α and γ generate the same field, say K. Let $\mathcal{O}_K$ be the ring of integers in K. By uniqueness of ideal factorization:

$$(\alpha)(\overline{\alpha}) = (\ell^2) = (\gamma)(\overline{\gamma}) = (\ell)^2 = \begin{cases} (\ell)^2 & \text{if } \ell \text{ is inert,} \\ \mathfrak{l}^4 & \text{if } \ell \text{ is ramified,} \\ (\mathfrak{l}\bar{\mathfrak{l}})^2 & \text{if } \ell \text{ splits.} \end{cases}$$

Since $\alpha \notin \mathbb{Q}$, ℓ cannot be inert. If ℓ is ramified, we see: $(\alpha) = (\gamma) = \mathfrak{l}^2$ and therefore $\alpha = u\gamma$ for some unit $u \in \mathcal{O}_K$. It is not possible that $\alpha = \pm\gamma$, so necessarily $\{\pm 1\} \subsetneq \mathrm{Aut}(E)$.

If ℓ splits, suppose without loss of generality that $(\alpha) = \mathfrak{l}^2$. Then either

$$(\gamma) = \mathfrak{l}^2 \longrightarrow \gamma = u \cdot \alpha$$
$$\text{or} \quad (\gamma) = \bar{\mathfrak{l}}^2 \longrightarrow \overline{\gamma} = u \cdot \alpha$$

for some unit $u \in \mathcal{O}_K$. Again $\mathcal{O}_K$ has non-trivial units and $\{\pm 1\} \subsetneq \mathrm{Aut}(E)$.

Curves with automorphism group larger than $\{\pm 1\}$ correspond to $j = 0, 1728$. Suppose $j(E_0) = 0$. If there is an isogeny from an elliptic curve E_0 to an elliptic curve E_a with j-invariant a, there have to be at least three distinct isogenies from E_0 to E_a: precompose the first one with $\zeta_3, \zeta_3^2 \in \mathrm{End}(E_0)$. However, suppose that there were three edges from 0 to a. By symmetry, there would be three edges from 0 to c in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. By combining these edges, we would have 6 cycles from $j = 0$ of length 2. But even in $\mathbb{Z}[\zeta_3]$, up to sign $(\pm)$, there are not enough different elements of norm ℓ^2.

Hence attachment along a vertex can only happen for $j = 1728$. □

Lemma 3.3.15 (The neighbor lemma)**.** *Suppose $\ell > 2$. Let $a \neq 1728$ and suppose that v_a, w_a are the two vertices in $\mathcal{G}_\ell(\mathbb{F}_p)$ corresponding to elliptic curves with j-invariant a. If the neighbors of v_a have j-invariants b, c, then then neighbors of w_a have j-invariants b, c.*

Proof. Let w_d, w_e be the neighbors of w_a in $\mathcal{G}_\ell(\mathbb{F}_p)$. If $\{d, e\} \neq \{b, c\}$, then there is attachment along the j-invariant a, which is only possible for $j = 1728$. Hence the neighbors of v_a and w_a have the same j-invariants. □

Recall that the attachment of components by a new edge implies a double edge by Corollary 3.3.13. The main result of this section is the following one.

Theorem 3.3.16 (Stacking, folding and attaching for $\ell > 2$)**.** *Let p be a prime and let ℓ be such that the order of the prime $\mathfrak{l}$ above ℓ in the class group $\mathrm{Cl}(-p)$ of $\mathbb{Q}(\sqrt{-p})$ is odd. While passing from $\mathcal{G}_\ell(\mathbb{F}_p)$ to $\mathcal{S}$, the following happens:*

1. *the components containing 1728 fold and get attached along $j = 1728$; this only happens if $p \equiv 3 \pmod 4$;*

2. *all other components stack;*

3. *the number of new edges is bounded by $\deg \mathrm{Res}_\ell(X)$ and there are at most n vertices at which a new edge is added, where n is the number of distinct roots of $\mathrm{Res}_\ell(X)$.*

Note that the conditions on p and ℓ are always satisfied when $p \equiv 3 \pmod 4$ and $p \equiv -1 \pmod \ell$, which is the most cryptographically relevant application.

Proof. We will pass from $\mathcal{G}_\ell(\mathbb{F}_p)$ to $\mathcal{S}$ in two steps: first, we identify the j-invariants and equivalent edges in $\mathcal{G}_\ell(\mathbb{F}_p)$, and after discussing what happens with the components of $\mathcal{G}_\ell(\mathbb{F}_p)$, we add the new edges in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$.

We showed in Lemma 3.3.15 that for $a \neq 1728$, the vertices v_a and w_a have the same neighbors.

Suppose there is a component V that does not stack. This either means that there is a vertex v_a whose neighbors are different than those of w_a. But in this case a is an attaching j-invariant and so $a = 1728$, which we treat later.

Or, there is a j-invariant a such that both the vertices v_a, w_a are in the component V. But V is a cycle with odd number of vertices. The vertices v_a, w_a divide the cycle in two halves. Choose either half H. Look at the neighbors of v_a and w_a. If they have the same j-invariant b, replace a with b and continue along the halves of the cycle, until either of the following happens:

(i) the vertices v_a and w_a are neighbors in $\mathcal{G}_\ell(\mathbb{F}_p)$ and hence will induce a loop in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. Note that then the number of vertices in H is necessarily even.

(ii) The only neighbor of v_a and w_a is a vertex v_j with j-invariant j. This is necessarily an attaching j-invariant as the neighbors of w_j cannot have j-invariants a. Hence $j = 1728$. This also means that $\#H$ is odd.

(iii) The neighbor v_b of v_a has j-invariant b and the neighbor w_c of w_a has j-invariant c, for $b \neq c$. Then either the other neighbor of w_a is w_b or a is an attaching j-invariant (and $a = 1728$). Suppose a is not an attaching j-invariant. Continuing along the whole cycle V in the direction of the edge $[v_a, v_b]$, and symmetrically in the direction of $[w_a, w_b]$, we reach a point at which the neighbor of some v_c is not a neighbor of the w_c. This happens when the cycle has an odd number of vertices since two identical paths would give us a cycle of even length. Here we also obtain an attaching j-invariant.

The cases above actually cover every possibility: if we are in case (Step i), the half H has an even number of vertices, and then $V \setminus H \cup \{v_a, w_a\}$ necessarily has an odd number of vertices and since the neighbors of v_a, w_a have to have the same j-invariant, we are in case (Step ii).

Let us remark that the last case (Step iii) shows that the isogenies between twists go 'in the other direction': if one walks on the cycle V in the direction of the edge $[v_a, v_b]$, one will encounter an edge $[w_b, w_a]$, not an edge $[w_a, w_b]$.

We now discuss what happens with the components that contain 1728. We know that there are two components V and W containing j-invariant 1728. Let $v_a \in V$ be on the surface and let $w_a \in W$ be on the floor. We have shown in Proposition 3.3.14 that both v_{1728} and w_{1728} have two neighbors with the same j-invariant. We show that both the components fold.

Starting at the vertex $v_{1728} \in V$, we know that its neighbors v_a, w_a have the same j-invariant by Proposition 3.3.14. Start walking away from v_{1728}. By Lemma 3.3.15, the neighbors of the vertices v_a, w_a that are different from v_{1728} have the same j-invariants. Since the cycle contains an odd number of vertices, one finally arrives at a pair of vertices v_b, w_b with same j-invariant b, which are connected by an ℓ-isogeny. The same proofs holds for W.

Finally, after adding new edges that are only defined over $\overline{\mathbb{F}}_p$, some components can get attached. But we proved in Corollary 3.3.13 that an attachment by a new edge from j to j' means a double edge $[j, j'] \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. Hence j and j' are both roots of $\mathrm{Res}_\ell(X)$ by Lemma 3.2.1, which has degree bounded by $2\ell \cdot (2\ell - 1)$. □

Remark 3.3.17 (The 'other side' of 1728)**.** We showed that the only component for which attachment by a vertex can happen are the two components containing 1728, moreover these components fold. This means that the neighbors of v_{1728} have the same j-invariant, their neighbors as well and if we continue walking along both sides, we will ultimately meet 'opposite of' v_{1728} at a pair of conjugate j-invariants. See Figure 3.3a. Conversely, any such pair of conjugate j-invariants needs to be 'opposite' of a vertex with j-invariant 1728.

Since posting the first version of our article which did not fully explain these ideas, Castryck, Panny and Vercauteren posted a preprint [CPV] that arrives at the same conclusion using more highbrow tools. Their approach identifies the 'opposite' j-invariant as a certain CM j-invariant.

As noted Corollary 3.3.13, attachments imply double edges. To find vertices with attachment, we can use Lemma 3.2.1. Conversely, if we want a prime p with no attachments, we can find a congruence condition on p such that none of the roots if $\mathrm{Res}_\ell(X)$ are supersingular j-invariants.

3.3.4 The spine for $\ell = 3$.

In this section, we describe the spine $\mathcal{S}$ for $\ell = 3$ by means of stacking, folding and attaching behavior for $\ell = 3$. The case $\ell = 3$ is interesting, because of the use of $\mathcal{G}_3(\overline{\mathbb{F}_p})$ in SIDH and SIKE (and analogously, the case $\ell = 2$ will be discussed in Section 3.3.5). Since the starting vertex for these protocols is typically taken in $\mathbb{F}_p$, understanding the spine is important for understanding where the $\mathbb{F}_p$ j-invariants are located in the graph $\overline{\mathbb{F}}_p$.

We start with factoring over $\mathbb{Z}$ the polynomial $\mathrm{Res}_3(x)$ introduced in (3.2):

$$\begin{aligned}\mathrm{Res}_3(x) = &(-1)\cdot 3^3\cdot x^2\cdot(x-8000)^2\cdot(x-1728)^2\cdot(x+32768)^2\\ &\cdot(x^2-52250000x+12167000000)^2\cdot(x^2-1264000x-681472000)^2\\ &\cdot(x^2+117964800x-134217728000)^2.\end{aligned}$$

The factors are Hilbert class polynomials for $\Delta = -3, -8, -4, -11, -32, -20, -35$. We see that there are at most 10 vertices at which a double edge can occur.

Some of the double edges come from loops: We find the self loops by factoring

$$\Phi_3(x,x) = (-1)\cdot x\cdot(x-54000)\cdot(x-8000)^2\cdot(x+32768)^2.$$

At $j = 8000$ and $j = -32768$, there are 2 loops and no attachment at the vertices.

Example 3.3.18 (Vertices with loops)**.** In this example, we determine the $\mathcal{G}_3(\mathbb{F}_p)$ neighbors of $j = 0, 8000, 54000$ and -32768. This is done by factoring $\Phi_3(j,x)$:

1. $j = 0$: Factor $\Phi_3(0,x) = x\cdot(x+12288000)^3$. The isogeny v_0, w_0 is defined over $\mathbb{F}_p$: the factor x has multiplicity 1, so it is not a double-edge. The neighbors of v_0 are w_0 and $v_{-12288000}$ and the neighbors of w_0 are $w_{-12288000}$ and v_0. Hence there cannot be attaching edges.

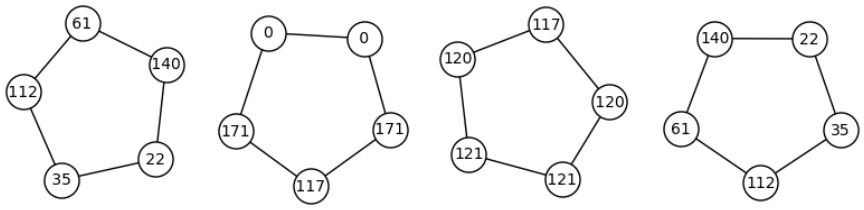


Figure 3.4: The graph $\mathcal{G}_3(\mathbb{F}_p)$ for $p = 179$. We see that the neighbors of vertices with j-invariant 0 both have j-invariant $-12288000 \bmod 179 = 171$.

2. $j = 54000$: There is one self-3-isogeny which arises from a 3-isogeny ψ between quadratic twists with $j = 54000$. Factoring $\phi_3(54000, x)$ we can check that the j-invariant 54000 does not admit a double edge.

3. $j = 8000$: we have $(x - 8000)^2 | \Phi_3(8000, x)$ and there is a double loop from $j = 8000$ in $\mathcal{G}_3(\overline{\mathbb{F}_p})$. For large p, both cannot occur over $\mathbb{F}_p$ because there are no double edges in $\mathcal{G}_3(\mathbb{F}_p)$; if one of them came from a $\mathbb{F}_{p^2}$-isogeny, we use Lemma 3.3.12 to get a third loop.

4. $j = -32768$: again, $(x + 32768)^2 | \Phi_3(-32768, x)$. Repeating the argument for $j = 8000$, the self loops cannot come from isogenies over $\mathbb{F}_p$.

Theorem 3.3.19 is a specialization of Theorem 3.3.16. As we are interested in the cryptographic applications, we restrict to the case $p \equiv 3 \pmod 4$. Then the class numbers $h(-p)$ and $h(-4p) = 3 \cdot h(-p)$ are both odd. Together with our assumption $p \equiv 2 \pmod 3$, we have $p \equiv 11 \pmod{12}$.

Theorem 3.3.19 (Stacking, folding and attaching for $\ell = 3$)**.** *Let p be prime and assume $p \equiv 11 \pmod{12}$. When passing from $\mathcal{G}_3(\mathbb{F}_p)$ to the spine $\mathcal{S} \subset \mathcal{G}_3(\overline{\mathbb{F}_p})$,*

1. *all components that do not contain 0 or 54000 stack,*

2. *there are two distinct connected components V and W that contain a j-invariant 1728, one of them contains both vertices with j-invariant 0 and the other one both vertices with j-invariant 54000. V and W fold and get attached at the j-invariant 1728.*

3. *At most 8 vertices admit new edges, attaching at most 4 pairs of components by a new edge.*

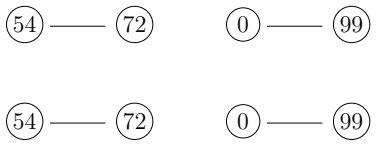
Proof. This follows from Theorem 3.3.16. In Example 3.3.18, we showed that the possible opposite vertices have j-invariants 0 and 54000. For $p \equiv 3 \pmod 4$, there are two components containing 1728: by Example 3.3.5, one of the vertices corresponding to 1728 is on the floor, the other one is on the surface, so they are on different components of $\mathcal{G}_3(\mathbb{F}_p)$. One of these vertices is on the same component of $\mathcal{G}_3(\mathbb{F}_p)$ as the vertices with j-invariant 0 and the other on a component with both vertices with j-invariant 54000. The rest are the factors of $\mathrm{Res}_3(x)$. □

3.3.5 The spine for $\ell = 2$

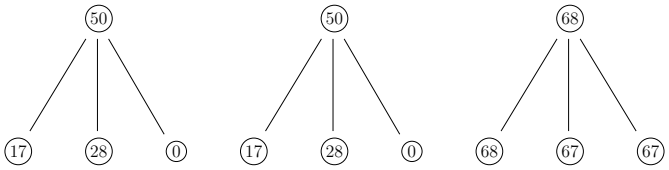
In this section, we describe the spine $\mathcal{S}$ for $\ell = 2$. The added difficulty are vertical isogenies. We describe how the components of $\mathcal{G}_2(\mathbb{F}_p)$ form $\mathcal{S} \subset \mathcal{G}_2(\overline{\mathbb{F}_p})$ in a way analogous to $\ell > 2$ (see Theorem 3.3.27). We determine whether attachment can happen for $p \equiv 1 \pmod 4$ and $p \equiv 3 \pmod 8$ (see Corollary 3.3.28) and we give experimental data on $p \equiv 7 \pmod 8$. Delfs and Galbraith [DG16] described the possible shapes for $\mathcal{G}_2(\mathbb{F}_p)$, which we record in Table 3.1.

p mod 8	$\mathrm{End}_{\mathbb{F}_p}(E)$	Shape of connected components $\mathcal{G}_2(\mathbb{F}_p)$
$p \equiv 1 \pmod 4$	$\mathbb{Z}[\sqrt{-p}]$	*horizontal* edges between two vertices (Figure 3.5a).
$p \equiv 3 \pmod 8$	$\mathbb{Z}[\sqrt{-p}], \mathbb{Z}\left[\frac{1+\sqrt{-p}}{2}\right]$	*claws*: one vertex on the surface connected by vertical isogenies to three vertices on the floor (Figure 3.5b.)
$p \equiv 7 \pmod 8$	$\mathbb{Z}[\sqrt{-p}], \mathbb{Z}\left[\frac{1+\sqrt{-p}}{2}\right]$	2-level *volcanoes*: surface is a cycle, any vertex on the surface admits one vertical isogeny to the floor. No horizontal isogenies on the floor (Figure 3.1a.)

Table 3.1: Shape of $\mathcal{G}_\ell(\mathbb{F}_p)$ for $\ell = 2$.



(a) Horizontal edges in $\mathcal{G}_2(\mathbb{F}_p)$ for $p = 113$.



(b) Claws in $\mathcal{G}_2(\mathbb{F}_p)$ for $p = 83$.

Figure 3.5: Possible shapes for $\mathcal{G}_2(\mathbb{F}_p)$ for $p \equiv 1 \pmod 4$ (left) and $p \equiv 3 \pmod 8$.

We form the graph $\mathcal{S}$ from $\mathcal{G}_2(\mathbb{F}_p)$ in two steps: first identify vertices with the same j-invariant and their edges, then add new edges. We will show that:

1. Two vertices with the same j-invariant also have the same j-invariants as neighbors (Proposition 3.3.20). This will imply that only stacking and folding is possible in Theorem 3.3.27.
2. At most one component folds, and for $p \equiv 3 \pmod 4$ this is the component containing $j = 1728$ (Proposition 3.3.24).
3. Attaching of components by a new edge happens between at most one pair of vertices, and those vertices are roots of the Hilbert class polynomial of $\mathbb{Q}(\sqrt{-15})$ (Proposition 3.3.23).

We begin with some results on the neighbors of vertices with the same j-invariant. From Corollary 3.3.6, we know that quadratic twists lie on the same level in the volcano. More is true:

Proposition 3.3.20. *Let j be a supersingular j-invariant and let v_j and w_j be the two corresponding vertices in $\mathcal{G}_2(\mathbb{F}_p)$. If $j \neq 1728$, then two vertices with the same j-invariants have the same neighbors, that is:*

1. *If* $p \equiv 1 \pmod 4$ *and the neighbor of* v_j *is* $v_{j'}$*, then the neighbor of* w_j *is* $w_{j'}$.
2. *If* $p \equiv 3 \pmod 4$ *and if*
   (a) *the vertices* v_j *and* w_j *are both on the floor, then* v_j *and* w_j *are each connected to a vertex with* j*-invariant* j'*,*
   (b) *the vertices* v_j *and* w_j *are both on the surface. Then* v_j *has 3 neighbors with distinct* j*-invariants* a, b, c *and* w_j *has three neighbors with the same distinct* j*-invariants* a, b, c*.*

Proof. 1. For $p \equiv 1 \pmod 4$, any connected component of $\mathcal{G}_2(\mathbb{F}_p)$ is an edge. If v_j and w_j lie on the same component, the result follows. Otherwise, the result can be obtained by contradiction and an application of Lemma 3.3.12.

2. If $p \equiv 3 \pmod 4$, Corollary 3.3.7 gives v_j, w_j are either both on the surface or both on the floor of their respective components.

   If they are both on the surfaces of their respective components, then they each have three neighbors: one on the floor and two on the surface. The j-invariants of the neighbors of v_j match the j-invariants of the neighbors of w_j. Any collision of these j-invariants contradicts the fact that there are no cycles of length 2: for $p \equiv 3 \pmod 4$, the class number $h(-p)$ is odd.

   If they are both on the floors of their respective components, they each has one neighbor on the surface. The j-invariants of these neighbors must be the same, otherwise we arrive at a contradiction via Lemma 3.3.12. □

Corollary 3.3.21 (Isogenies for twists)**.** *Let* $\phi : E \to E'$ *be an* $\mathbb{F}_p$*-isogeny of degree* 2 *and assume* $j(E), j(E') \neq 1728$*. Then, there is an* 2*-isogeny over* $\mathbb{F}_p$ *between the quadratic twists* $\tilde{E} \to \tilde{E}'$*.*

Corollary 3.3.22 (Attachment along a j-invariant for $\ell = 2$)**.** *Attachment along a* j*-invariant cannot happen for* $\ell = 2$*.*

We already know that attaching two components by a new edge would imply a double edge (Corollary 3.3.13).

Proposition 3.3.23 (Possible attachment by a double edge)**.** *Attachment by an edge can only happen between vertices whose* j*-invariants are* $\mathbb{F}_p$*-roots of*

$$f(X) = X^2 + 191025X - 121287375$$

provided these are supersingular $\mathbb{F}_p$ j*-invariants not equal to* $-3375, 1728$ *or* 0*.*

Proof. By Lemma 3.2.1 and Corollary 3.3.13, any such attaching j-invariants need to be roots of $\mathrm{Res}_2(X)$. Examining the neighbors of $j = 0, 1728$ and -3375, we see that they do not admit new edges coming from isogenies not defined over $\mathbb{F}_p$. □

Proposition 3.3.24 (The component of $j = 1728$ folds)**.** *Let $p \equiv 3 \pmod 4$ be prime. The connected component $V \in \mathcal{G}_2(\mathbb{F}_p)$ containing the vertices corresponding to $j = 1728$ is symmetric over a reflection passing through the vertices v_{1728} lying on the surface of V and w_{1728} lying on the floor of V. In particular, the component V folds when we pass from $\mathcal{G}_2(\mathbb{F}_p)$ to $\mathcal{S}$.*

To understand this symmetry, picture the surface of the component V as a perfect circle with equidistant vertices and all the edges to the floor are perpendicular to the surface. Then V is symmetric with respect to the line extending the edge $[v_{1728}, w_{1728}]$. See the leftmost component in Figure 3.1a. This symmetry has already mentioned in Remark 5 of [CLM+18], albeit without proof or reference.

Proof. 1. Case $p \equiv 3 \pmod 8$. V is a claw (see Figure 3.5b) and the proof of Proposition 3.3.23 shows that there is one 2-isogeny down from the surface vertex corresponding to $j = 1728$ to each vertex with j-invariant 287496 and the other vertex corresponding j-invariant 1728. The claw V is clearly symmetric and folds as described.

2. Case $p \equiv 7 \pmod 8$. In this case, $h(-p)$ is odd. We assume that $h(-p) > 1$ (otherwise we are in the claw situation discussed above).

   Then $v = v_{287496}$ and $w = v_{287496}^t$ are both on the surface, see Figure 3.6. By Proposition 3.3.20, their neighbors have j-invariants $1728, a, b$. Say the neighbors of v are v_a, v_b and the neighbors of w are w_a, w_b. Assume that v_a is on the floor. Since $a \neq 1728$, Corollary 3.3.7 tells us w_a is also on the floor. Thus, both v_b and w_b are on the surface and the symmetry is preserved. □

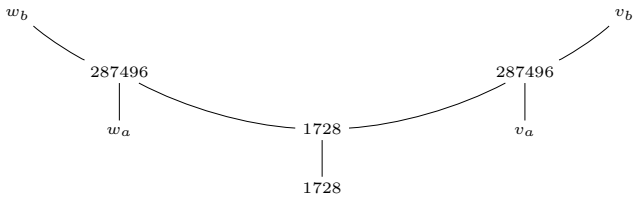


Figure 3.6: Symmetry around $j = 1728$.

Remark 3.3.25. In Proposition 3.3.24 for $p \equiv 7 \pmod 8$, the opposite vertices v_c, w_c from $j = 1728$ necessarily induce a loop in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$. Since v_c, w_c are on the surface of $\mathcal{G}_2(\mathbb{F}_p)$, we conclude that $c = 8000$ (see Section 3.2). So there always is an $\mathbb{F}_p$-rational 2-power isogeny between any two supersingular elliptic curves with j-invariants 1728 and 8000.

Corollary 3.3.26 (Folding)**.** *Suppose $V \subset \mathcal{G}_2(\mathbb{F}_p)$ is a component which folds when passing from $\mathcal{G}_2(\mathbb{F}_p)$ to $\mathcal{S} \subset \mathcal{G}_2(\overline{\mathbb{F}}_p)$.*

1. *If $p \equiv 1 \pmod 4$, then V is an edge between vertices with j-invariant* 8000.

2. *If* $p \equiv 3 \pmod 4$, *then* V *contains both the vertices with* $j = 1728$.

Proof. 1. If $p \equiv 1 \pmod 4$, then V is an edge: $[v_a, v_b]$. Folding happens if and only if $a = b$, resulting in a self-2-isogeny in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. For $p \equiv 1 \pmod 4$, the only vertices with self-2-isogenies are $j = -3375, 8000$, when these j-invariants are supersingular (see Section 3.2).

For $j = 8000$, there is a 2-isogeny from the curve with j-invariant 8000 given by $E : y^2 = x^3 - 4320x - 96768$ to its twist $\tilde{E} : y^2 = x^3 - 17280x - 774144$. The j-invariant $j = 8000$ is only supersingular for $p \equiv 5 \pmod 8$, so 2 is a nonsquare modulo p and $\tilde{E}$ is a quadratic twist (with $d = 2$).

For $j = -3375$, there are two self-loops in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$, and at least one of them not defined over $\mathbb{F}_p$. Applying Lemma 3.3.12 to this loop, w neither of these loops are defined over $\mathbb{F}_p$ and folding does not happen for the edge containing -3375.

2. If $p \equiv 3 \pmod 4$, let V be a component that folds. The surface has $h(-p)$ vertices and this class number is odd. We assume that V folds, so every vertex in it gets identified with the vertex corresponding to its twist. By Corollary 3.3.7, for $j \neq 1728$, the two vertices are either both on the surface or both on the floor. Since there are odd number of vertices on the surface, there cannot only be pairs of twists on the surface, so V must contain the two vertices with $j = 1728$, one on the floor and the other on the surface. □

We are now ready to prove the main theorem describing the spine $\mathcal{S}$ for $\ell = 2$:

Theorem 3.3.27 (Stacking, folding and attaching)**.** *For* $\ell = 2$*, only stacking, folding or at most* 1 *attachment by a new edge are possible. No attachments at a* j*-invariant are possible.*

Proof. First, let $p \equiv 1 \pmod 4$. The components of $\mathcal{G}_2(\mathbb{F}_p)$ are edges. Corollary 3.3.26 shows that the edge containing the two vertices with j-invariant 8000 folds (if 8000 is a supersingular j-invariant for p, i.e. $p \equiv 5 \pmod 8$). For the other edges, Proposition 3.3.20 says that for any edge $[v_a, v_b] \in \mathcal{G}_2(\mathbb{F}_p)$ the twists w_a, w_b also give an edge $[w_a, w_b] \in \mathcal{G}_2(\mathbb{F}_p)$. Moreover, Proposition 3.3.23 gives that there is at most 1 attachment among these edges.

For $p \equiv 3 \pmod 4$, take any component V of $\mathcal{G}_2(\mathbb{F}_p)$ and any vertex v_a on the surface of V, $a \neq 1728$. Choose a neighbor v_b of v_a. Continue along the surface in the direction of the edge $[v_a, v_b]$ and consider the sequence j-invariants of neighbors $\mathcal{V} = \{a_i\}$ until we reach a vertex with j-invariant a. Similarly, on the component W containing the edge w_a, w_b, consider the sequence of j-invariants of the neighbors $\mathcal{W} = \{b_i\}$ until we reach a vertex with j-invariant a (every surface is a cycle, so this will happen in finitely many steps). We have the following possible outcomes:

1. For some i, we find that $a_i \neq b_i$. So the curve i steps away from v_a on V has a different neighbor than its twist, which is i away from w_a. But this can only happen for $b_i = 1728$ and hence the component folds by Proposition 3.3.24.

2. The sequences are equal, but $\mathcal{V}$ stops at the twist w_a and $\mathcal{W}$ stopped at the curve v_a. Then v_a, w_a are on the same component V and the cycle on the surface has length $2 \cdot \text{length}(\mathcal{V})$. As $h(-p)$ is odd, this is not possible.

3. The sequences $\mathcal{V}$ and $\mathcal{W}$ are the same: upon replacing the labels of vertices by the j-invariants, the graphs are identical, and the two components stack.

Finally, Proposition 3.3.23 implies that at most one attachment is possible. □

To conclude this section, we study the possible attachments given by the roots of $f(X) = X^2 + 191025X - 121287375$. Because the polynomial $f(X)$ is the Hilbert class polynomial of $\mathbb{Q}(\sqrt{-15})$, its roots in $\overline{\mathbb{F}}_p$ give a supersingular j-invariant if and only if $\left(\frac{-15}{p}\right) = -1$. The discriminant of $f(X)$ is $3^6 \cdot 5^3 \cdot 7^4 \cdot 13^2$, so there is a root in $\mathbb{F}_p$ if and only if $p \equiv \pm 1 \pmod 5$. Taken together, the roots of $f(X)$ are j-invariants of a supersingular elliptic curves defined over $\mathbb{F}_p$ if and only if $p \equiv 1, 11, 24$ or $59 \pmod{60}$.

Corollary 3.3.28. *Suppose that $p \not\equiv 7 \pmod 8$ and j and j' are distinct roots of $f(X) = X^2 + 191025X - 121287375$ in $\mathbb{F}_p$. The new edge $[j, j'] \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ is an attaching edge.*

Proof. First, let $p \equiv 1 \pmod 4$. The $\mathcal{G}_2(\mathbb{F}_p)$ components are horizontal edges. Suppose that the j-invariant j admits a double edge $[j, j'] \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ that it is not an attaching edge, i.e., there is an edge $[v_j, v_{j'}]$ in $\mathcal{G}_2(\mathbb{F}_p)$. By Lemma 3.3.12, there is then a triple edge $[j, j]$. This is only possible if $j = 0$. For j or j' to be equal to 0, we would need X to be a factor of $f(X)$. Since $121287375 = 3^6 \cdot 5^3 \cdot 11^3$, as soon as $p > 11$, attachment happens whenever it can.

Next, let $p \equiv 3 \pmod 4$. The components of $\mathcal{G}_2(\mathbb{F}_p)$ are claws. If the double edge is not between two different components, then v_j and $v_{j'}$ are on the same claw (for some choice of the twists). Assume, $j \neq 1728$, they both lie on the floor.

Let v_a be the unique surface vertex of V (see Figure 3.7).

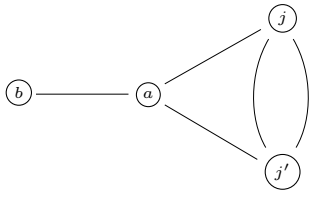


Figure 3.7: The double edge from j to j'.

This gives two distinct loops in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ of length 3, and two endomorphisms of norm 8 in $\text{End}_{\overline{\mathbb{F}}_p}(E_j)$.

To check for the existence of such an endomorphism, we check whether the roots of $f(X)$ can simultaneously be the roots of the modular polynomial $\Phi_8(X, X)$. Taking the resultant $\mathrm{Res}(f(X), \Phi_8(X, X))$ we get

$$(-1) \cdot 3^{72} \cdot 5^{36} \cdot 7^{48} \cdot 11^{34} \cdot 13^{24} \cdot 37^{10} \cdot 41^2 \cdot 43^8 \cdot 59^2 \cdot 71 \cdot 89^2 \cdot 101^2.$$

For primes $p > 101$, this will be nonzero, and there is no such a loop in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$, hence attachment happens. In the factorization of the resultant, there is one prime $p \equiv 11, 59 \pmod{60}$ and $3 \pmod 4$. For $p = 11$, we only have one connected component of $\mathcal{G}_2(\mathbb{F}_p)$, for $p = 59$, attachment happens. □

This means that attachment happens whenever it can happen for $p \not\equiv 7 \pmod 8$; moreover, the two roots are distinct if $p > 101$.

In the case $p \equiv 7 \pmod 8$, not all attachments that can happen necessarily do. We checked this for all primes $p \equiv 7 \pmod 8$ between 50000 and 100000 such that the primes above 2 do not generate the class group (otherwise $\mathcal{S}$ is already connected). There are 217 such primes, and for 41 of them the attachment happens. But there are 12 primes for which the attachment can happen but there is no attachment. For instance, for $p = 53639$, the two roots of $f(X)$ are $j = 30505$ and $j = 46665$. There are two elliptic curves with these j-invariants on the same component of $\mathcal{G}_2(\mathbb{F}_p)$ which are 48 edges apart.

The number of components in the spine for $\ell = 2$. Finally, we give the number of connected components of $\mathcal{S}$ and the number of connected components in $\mathcal{G}_2(\mathbb{F}_p)$. By Theorem 3.3.27, we know at most one component of $\mathcal{S}$ folds, and at most two components are attached by a new edge.

The number of vertices in $\mathcal{S}$ can be determined from [Cox89]

$$\#\mathcal{S} = \begin{cases} \frac{1}{2}h(-4p) & \text{if } p \equiv 1 \pmod 4 \\ h(-p) & \text{if } p \equiv 7 \pmod 8 \\ 2h(-p) & \text{if } p \equiv 3 \pmod 8. \end{cases}$$

where $h(d)$ is the class number of the imaginary quadratic field $\mathbb{Q}(\sqrt{d})$. Using standard estimates for class numbers, one typically approximates $h(d) \approx \sqrt{|d|}$.

prime mod 8	shape of $\mathcal{G}_2(\mathbb{F}_p)$	$\#\mathcal{S}$	$\approx \#(\mathcal{S}\text{-Components})$
1 mod 4	edges	$\frac{1}{2}h(-4p)$	$\frac{1}{4}h(-4p)$
3 mod 8	claw	$2h(-p)$	$\frac{1}{4} \cdot 2h(-p) = \frac{1}{2}h(-p)$
7 mod 8	volcanoes	$h(-p)$	$\frac{1}{2n} \cdot h(-p)$

Table 3.2: Size and shape of the spine, depending on primes modulo 8, the integer n denotes the order of any prime above 2 in $\mathrm{Cl}(\mathcal{O}_K)$.

3.3.6 Distances between the components of the $\mathcal{S} \subset \mathcal{G}_2(\mathbb{F}_p)$

In Sections 3.3.3 and 3.3.5, we described how the spine $\mathcal{S}$ is formed by passing from $\mathcal{G}_\ell(\mathbb{F}_p)$ to $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. In principle, one can then describe the shape and connectedness of $\mathcal{S}$ based on the knowledge of $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ and congruence conditions on p. A natural question is how the spine $\mathcal{S}$ sits inside the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. We study this question for $\ell = 2$.

For primes $p \equiv 1 \pmod 4$, the subgraph is given by single edges, with a possibility of an isolated vertex and one component of size 4 (Section 3.3.2). We expect the distances between components of the graph to behave like the distances between random vertices of the graph. To investigate if the $\mathbb{F}_p$-components are somehow distinguished, we compare the distances between $\mathbb{F}_p$-components with the distances between random vertices of the graph: In Figure 3.8, we compare these distances for 254 primes with $p \equiv 1 \pmod 4$ from 10253 to 65437. The primes were chosen to be spaced with a gap of at least 200.

In Figure 3.8, we sample the distances between 100 pairs of random points on the graph. The vertical axis represents the difference [(average distance of $\mathbb{F}_p$-components) − (average distance between 100 random pairs of points)]. For this range of primes, the average distances between $\mathbb{F}_p$-components ranged between 8.20 and 10.95. The average distances between pairs of randomly chosen vertices ranged between 7.82 and 10.85. These differences are mostly positive: The average distances between components is slightly larger than distance between two random points in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ (around 3.6% larger).

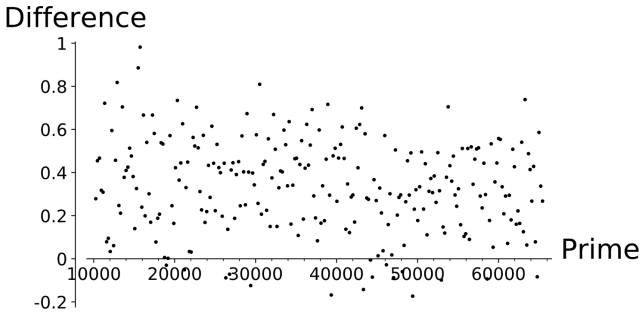


Figure 3.8: Comparing average distances between random vertices in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ and between connected components of $\mathcal{S} \subset \mathcal{G}_2(\overline{\mathbb{F}}_p)$. On the horizontal axis, we have primes $p \equiv 1 \pmod 4$. The height of each point represents (avg. distance between $\mathbb{F}_p$-components) - (avg. distance between 100 random points of $\mathcal{G}_2(\overline{\mathbb{F}}_p)$).

For $p \equiv 7 \pmod 8$, the graph $\mathcal{G}_2(\mathbb{F}_p)$ is a union of 2-level volcanoes, each with $h(-p)$ vertices on the surface and $h(-p)$ vertices on the floor. The size of the surface of any volcano is the order of the prime above 2 in the class group $\mathrm{Cl}(\mathcal{O}_K)$. If a prime above 2 generates the class group, $\mathcal{G}_2(\mathbb{F}_p)$ is connected and so is $\mathcal{S}$. The converse is not true, as it is possible for $\mathcal{S}$ to become connected via attaching.

In this case, we again compare the distances between $\mathbb{F}_p$-components and the distances between random vertices. For $p \in [50000, 100000]$, there are 217 primes for which a prime $\mathfrak{p}_2$ above 2 does not generate the class group; for 12 of these primes, the spine $\mathcal{S}$ is nonetheless connected. For 57 of those primes, there were exactly two connected components with distance less than or equal to 6.

The graphs have between 5400 and 8300 vertices and diameter between 14 and 16. For 2-isogeny graphs of this size, the average distance between two random vertices is around 9 ($\sim 0.6\times$ diameter). This number grows slowly (for primes $p \approx 500000$, the average distance between two random vertices is $\sim 0.7\times$ diameter) and we expect it to converge to the diameter. We also computed the average of the mean distances between connected components of $\mathcal{S} \subset \mathcal{G}_2(\mathbb{F}_p)$ for these primes. The mean is 4.3395, with standard deviation 1.1092 (maximum 7.000 and minimum 2.333).

3.4 Conjugate vertices, distances, and the spine

If j is a supersingular j-invariant, so is its $\mathbb{F}_{p^2}$-conjugate j^p. Because modular polynomials have integer coefficients, if j and j' satisfy $\Phi_\ell(j, j') = 0$ then also

$$\Phi_\ell(j^p, (j')^p) = (\Phi_\ell(j, j'))^p = 0.$$

This means that for any edge $[j, j'] \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$, there is a *mirror* edge $[j^p, (j')^p]$.

This leads to the idea of a mirror involution on $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$:

Definition 3.4.1. The *mirror involution* on $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ is the map defined by sending the vertex represented by $j \in \mathbb{F}_{p^2}$ to the vertex represented by j^p.

A vertex $j \in \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ is fixed under the mirror involution if and only if $j \in \mathbb{F}_p$.

Definition 3.4.2. We say that a path $(j_0, j_1, j_2, \ldots, j_{n-1}, j_n)$ in the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ is a mirror path if it is invariant under the mirror involution.

There exists at least one mirror path between any two conjugate j-invariants: it suffices to find a path from one j-invariant, say j_0, to an $\mathbb{F}_p$ j-invariant and then conjugate this path. Mirror paths either pass through a $\mathbb{F}_p$-vertex j

$$(j_0,\ j_1,\ \cdots,\ j_n,\ j,\ j_n^p,\ \cdots,\ j_1^p,\ j_0^p),$$

or through a pair of conjugate j-invariants:

$$(j_0,\ j_1,\ \cdots,\ j_n,\ j_n^p,\ \cdots,\ j_1^p,\ j_0^p).$$

The existence of mirror paths (see Definition 3.4.2) in $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ motivates us to question their length in relation to the diameter of the graph. Delfs and Galbraith [DG16] show that if one navigates to the spine $\mathcal{S}$, one can solve the path finding

problem in subexponential time using the quantum algorithm in [BJS14]. This leads naturally to the question of how the spine $\mathcal{S}$ sits in the full ℓ-isogeny graph. In Section 3.4.1 we study the distance between Galois conjugate pairs of vertices, that is, pairs of j-invariants of the form j, j^p. Our data suggest these vertices are closer to each other than a random pair of vertices in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$. In Section 3.4.2 we test how often the shortest path between two conjugate vertices goes through the spine $\mathcal{S}$ (contains a j-invariant in $\mathbb{F}_p$). We find conjugate vertices are more likely than a random pair of vertices to be connected by a shortest path through $\mathcal{S}$. Finally, we examine the question of navigating to $\mathcal{S}$ by considering the distance between arbitrary vertices and $\mathcal{S}$ in Section 3.4.3.

3.4.1 Distance between conjugate pairs

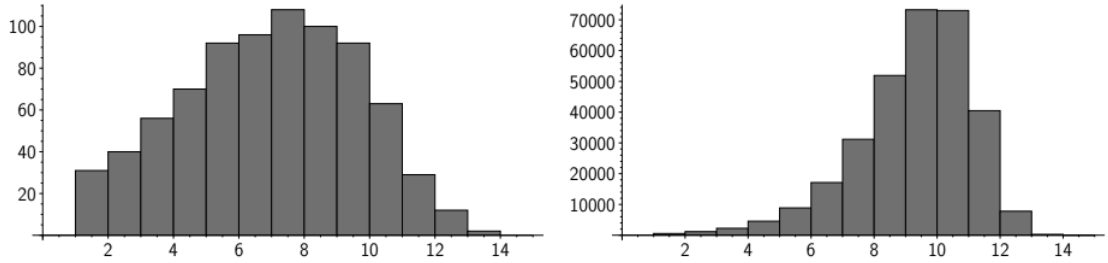
Isogeny-based cryptosystems such as cryptographic hash functions and key exchange algorithms rely on the difficulty of computing paths (*routing*) in the supersingular graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. Our experiments with $\ell = 2$ show that two random conjugate vertices are closer than two random vertices. These shorter distances may indicate that it is computationally easier to find paths between conjugate vertices, when compared with random graph vertices. We analyze the differences between these distances in this section, using the following experimental practices: Given prime p, we constructed the graph $\mathcal{G}_2(\overline{\mathbb{F}}_p)$. Then we computed the distances $\operatorname{dist}(j_1, j_2)$ between all pairs of $(\mathbb{F}_{p^2} \setminus \mathbb{F}_p)$-vertices $j_1, j_2 \in \mathcal{G}_2(\overline{\mathbb{F}}_p)$. These values were organized into two lists:

$$C_p = [\operatorname{dist}(j, j^p) \colon j \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p]$$
$$A_p = [\operatorname{dist}(j_1, j_2) \colon j_1, j_2 \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p].$$

We call the pairs from C_p *conjugate* pairs and pairs from A_p *arbitrary* pairs.

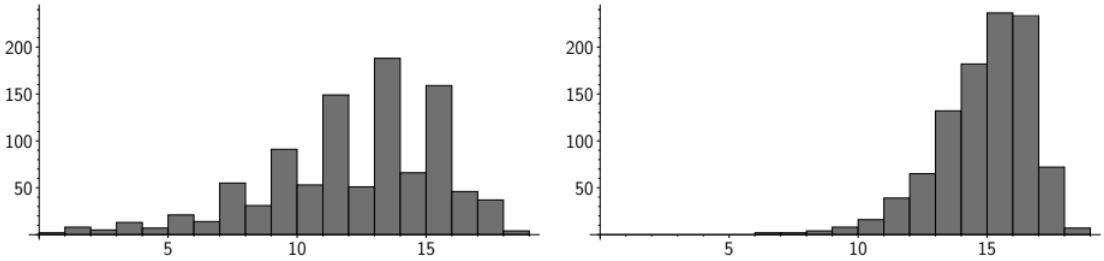
The distributions C_p and A_p for $p = 19489$ are shown in Figure 3.9. For a larger prime, it is too costly to iterate over all vertices. Instead, we took a random sample of 1000 conjugate and arbitrary pairs. The data collected for the prime $p = 1000003$ is shown in Figure 3.10. We see different distributions for the conjugate pair distances compared with the arbitrary pair distances. Conjugate pairs are more likely to be closer than arbitrary pairs. This is likely related to the fact that the neighbors of curves with conjugate j-invariants are also conjugate, as discussed in Section 3.4.

In Figure 3.10, we see a clear bias towards paths of odd length (that is, paths with an odd number of edges). This is due to the fact that conjugate j-invariants often admit a shortest path that is a mirror path (Definition 3.4.2). These paths do not usually go through the spine $\mathcal{S}$, so they have an even number of vertices and an odd number of edges. This topic is studied further in Section 3.4.2.



(a) Distances between conjugate pairs. (b) Distances between arbitrary pairs.

Figure 3.9: Histogram of distances measured between conjugate pairs and arbitrary pairs of vertices not in $\mathbb{F}_p$ for the prime $p = 19489$.



(a) Distances between conjugate pairs. (b) Distances between arbitrary pairs.

Figure 3.10: Histogram of distances between 1000 randomly sampled pairs of arbitrary and conjugate vertices for the prime $p = 1000003$.

3.4.2 How often do shortest paths go through the spine

We consider the question of how easy it is to navigate to the spine $\mathcal{S}$. Delfs and Galbraith [DG16] use L-isogenies to navigate to the spine $\mathcal{S}$, where L is a set of small primes. We study the situation when $L = \{2\}$. Any path from j to a vertex j in the spine $\mathcal{S}$ can be *mirrored* to obtain a path of equal length from j to j^p, and hence a path between j and j^p passing through the spine. This construction motivates the following definition:

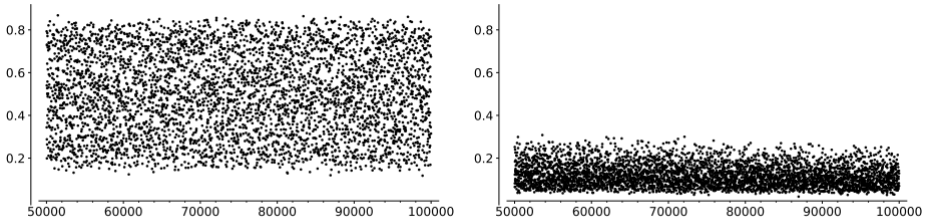
Definition 3.4.3. A pair of vertices are *opposite* if there exists a shortest path between them that passes through the $\mathbb{F}_p$ spine.

Experimental methods. We used built-in functions of Sage [The24] to perform our computations. We tested how often a shortest path between two conjugate vertices went through the spine $\mathcal{S}$. Shortest paths are not necessarily unique, so it is not enough to compute a shortest path and check whether it passes through the spine. Instead, to verify whether a pair j_1, j_2 is opposite, we run over all vertices in $j \in \mathbb{F}_p$ and check whether there is a j such that

$$\mathrm{dist}(j_1, j_2) = \mathrm{dist}(j_1, j) + \mathrm{dist}(j, j_2).$$

For smaller primes (< 5000) we computed the proportions for all pairs of vertices in $\mathbb{F}_{p^2}\backslash\mathbb{F}_p$. For larger primes, we randomly selected 1000 pairs of points j_1, j_2 in $\mathbb{F}_{p^2}\backslash\mathbb{F}_p$ and checked whether each of the pairs $(j_1, j_2), (j_1, j_1^p)$ were opposite.

Conjugate pairs vs arbitrary pairs. For each prime p ranging between 50000 and 100000, we sampled data from 1000 random conjugate pairs of vertices and 1000 pairs of arbitrary vertices. We computed the proportions of conjugate pairs which are opposite and the proportions of arbitrary pairs which are opposite. This data is displayed in Figure 3.11.



(a) Conjugate pairs (b) Arbitrary pairs

Figure 3.11: Proportions of opposite vertex pairs; x-axis represents primes. The data are for a random sample of 1000 pairs of conjugate and arbitrary pairs.

Our data suggest conjugate vertices are more likely to be opposite than arbitrary vertices and that this bias increases with p. For the primes displayed in Figure 3.11, conjugate pairs are more than four times as likely to be opposite, compared with arbitrary pairs. For $p \in [1000, 5000]$ (not displayed), conjugate pairs are approximately twice as likely to be opposite than random pairs.

The graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ can be symmetrically constructed from $\mathcal{S}$ by adding edges and mirror edges at once, leading one to suspect $\mathcal{S}$ to be central to the graph. However, we have found the shortest paths between arbitrary pairs of vertices are less likely to pass through the spine, contradicting that perspective.

Varying the residue class of p. In this section, we consider only arbitrary pairs of vertices, and we study the overall connectivity of the graph by looking at the proportions of opposite arbitrary vertices, for primes p in different congruence classes modulo 8. See Figure 3.12.

In this data, we have computed for each prime p a random sample of 1000 arbitrary pairs of vertices and checked how many are opposite. Then, we take this number and divide by $1000 \cdot |\mathcal{S}|$, to account for differences in the numbers of $\mathbb{F}_p$ vertices for these primes. We have sorted the data by primes $p \bmod 8$ and displayed it in the graphs in Figure 3.12. Our results suggest that the size and connectedness of the spine $\mathcal{S}$ affect the proportion of opposite pairs:

1. Size of $\mathcal{S}$: Shortest paths are more likely to pass through $\mathcal{S}$ if it is larger. For this reason, we divide the proportions of opposite arbitrary pairs of vertices

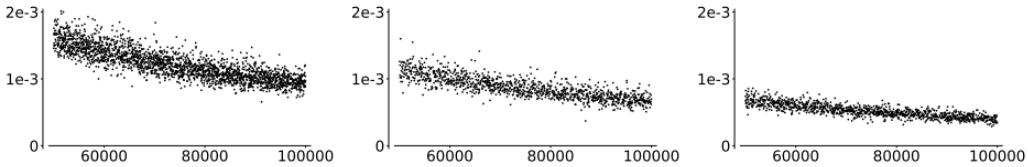


Figure 3.12: The horizontal axes of these figures are primes. The heights are proportions of 1000 pairs of arbitrary opposite vertices, normalized by $|\mathcal{S}|$.

by $|\mathcal{S}|$ in Figure 3.12. The normalized proportions still differ modulo 8, indicating another factor is at play, possibly the connectedness of $\mathcal{S}$.

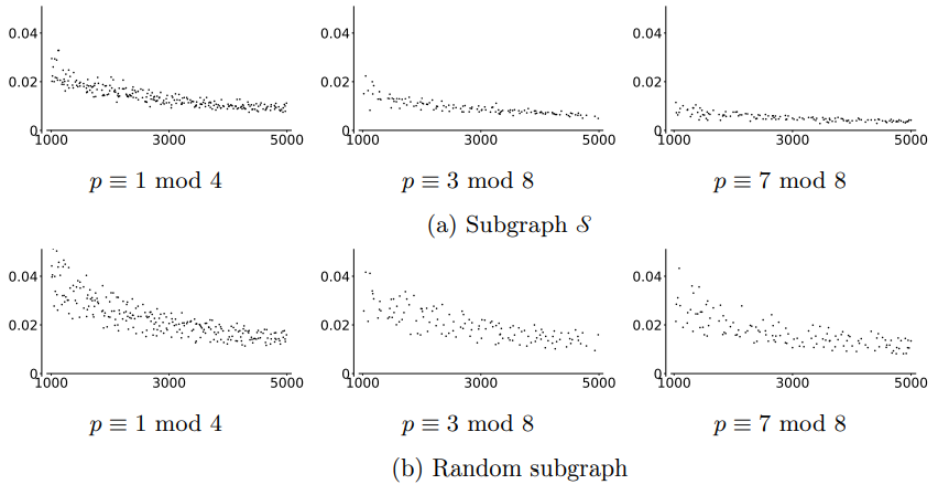
2. Connectedness: When $\mathcal{S}$ is less connected, pairs are more likely to have shortest paths through it. The spine is the least connected when $p \equiv 1 \pmod 4$, and most connected when $p \equiv 7 \pmod 8$ (Table 3.2). This could explain the difference in proportions when normalized by the size of $\mathcal{S}$.

   For example, for $p_1 = 19991$ ($p_1 \equiv 7 \pmod 8$, $\mathcal{S}$ is connected, $|\mathcal{S}| = 199$) and $p_2 = 19993$ ($p \equiv 1 \pmod 4$, $\mathcal{S}$ is maximally disconnected, $|\mathcal{S}| = 30$), we would expect $199/30 > 6$ times more opposite pairs in the p_1 case. But out of 1000 arbitrary pairs, only 266 pairs were opposite for p_1 compared to 112 pairs for p_2.

To further study whether differences occurring in the normalized proportions of vertices which are opposite for $p \bmod 8$ were due to the connectedness of $\mathcal{S}$, we took a random subgraph of the same size as $\mathcal{S}$ and obtained the proportion of pairs of arbitrary vertices with a shortest path passing through the random subgraph. We took the average over 10 random subgraphs for each prime between 1000 and 5000. The data, in Figure 3.13, show less distinction $\bmod 8$ compared to the data in Figure 3.12. This suggests that the connectedness of $\mathcal{S}$ is a dominant factor affecting the normalized proportion of opposite pairs of vertices.

3.4.3 Distance to spine

Distance to the spine versus a random subgraph. For two fixed primes we study how the spine fits into the graph, and compare this with a randomly selected subgraph. Specifically, we compare a random vertex's distance to the spine, with a random vertex's distance to a random subgraph of the same size as the spine. We observe that if the spine is connected, then the distance to the spine seems greater than the distance to a random subgraph. This agrees with the intuition that a small connected subgraph ($|\mathcal{S}| \approx O(\sqrt{p})$) will be further from most vertices than a random subgraph, which will have many connected components uniformly distributed throughout the graph.



(a) Subgraph $\mathcal{S}$

(b) Random subgraph

Figure 3.13: Normalized proportion of pairs with a shortest path through the specified subgraph, as p varies.

We studied the distances for $p = 19991$ and $p = 19993$. For $p = 19991$, the subgraph $\mathcal{S}$ is connected, and for $p = 19993$, $\mathcal{S}$ is maximally disconnected (see [DG16]). For each value of p, we constructed the graph $\mathcal{G}_2(\overline{\mathbb{F}}_p)$, the spine $S_0 := \mathcal{S}$, and chose several random subgraphs $S_1, \ldots, S_n$. We define the distance between a vertex j and a subgraph S_i to be

$$\operatorname{dist}(j, S_i) = \min\{\operatorname{dist}(j, j') : j' \in S_i\}.$$

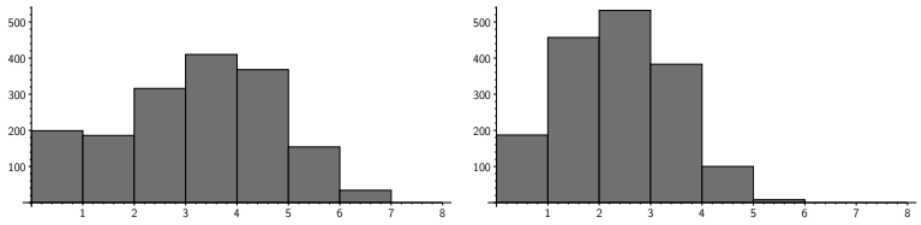
We computed lists $d_i = [\operatorname{dist}(j, S_i) \colon j \in \mathcal{G}_2(\overline{\mathbb{F}}_p)]$ to measure how dispersed S_i is in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$. Histograms of the distributions of the d_i are given in Figure 3.14.

The significant difference between the two primes shown in Figure 3.14 can also be explained by the number of vertices in $\mathcal{S}$. Since $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ is a 3-regular graph, for a random vertex j, there are at most $3 \cdot 2^{d-1}$ vertices of distance d away from j (and this limit is achieved if there are no collisions on the paths leaving j). If $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ has N vertices and H is a random subgraph with M vertices, then the expected distance to H from a random vertex should be $\approx \log_2(N/M)$. For $p = 19991$, $|\mathcal{S}| = 199$, so we expect the average distance to $\mathcal{S}$ to be 3.06. For $p = 19993$, $|\mathcal{S}| = 30$, so we expect the average distance to $\mathcal{S}$ to be 5.80.

When maximally disconnected, the distances to the spine were similar to that of a random subgraph. However, when the spine is connected, the distances are slightly longer. This shows that modeling the spine as a random subgraph may lead to an underestimate of the distances.

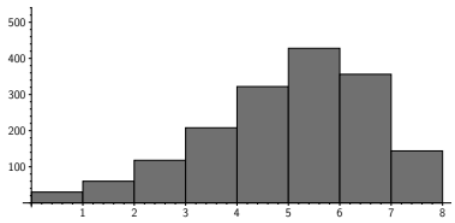
Comparison across primes p. In order to compare the distances to $\mathcal{S}$ across different primes and account for the expected average distance based on the size of $\mathcal{S}$ we consider normalized distances as follows:

$$d_p = (\text{average distance to } \mathcal{S} \text{ for prime } p)/\log_2(|\mathcal{G}_2(\overline{\mathbb{F}}_p)|/|\mathcal{S}|)$$

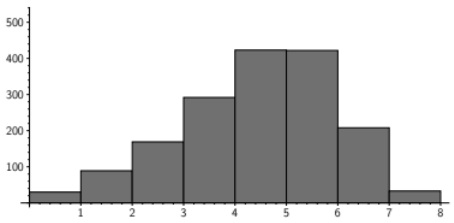


(a) Distances to $\mathcal{S}$ for $p = 19991$.

(b) Distances to R for $p = 19991$.



(c) Distances to $\mathcal{S}$ for $p = 19993$.



(d) Distances to R for $p = 19993$.

Figure 3.14: Distances to the spine $\mathcal{S}$ contrasted with distances to a random subgraph R of the same size. The spine $\mathcal{S}$ is connected for $p = 19991$ and a union of disconnected edges for $p = 19993$.

Recall that $\log_2(|\mathcal{G}_2(\overline{\mathbb{F}}_p)|/|\mathcal{S}|)$ is the expected distance to the spine from a random vertex. We observed that the average distances were lower than the expected distance based on the connectedness of $\mathcal{S}$. There are also clear differences in the distributions of d_p when considering residue classes of p modulo 8. This is shown in Figure 3.15. In particular, the data mod 8 match our findings on the proportion of opposite pairs, see Figure 3.12.

The different behavior of d_p for the different congruence classes mod 8 can be explained by the size of the spine. If $|\mathcal{S}|$ is large, we will need fewer steps to reach the spine from a random vertex v. Hence, when counting the paths of length d from v, we will encounter less backtracking and the estimate is more precise. Looking at Figure 3.16, we see that for $p \equiv 7 \pmod 8$, the size of the spine is the largest, and for $p \equiv 1 \pmod 4$, the size of the spine is the smallest.

We tested this in a fixed congruence class: for primes p with $p \equiv 7 \pmod 8$ and $p \in [15\,000, 20\,000]$, the mean distance to the spine is 4.040 with standard deviation $\sigma = 0.413$ if $|\mathcal{S}| < 100$ and mean 3.007 with $\sigma = 0.335$ if $|\mathcal{S}| > 100$.

3.5 When are conjugates ℓ-isogenous?

In Section 3.4.2 we studied paths between conjugate j-invariants in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ that go through the spine $\mathcal{S}$. If j and j^p happen to be 2-isogenous, then that is the shortest path between them and this path does not pass through $\mathcal{S}$. This leads us to the natural question:

Question 3.5.1. How often are conjugate j-invariants in $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$ ℓ-isogenous?

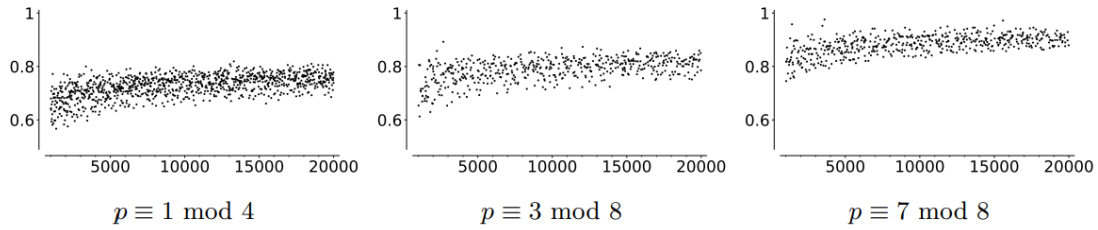


Figure 3.15: Normalized average distances d_p to the spine $\mathcal{S}$, as prime p varies on the horizontal axes.

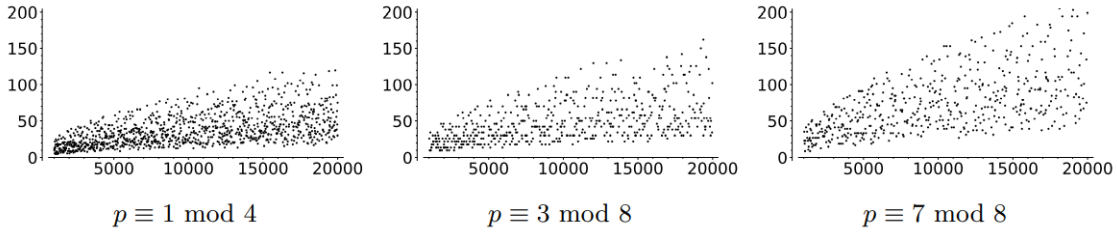


Figure 3.16: Size of the spine as prime p varies on the horizontal axes.

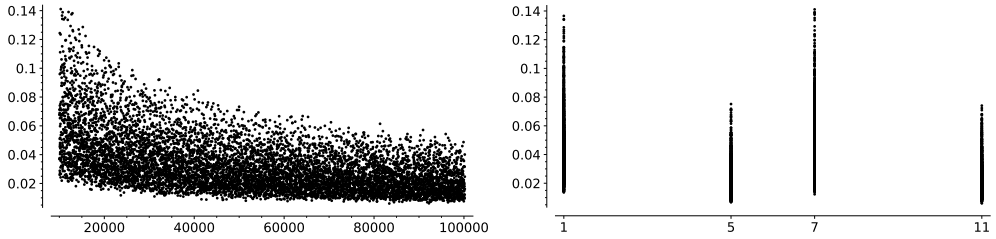
3.5.1 Experimental data: 2-isogenies

We collected data on supersingular j-invariants for all primes $5 \leq p \leq 100193$ (a total of 9605 primes). For each p, we determined all of the $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$ j-invariants and counted those that are also 2-isogenous to their conjugates.

With a few exceptions, all of the proportions computed are positive and strictly less than 1. The small primes (roughly $p < 5000$) have a wide range of proportions, between 0 and 1. This is expected due to the small number of points on their $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ graphs. For example: there are some primes p such that all of the pairs of conjugates are 2-isogenous. On the other hand, if $\mathcal{G}_\ell(\overline{\mathbb{F}}_p) \setminus \mathcal{G}_\ell(\mathbb{F}_p) = \emptyset$ this proportion is trivially zero. This happens only for 15 small primes, namely those dividing the order of the Monster group [Ogg75]. Notably, the only examples of p for which the proportion is zero, but not trivially zero, are $p = 101, 131$.

To avoid small prime phenomena, we focused on analyzing the data we collected for $10007 \leq p \leq 100193$, a total of 8378 primes. The graph of proportions for these data can be found in Figure 3.17. We found that the mean proportion in the data is 0.032780 with standard deviation of 0.019134.

We then sorted the data by congruence conditions to look for patterns. The biggest difference appeared when we re-sorted the data according to the congruence class of the primes modulo 12. The primes in this range are fairly evenly distributed into the congruence classes modulo 12, as one can see in Table 3.3.



(a) As p varies (b) As p varies mod 12

Figure 3.17: Proportion of 2-isogenous conjugate pairs in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ for primes in range $10007 \leq p \leq 100193$.

p mod 12	1 mod 12	5 mod 12	7 mod 12	11 mod 12
Total # of primes:	2079	2104	2101	2094
Mean:	0.043551	0.021969	0.043375	0.022244
Standard Deviation:	0.019815	0.010206	0.020140	0.010512

Table 3.3: Proportions of 2-isogenous conjugates, $10007 \leq p \leq 100193$

Primes modulo 12. In Table 3.3, we summarize the differences between the congruence classes modulo 12. Note the similar, higher means for $p \equiv 1, 7 \pmod{12}$ and the similar, lower means for $p \equiv 5, 11 \pmod{12}$. These distributions are skewed according to the congruence class, as we can also see from the graph in Figure 3.17b. There appears to be a correlation between primes $p \equiv 1, 7 \pmod{12}$ and between primes $p \equiv 5, 11 \pmod{12}$. A two-sample t-test confirms correlation at the 99.8% level.

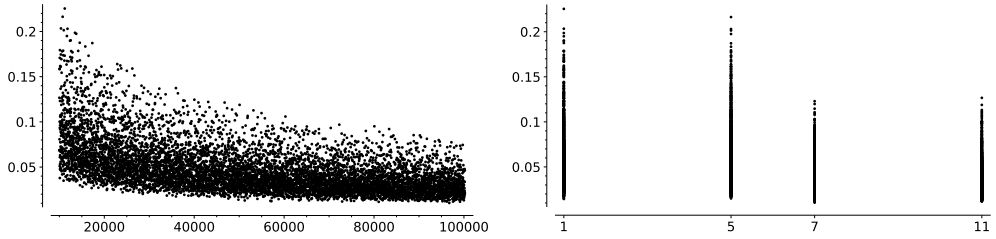
3.5.2 Experimental data: 3-isogenies

We collected data on the supersingular for all the primes $5 \leq p \leq 100193$ (a total of 9,605 primes) and computed the proportion of conjugate pairs that are also 3-isogenous. We present these data in the same format as the 2-isogeny data presented in Section 3.5.1.

Once again, we observe some small prime phenomena (proportions of 1 and 0 for p small). However, in the 3-isogeny case we do not have nontrivial examples of primes p for which the proportion of 3-isogenous conjugates is 0. On the contrary, if there exist conjugate j-invariants in $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$, then there is at least one pair of 3-isogenous conjugates. (Recall that the two counterexamples to this statement in the 2-isogeny case were $p = 101, 131$.)

To avoid small prime phenomena, we again focused on analyzing the data we collected for $10007 \leq p \leq 100193$, a total of 8378 primes. The graph of proportions for these data can be found in Figure 3.18. In this collection of data,

for primes $10007 \leq p \leq 100193$, we found there to be a mean proportion of 3-isogenous conjugate pairs of 0.047306, with standard deviation of 0.026568.



(a) As p varies (b) As p varies mod 12

Figure 3.18: Proportion of 3-isogenous conjugate pairs in $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ for primes in range $10007 \leq p \leq 100193$.

As in the 2-isogeny case, we again sorted the data by congruence conditions to look for patterns. The biggest difference appeared when we re-sorted the data according to the congruence class of the primes modulo 12.

Primes modulo 12. In Table 3.4, we report the data for different congruence classes modulo 12. Note the similar and higher means for $p \equiv 1, 5 \pmod{12}$ and the similar and lower means for $p \equiv 7, 11 \pmod{12}$, as we can also observe in Figure 3.18b. There appears to be a correlation between $p \equiv 1, 5 \pmod{12}$ and between primes $p \equiv 7, 11 \pmod{12}$. A two-sample t-test confirms these correlations at the 99.8% level.

p mod 12	1 mod 12	5 mod 12	7 mod 12	11 mod 12
# of primes:	2079	2104	2101	2094
Mean:	0.058526	0.059034	0.035620	0.036107
standard deviation:	0.029488	0.029729	0.016369	0.016706

Table 3.4: Proportions of 3-isogenous conjugates for $10007 \leq p \leq 100193$.

3.5.3 Further questions

Our experimental data suggest that, at least for $\ell = 2, 3$ and with the exception of a few small primes, the proportion of conjugate pairs that are ℓ-isogenous is a small positive number. In particular, all of the primes $p \neq 101, 131$ with supersingular j-invariants in $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$ observed have at least one such pair. This motivates the following two questions:

Question 3.5.2. For $p > 131$, is there always at least one pair of ℓ-isogenous conjugate j-invariants on $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$?

Question 3.5.3. For large p, is there a nontrivial lower and/or upper bound for the proportion of ℓ-isogenous conjugate j-invariants on $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$?

A partial answer to Question 3.5.3 is in [CLG09, Lem. 6]: for $p \equiv 1 \pmod{12}$, the set of j-invariants that are ℓ-isogenoues to their conjugates is of size $\sqrt{\ell}\tilde{O}(\sqrt{p})$, which yields an upper bound. We discuss later results (such as [EHL+b] and [CS22]) in Section 3.8.

We note that this bound does not explain the significant difference in the average of the proportion of ℓ-isogenous conjugate pairs when we vary the congruence class of p modulo 12. This prompts us to formulate the question:

Question 3.5.4. How does the proportion of ℓ-isogenous conjugate j-invariants on $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ relate to the conjugacy class of p mod 12?

Remark 3.5.5. We can also consider ℓ-isogenous conjugate pairs of j-invariants in the context of the modular curve $X_0(\ell)$. In [Ogg75], Ogg defined $X_0(\ell)^+$, the Atkin-Lehner quotient of $X_0(\ell)$ by the Fricke involution. If two conjugate $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$ j-invariants j, j^p are ℓ-isogenous, then (j, j^p) is a point of $X_0(\ell)$. The image of (j, j^p) under the quotient map is an $\mathbb{F}_p$-rational point of $X_0(\ell)^+$. Conversely, any $\mathbb{F}_p$-rational point of $X_0(\ell)^+$ whose preimage in $X_0(\ell)$ contains a non-$\mathbb{F}_p$ point corresponds to an ℓ-isogenous conjugate pair j, j^2. Intersecting with the supersingular locus, we have an alternative description of the set of $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$ ℓ-isogenous conjugate pairs in the Supersingular Isogeny Graph.

3.6 Diameter

Sardari [Sar18] estimated the diameters of k-regular LPS [LPS88] Ramanujan graphs and random Cayley graphs with n vertices to be asymptotically $\frac{4}{3}\cdot\log_{k-1}(n)$, resp. $\log_{k-1}(n)$. We present data on the diameters of the supersingular 2-isogeny graphs, which are 3-regular on approximately $p/12$ vertices.

For $p \equiv 3 \pmod{4}$, the supersingular j-invariant 1728 is 2-isogenous to only one other curve. This isolated position in the graph may lead to longer diameters.

We can see a lower bound $\log_2\left(\left\lfloor \frac{p}{12} \right\rfloor\right) - \log_2(3) + 1$ on the diameter as follows. Starting from a random vertex and taking a walk of length n, the walk reaches at most $3 \cdot 2^{n-1}$ vertices as endpoints (exactly that number if there are no collisions). Since there are $\left\lfloor \frac{p}{12} \right\rfloor + \epsilon$ vertices in the graph, with $\epsilon = 0, 1, 2$, the diameter cannot be less that the smallest n_0 such that $3 \cdot 2^{n_0-1} + 1 \geq \left\lfloor \frac{p}{12} \right\rfloor + \epsilon$.

We collected graph data for the diameters of the supersingular 2-isogeny graphs $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ for 4600 primes p. We used the built-in Sage "diameter" function [The24] on the graphs. The implementation can be found in the walk.sage worksheet available on our github repository. We collected diameters of $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ in batches for ranges of primes. The smallest prime we have data for is $p = 1009$ and the largest is $p = 9501511$. This data is displayed in Figure 3.19.

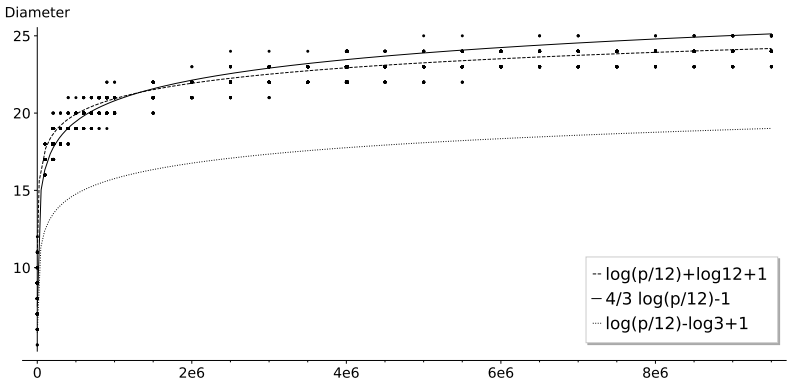
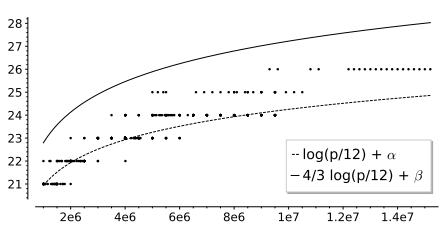


Figure 3.19: Diameter of $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ for 4600 primes p with $1009 \leq p \leq 9501511$ with $y = \log_2(p/12) + 1 + \log_2(12)$ (dashed), $y = 4/3 \log_2(p/12) + 1$ (solid) and the proven lower bound $y = \log_2(p/12) - \log_2(3) + 1$ (dotted).

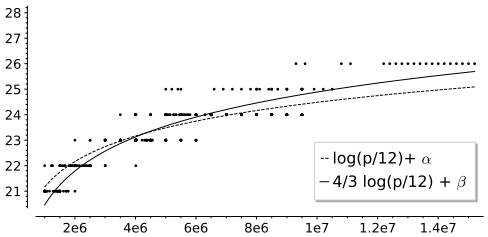
The shifted $(4/3)\log_2(p/12)$ graph (a solid line) increases too steeply for larger primes. This could suggest that the 2-isogeny graph diameters may be more like random Cayley graphs, i.e. with distribution closer to $\log_2(p/12)$. Diameters for larger primes would be needed to make a conclusion, but our algorithm would not be able to find those in a reasonable amount of time.

For this reason, we study *lower bounds* on the diameter: We chose $j = 0$ as the starting vertex and in every step, we added the neighbors of all the known vertices until we reach all the vertices. This algorithm gives a lower bound for the diameter of $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$, and we were able to reach larger primes p, shown in Figure 3.20. The dashed curve $y = \log_2(p/12) + \log_2(12) + 1$ no longer fits the data (Figure 3.20b). We chose primes $p \equiv 23 \pmod{24}$: we chose $p \equiv 7 \pmod 8$ so that the spine $\mathcal{S}$ has the same structure and approximate size (see Table 3.2).

We tried fitting the curves $\log_2(p/12) + \alpha$ and $4/3 \log_2(12) + \beta$ with the best constants α, β and as a first approximation we tried minimizing the sum of squares of distances to these curves, yielding $\alpha \approx 4.8168$ and $\beta \approx -1.3356$.



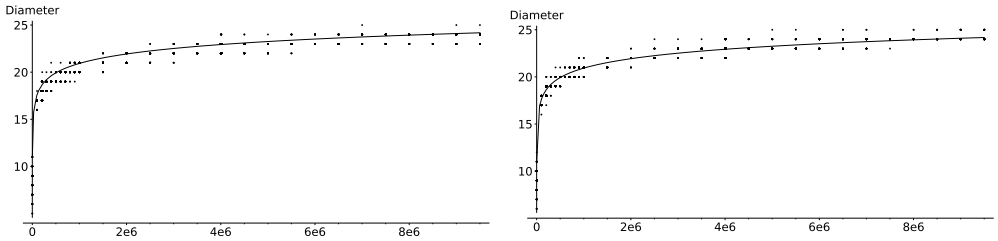
(a) $\alpha = \log_2(12) + 1$, $\beta = 1$.



(b) $\alpha \approx 4.8168$ and $\beta \approx -1.3356$

Figure 3.20: Lower bounds on the diameter δ of $\mathcal{G}_2(\overline{\mathbb{F}}_p)$ for $p \equiv 23 \pmod{24}$ with $1020431 < p < 15201671$ (a total of 431 primes), along with the graphs of $y = \log_2(p/12) + \alpha$ (dashed) and $y = 4/3 \log_2(p/12) + \beta$ (solid).

Diameter for different classes of primes modulo 12. We investigated the behavior of the diameter as p varies modulo 12. We found a slight, but noticeable, bias: Primes $p \equiv 5, 11 \pmod{12}$ tend to have a 2-isogeny graph of larger diameter compared with primes $p \equiv 1, 7 \pmod{12}$. This is visible in Figures 3.21a and 3.21b. In Figure 3.21b, the diameters tend to be slightly larger than the graph of $y = \log_2(p/12) + \log_2(12) + 1$, whereas those in Figure 3.21a tend to be more evenly distributed above and below $y = \log_2(p/12) + \log_2(12) + 1$. Table 3.5 supports the visible bias.



(a) $p \equiv 1, 7 \pmod{12}$ (b) $p \equiv 5, 11 \pmod{12}$

Figure 3.21: Diameters of 2-isogeny graph over $\overline{\mathbb{F}}_p$ for different congruence classes of the prime p, shown with $y = \log_2(p/12) + \log_2(12) + 1$

average δ for $100,000 < p < 300,000$				average δ for $300,000 < p < 500,000$			
1 mod 12	17.22	5 mod 12	17.88	1 mod 12	18.40	5 mod 12	18.92
7 mod 12	17.73	11 mod 12	17.99	7 mod 12	18.82	11 mod 12	19.10

Table 3.5: Average diameters sorted by primes modulo 12. The first data set contains around 100 primes per class, the latter between 10 to 17 primes.

3.7 Conclusions

We determined how the connected components of $\mathcal{G}_\ell(\mathbb{F}_p)$ merge together to give the spine $\mathcal{S} \subset \mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. For any specific ℓ and any p, one can determine the resulting shape explicitly if one knows the structure of the class group $\mathrm{Cl}(\mathcal{O}_K)$.

For $\ell = 2$, we summarize the biases we observed in our heuristic data, for the different congruence classes of $p \bmod 12$. The data suggest the following, although more careful analysis is needed to confirm:

- $p \equiv 1 \bmod 12$:
  - smaller 2-isogeny graph diameters,

  - larger number of spinal components,
  - larger proportion of 2-isogenous conjugate pairs,

- $p \equiv 11 \bmod 12$:
  - larger 2-isogeny graph diameters,
  - smaller number of spinal components,
  - smaller proportion of 2-isogenous conjugate pairs.

3.8 Related work

Adj, Ahmadi and Menezes [AAM19] were the first to study the fine structure of the graph $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. They showed that the graph $\mathcal{G}_\ell(\mathbb{F}_{p^2}, -2p)$ of supersingular elliptic curves and isogenies over $\mathbb{F}_{p^2}$ with trace $t = -2p$ is isomorphic as a graph to $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$. Our approach in the current paper also studies a subclass of curves and the paths passing through these curves. After our preprint [ACL+19], a related approach was introduced by Boneh and Love [LB20] to study the subgraph of supersingular curves with small non-integer endomorphisms. Small endomorphisms can be used to efficiently compute isogenies between the special curves.

Shortly after our work was first posted [ACL+19], Castryk, Panny, and Vercauteren posted a paper on rational isogenies [CPV]. Their work gives an algorithm for computing an ideal to connect two supersingular elliptic curves over $\mathbb{F}_p$ which have the same $\mathbb{F}_p$-endomorphism ring. Their focus on $\mathbb{F}_p$-curves is parallel to our results regarding the structure of the $\mathbb{F}_p$ isogeny graphs $\mathcal{G}_\ell(\mathbb{F}_p)$ and $\mathcal{S}$.

More recently, Eisenträger et al. [EHL+b] gave a cycle finding algorithm to compute endomorphism rings of supersingular elliptic curves. A key part of their algorithm involves finding j-invariants that are ℓ-isogenous to their conjugates. [EHL+b] provided an answer regarding the lower bound in Question 3.5.3, and thus also for Question 3.5.2. They proved that the cardinality of the set of ℓ-isogenoues conjugate j-invariants is larger than $C\frac{\sqrt{p}}{\log\log p}$, for some constant C depending on ℓ (assuming GRH). Both proofs assume that $\ell < p/4$. Chenu and Smith [CS22] show that the ℓ-isogenoeus conjugate j-invariants are oriented by the order $\mathbb{Z}[\sqrt{-\ell p}]$, in the sense of Remark 2.5.8.

The relationship between LPS graph and $\mathcal{G}_\ell(\overline{\mathbb{F}}_p)$ was also studied in [CFL+].

Florit and Finol [FF20] constructed a dababase of supersingular isogeny graphs for primes up to $30,000$ and for small $\ell \in \{2, 3, 5, 7, 11\}$, complementing some of the questions we investigated in this paper.

Chapter 4

Background on CSIDH

Starting from this chapter, we leave behind the theory of supersingular isogeny graphs over $\mathbb{F}_{p^2}$, and study the theory of supersingular elliptic curves over $\mathbb{F}_p$. In particular, we focus on the CSIDH scheme [CLM+18].

CSIDH is the main contender of the isogeny group-action schemes. It is based on the CRS framework [Cou06, RS06] and the modern instantiation thereof by [DKS18]. CSIDH was the first procotol that made group-action isogeny-based protocols potentially interesting for practical use.

We first define the general framework of constructing cryptographic schemes from group actions in Section 4.1. Then we specialize to the specific situation of CSIDH in Section 4.2. We explain the relevant parameter choices, and lastly discuss the implementation of the CSIDH scheme in Section 4.3.

This chapter is based on the background sections of [BBC+21, BKL+23], which are part of this thesis in Chapters 6 and 7. The background sections from these two papers have significant overlap: both are concerned with computing the CSIDH group action efficiently. The shared parts are consolidated into Sections 4.2 and 4.3 and are provided with further expository details.

4.1 Group action in cryptography

In this section, we start from the Diffie–Hellman protocol (Section 4.1.1) and build up to the group-action based Diffie–Hellman protocols (Section 4.1.2), defining and examining the main hard problems underlying its security. We also explain how to obtain an instantiation of this framework from the class group action on elliptic curves with CM (Section 4.1.3). This is only a high-level overview, the practical choices are discussed in depth in Section 4.2.

The goal of this section is to give an intuitive understanding of the cryptographic assumptions relevant for later chapters. Proper definitions can be found in any provable-security based cryptography textbook, such as [Gol00] or [KL07].

4.1.1 Diffie–Hellman key exchange

We start with an overview of the Diffie–Hellman key exchange [DH76], which is "...both literally and figuratively, at the foundation of public-key cryptography" [Smi18].

The Diffie–Hellman key exchange is the following protocol for two parties – Anouk and Bas – who try to agree on a shared secret while communicating on an open channel.

Anouk and Bas agree on a prime p and a generator g of the multiplicative group of $(\mathbb{Z}/p\mathbb{Z})^\times$. Anouk then chooses their own secret $a \in \mathbb{Z}/(p-1)\mathbb{Z}$, and computes the power g^a, which is their public key pk_A. The value sk_A is kept by Anouk in private. Bas chooses their own secret $b \in \mathbb{Z}/(p-1)\mathbb{Z}$, and computes the power g^b, so Bas has a secret-public key pair $(\mathsf{sk}_B, \mathsf{pk}_B) = (b, g^b)$. Remember that all the exponentiations are done $\bmod p$.

Anouk and Bas exchange their public keys over the insecure channel. Anouk can compute $\mathsf{pk}_B^{\mathsf{sk}_A} = (g^b)^a \bmod p$. Bas can compute $\mathsf{pk}_A^{\mathsf{sk}_B} = (g^a)^b \bmod p$. They both arrive at the same *shared key* $g^{ab} \bmod p$.

The protocol extends naturally to any finite cyclic group G of size q with generator g, and we assume we are in this situation for the rest of the section.

Computational Diffie–Hellman problem. For the protocol to be secure in an intuitive sense, any attacker eavesdropping on the channel should not be able to obtain the shared key (this is called the one-wayness). The *transcript* of the conversation is the prime p, the choice of g, and the values $\mathsf{pk}_A = g^a$ and $\mathsf{pk}_B = g^b$. We would like to say that any eavesdropper cannot compute the shared key g^{ab}; however, this is impossible. An attacker can always use *brute force* and compute all possible values g^x and identify a, b, upon which computing g^{ab} is easy.

The brute-force attack (in the worst case) requires going through all the q possible values for a and b. If the group G is large, this will be computationally infeasible. We can therefore make it "hard" for the attacker by choosing a large G.

From now on, we assume that attackers are efficient: an *efficient algorithm* is a randomized algorithm that runs in polynomial time in the size of the input. In the context of the Diffie–Hellman key exchange, the input is *not* the group G, but the transcript, which consists of a description of the group G and some elements in G. This can be represented in $O(\log_2 q)$ bits. Therefore, an efficient attacker can only perform $O(\log^c(q))$ operations for some constant c. This rules out brute-force attacks as those require $\Omega(q)$ operations.

Note also that for any fixed a, b, computing g^{ab} from g^a and g^b is easy. The values a and b therefore need to be chosen randomly.

Definition 4.1.1. Let G be a group of order q and let g be a generator. We say that the *Computational Diffie–Hellman problem* (CDH) problem is hard in the group G if there is no efficient algorithm to compute g^{ab} from g^a and g^b for $a, b \in \mathbb{Z}/q\mathbb{Z}$ random.

Remark 4.1.2 (Negligible success probability)**.** We rephrase the hardness of the CDH problem using negligible functions: The CDH problem in the group G is hard if any efficient algorithm $\mathcal{A}$ with input (G, q, g, g^a, g^b) and output $h \in G$ succeeds in outputting $h = g^{ab}$ with only negligible probability $\mathsf{negl}(\log q)$.

The formal definition of a negligible function negl can be found in [Gol00, Def. 1.3.5]; however, the definition is far less enlightening than the name. Since our functions are functions in the size of the input (that is, in $\log q$), an example of a negligible function is $2^{-\log q} = 1/q$ (guessing at random), an example of a non-negligible function is $\frac{1}{\log q}$ (*exponentially better* than guessing at random).

Discrete logarithm. In the scenario above, the attacker succeeds if they can compute the *shared key* $g^{ab} \in G$. An even stronger attacker might be able to determine the secret keys sk_A or sk_B from the public keys pk_A, resp. pk_B.

Definition 4.1.3. The *Discrete logarithm problem* (DLP) problem in the group G is said to be hard if there is no efficient algorithm to compute a, given (G, q, g, g^a) for $a \in \mathbb{Z}/q\mathbb{Z}$ random.

Clearly an attacker who can solve the DLP can solve the CDH problem. In cryptanalysis, it is usually the DLP problem that is studied. There are groups in which these problems are equivalent but also groups in which the DLP is considered hard while the CDH is easy [MW00].

Decisional Diffie–Hellman problem. Anouk and Bas perform the key exchange not only to share a secret among them, but to use this secret in other protocols. Consider the following ElGamal encryption protocol, in which the shared secret is *masking* the message. Anouk wants to send a message $m \in G$ to Bas. Anouk takes the public key $\mathsf{pk}_B = g^b$ of Bas, generates a new $y \in \mathbb{Z}/q\mathbb{Z}$, and

sends Bas the pair $(c_1, c_2) = (g^y, (g^b)^y \cdot m)$. Bas can then decrypt the message by taking the tuple (c_1, c_2) and their secret key $\mathsf{sk}_B = b$ and compute $c_2 \cdot (c_1^b)^{-1} = m$.

Despite all the incredible literary works of art humanity has created, most human communication is fairly mundane and repetitive. Assume therefore that there is only a small set of possible messages: for simplicity, say, $m \in \{1, h\}$ for $h \in G$ random. Therefore, an attacker seeing the pair (c_1, c_2) on the channel, can see that $c_2 = g^{by}$ or $c_2 = g^{by} \cdot h$. In the first case, it is the shared key for g^b and g^y, in the second case, it is a random element in G. Any attacker that can distinguish between the two can determine which message Anouk was sending.

This leads to the definition of the *Decisional Diffie–Hellman* problem (DDH):

Definition 4.1.4. We say that the *Decisional Diffie–Hellman problem* (DDH) problem is hard in G if any attacker seeing (G, q, g, g^a, g^b, h) for $a, b \in \mathbb{Z}/q\mathbb{Z}$ random and for which $h = g^{ab}$ with probability $1/2$ and $h \in G$ uniformly random otherwise, can decide whether $h = g^{ab}$ with probability *at most* $1/2 + \mathsf{negl}(\log q)$.

The success probability $1/2 + \mathsf{negl}$ intuitively means "not significantly better than a random guess". A tuple of the shape (g^a, g^b, g^{ab}) is called a *DDH tuple* or *DDH sample*. The DDH problem is therefore to distinguish between DDH tuples and tuples of random elements.

Example 4.1.5 (Easy DDH)**.** The DDH problem is not hard for $(\mathbb{Z}/p\mathbb{Z})^\times$ (c.f. Section 4.1.1). To see this, we use the Legendre symbol

$$\left(\frac{x}{p}\right) = \begin{cases} 1, & \text{if } x \text{ is a square mod } p, \\ -1, & \text{if } x \text{ is a not a square mod } p. \end{cases} \tag{4.1}$$

For a generator $g \in (\mathbb{Z}/p\mathbb{Z})^\times$ and any a, we have $\left(\frac{g^a}{p}\right) = \left(\frac{g}{p}\right)^a = (-1)^a$. We denote the resulting quadratic character by χ (as we will do in Chapter 5):

$$\chi(x) = \left(\frac{x}{p}\right). \tag{4.2}$$

The quadratic character $\chi(g^a) = \chi(g)^a$ is clearly determined by the parity of the exponent a. We emphasize that we do not have access to the exponents a, b, but only the group elements g^a, g^b.

In the DDH problem, we need to decide whether in the tuple (g^a, g^b, h) the element h is random or $h = g^{ab}$. We observe that $\chi(g^{ab}) = (-1)^{ab}$, but note that we do not immediately have the correct g^{ab} at our disposal. We can, however, *deduce* the correct value of $\chi(g^{ab})$ from $\chi(g^a)$ and $\chi(g^b)$.

First, assume that $\chi(g^a) = \chi(g^b) = -1$ (the exponents are odd). Then we *always* have $\chi(g^{ab}) = -1$ ("odd times odd equals odd"). In all other cases, at least one of the exponents is even, and ab is even, so we always have $\chi(g^{ab}) = 1$. In either of these cases, if h is random, then $\chi(h)$ has the correct value only half of the time. This leads to a distinguisher with success probability $3/4$, which is noticeably bigger than $1/2$.

Note that this attack can be avoided if we restrict to the subgroup of quadratic residues in $(\mathbb{Z}/p\mathbb{Z})^\times$: the Legendre character is trivial there. If we choose p such that $p = 2r + 1$ for some prime r, then DDH is believed hard in the subgroup of quadratic residues.

Remark 4.1.6 (Practical hardness)**.** Hidden constants in asymptotic statements can make a big difference in practice. A cryptographic problem is considered *hard in practice* if the adversary $\mathcal{A}$ with running time bounded by 2^{128} can break the problem with probability at most 2^{-128} (we say the problem has 128 *bits of security*). Based on our current understanding of the practical hardness of CDH and discrete log, we use elliptic curves over finite fields with at least $p \approx 2^{256}$, or subgroups of size at least $q = 2^{256}$ in a finite field $\mathbb{F}_p$ with $p \gtrsim 2^{3072}$ (see for instance [CSA18]).

Other notions of security. The discussion above considers passive security only in a very limiting model: the adversary can listen to the conversation between Anouk and Bas on an open channel, but cannot interfere in their communication. The key exchange also requires an *authenticated channel*, in which Anouk and Bas are talking to each other: the adversary cannot try to impersonate one of the players. Neither is the attacker nefariously trying to obtain the secret key of the other party by other means.

In Chapters 6 and 7 we study security of schemes in a very different model: we study the security of the specific implementations of cryptographic schemes. In Chapter 6 we assume the attacker can time Anouk's computation and try to deduce information about Anouk's secret; in Chapter 7 we assume physical access to Anouk's device so that the attacker can induce errors in Anouk's computation and try to recover Anouk's secret key.

Quantum easiness. The DLP in finite fields or elliptic curves is considered sufficiently hard to be used in practice, however, it can be solved efficiently using a large-scale quantum computer. Shor's algorithm [Sho99] solves the problem in quantum polynomial time and requiring a polynomial number of qubits.

4.1.2 Group actions in cryptography

Shor's algorithm crucially uses the fact that we can efficiently compose group elements in G. In an attempt to hide the group structure, one can generalize the Diffie–Hellman key agreement to group actions.

The following construction is due to Couveignes [Cou06] and Rostovtstev and Stolbunov [RS06]. First, we need a (left) *group action*. Let G be a group and let X be a set. A group action of G on X is a map $\star : G \times X \to X$ satisfying the following conditions:

- the identity $1 \in G$ acts trivially, that is, $1 \star x = x$ for all $x \in X$;

- the action respects the group operation: for any $g, h \in G$ and $x \in X$ we have $g \star (h \star x) = (gh) \star x$.

Usually we assume that G and X are finite; in the following, we will always assume that G is abelian and that X is non-empty.

We call the action *free* if for all $g \in G$ and $x \in X$ the property $g \star x = x$ implies $g = 1 \in G$. We call the group action *transitive* if for any $x, y \in X$ there exists $g \in G$ such that $y = g \star x$. The action is called *regular* if it is free and transitive; in a regular group action, for any $x, y \in X$ there exists a *unique* $g \in G$ such that $y = g \star x$. If the action of G on X is regular, upon fixing any base point $x \in X$ we obtain a bijection of sets $G \to X$ via $g \mapsto g \star x$, and we call X a *homogeneous space* for G (with the group action by G implicit and fixed).

Example 4.1.7 (Affine spaces)**.** In mathematics, a familiar example of a homogeneous space is an affine space $\mathcal{A}$ that is acted on by (the additive group) of its underlying vector space V by translations.

To make the group action interesting cryptographically, we require several more properties. Some of these are already in [Cou06]; the definition of an *effective group action* was formalized in [ADMP20, Def. 3.4]. An *effective* group action is a group action such that there exist efficient algorithms for the operations we typically wish to perform. For the operations in the group G, we require efficient algorithms for: testing whether a bit-string represents an element in G, testing equality of elements in G, sampling (typically uniformly random) elements from G, and composition and inversion of elements. For the set X, we again require efficient testing for membership and for computing an efficient unique representative. And, of course, computing the action $g \star x$ is required to be efficient for any $g \in G$ and $x \in X$.

Generalizing the Diffie–Hellman key agreement protocol to group actions is immediate [Cou06, RS06]: Anouk and Bas agree on an abelian group G, a homogeneous space X for G, and a base point $x \in X$. Anouk's secret is a random element $a \in G$, and the corresponding public key is $a \star x$. Bas's secret is a random element $b \in G$, and the public key is $b \star x$. Upon exchanging public keys, both Anouk and Bas can compute the shared value $a \star (b \star x) = ab \star x = b \star (a \star x)$. The equality relies on the fact that G is abelian.

The following two definitions, aptly named with the affine space example in mind, are analogues of the discrete logarithm problem and the computational Diffie–Hellman problem, respectively. For brevity, we suppress the description of the group action as input to the algorithms in the following definitions.

Definition 4.1.8 (Vectorization)**.** The vectorization problem is hard if there is no efficient algorithm that on input (x, y), for $x, y \in X$ random elements, outputs an element $a \in G$ satisfying $y = a \star x$.

Definition 4.1.9 (Parallelization)**.** The parallelization problem is hard if there is no efficient algorithm that on input the tuple $(x, a \star x, y)$, for $x, y \in X$ and $a \in G$ random, outputs $z \in X$ satisfying $z = a \star y$.

In the example of an affine space $\mathcal{A}$ being acted on by its underlying vector space, the parallelization problem is to translate the point y by a vector a parallel to the vector given x and $a \star x$. In general, it is the analogue of the CDH problem: note that by regularity, we can write $y = b \star x$ for some $b \in G$. The parallelization problem then asks for $z = a \star y = (ab) \star x$.

A homogeneous space X for G for which the vectorization and parallelization problems are hard is called a *hard homogeneous space* [Cou06].

As in the discussion of the ElGamal protocol, we might wish for an analogue of the decisional Diffie–Hellman problem. This problem was called *parallel testing* by Couveignes but this name is not widely used.

In [CSV20], the following problem is called DDH-CGA to emphasize it is the DDH problem in the group action setting, or more specifically in the context of CRS/CSIDH. We will use the shorter version DDH from now on as we will only talk about the computational assumptions in the group-action setting, and no confusion should arise.

Definition 4.1.10 (Decisional Diffie–Hellman)**.** The Decisional Diffie–Hellman problem for the group action of G on X is hard if there is no efficient algorithm distinguishing the following scenarios, each happening with probability $1/2$: for $x \in X$ and $a, b \in G$ random,

1. in the tuple $(x, a \star x, b \star x, y)$, the point $y = (ab) \star x$;
2. in the tuple $(x, a \star x, b \star x, y)$, the point $y \in X$ is random.

4.1.3 Isogeny-based group action

Couveignes [Cou06] and Rostovtsev and Stolbunov [RS06] independently proposed an instantiation of a group-action framework discussed in Section 4.1.2 based on the complex multiplication theory (cf. Section 2.4.1). The protocol is now known as the CRS protocol.

Fix a finite field $\mathbb{F}_q$, an imaginary quadratic order $\mathcal{O}$ and a trace t and assume that the set $X = \mathcal{E}\ell\ell_q(\mathcal{O}, t)$ is not empty. The group $G = \mathrm{Cl}(\mathcal{O})$ is the class group of the order $\mathcal{O}$ and by Theorem 2.4.4 acts regularly on the set $\mathcal{E}\ell\ell_q(\mathcal{O}, t)$.

The action $\mathrm{Cl}(\mathcal{O}) \times \mathcal{E}\ell\ell_q(\mathcal{O}, t) \to \mathcal{E}\ell\ell_q(\mathcal{O}, t)$ is computed via isogenies: to compute the action of some ideal class $[\mathfrak{a}]$ on $E \in \mathcal{E}\ell\ell_q(\mathcal{O}, t)$, represent the class $[\mathfrak{a}]$ by some ideal $\mathfrak{a} \subset \mathcal{O}$ and compute the isogeny $E \to E/E[\mathfrak{a}]$. Then

$$[\mathfrak{a}] \star E = E/E[\mathfrak{a}].$$

This is a well-defined abelian group action by Section 2.4.1 and it is believed that $\mathcal{E}\ell\ell_q(\mathcal{O}, t)$ is a hard homogeneous space for $\mathrm{Cl}(\mathcal{O})$ [Cou06, RS06, CLM+18]

Before discussing the particular choices for q, t and $\mathcal{O}$ in Section 4.2, we quickly include some notes on the security of these schemes. We primarily focus on the hardness of the vectorization problem (cf. Definition 4.1.8).

Remark 4.1.11 (Classical security)**.** The best classical attack on the vectorization problem is a meet-in-the-middle attack running in $O(\sqrt{|\Delta_{\mathcal{O}}|})$; a low-memory version was developed for the supersingular case in [DG16] and runs in $O(\sqrt[4]{p})$, as in this case we have $\Delta_{\mathcal{O}} = O(\sqrt{p})$.

In practice, secret keys are restricted to a subset of $\mathrm{Cl}(\mathcal{O})$; to prevent the meet-in-the-middle attack it is typically chosen to have size $\approx 2^{256}$ for 128 bits of security. We discuss choices for keyspaces in Section 4.3.4.

Remark 4.1.12 (Quantum security)**.** Childs, Jao, and Soukharev [CJS14] showed that vectorization is an instance of the hidden-subgroup problem. This problem can be solved using Kuperberg's algorithm with $L_{\sqrt{p}}(1/2)$ calls to a quantum oracle with

$$L_{\sqrt{p}}(1/2) = e^{\log^{1/2}(\sqrt{p})\log\log^{1/2}(\sqrt{p})+o(1)}.$$

For concrete parameters, the number of oracle calls was further analyzed in [BS20] and [Pei20]; [BLMP19] 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 perfect qubits, i.e., not taking into account the overhead for quantum error correction. The paper [CCJR22] extends the quantum analysis and suggest the smallest parameters should 4096-bit primes.

Implementation papers such as CTIDH [BBC+21] use the CSIDH-512 prime for comparison purposes and also offer larger parameters. Likewise, we use the CSIDH-512 parameters for concrete examples.

SIDH attacks do not impact the security of CSIDH. We emphasize that CSIDH, its variants, and the protocols based on the CSIDH group action are *not* affected by the recent attacks that break the isogeny-based scheme SIDH [CD23, MM22, Rob23]. These attacks exploit specific auxiliary information which is revealed in SIDH but does not exist in CSIDH.

4.2 CSIDH setup

In Section 4.1.3, we discussed the group action of $\mathrm{Cl}(\mathcal{O})$ on $\mathcal{E}\ell\ell_q(\mathcal{O}, t)$. In this section, we explain what choices of $q, \mathcal{O}, t$ are made in the scheme CSIDH [CLM+18] so that the group action can be computed efficiently.

4.2.1 Choosing group elements to act by

Acting by large norm ideals in $\mathcal{O}$ means having to compute large-degree isogenies, which is expensive in general. To counter this, Couveignes [Cou06] suggests computing the group structure of $\mathrm{Cl}(\mathcal{O})$ using a factor basis of ideal classes represented by ideals $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ of small norm. However, computing the class group is still very expensive. The record computation of a 512-bit class group in CSI-FiSh [BKV19] shows that analogous computations for larger parameters are currently impractical. Computing class groups will become possible with a large-scale quantum computer, leading to more efficient instantiations. Some joke that this makes the CRS construction a truly post-quantum protocol.

Should one prefer not to wait for the arrival of large-scale quantum computers to instantiate the scheme, one can avoid computing the class group altogether, and act only by the subgroup $G = \langle \mathfrak{l}_1, \ldots, \mathfrak{l}_n \rangle \subset \mathrm{Cl}(\mathcal{O})$. This leads to the definition of a *restricted efficient group action (REGA)* framework [Cou06, ADMP20].

Restricting the action to only acting by combinations of elements $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ induces an action of $\mathbb{Z}^n$ on $\mathcal{Ell}_q(\mathcal{O}, t)$. A vector $(e_1, \ldots, e_n) \in \mathbb{Z}^n$ acts on an elliptic curve $E \in \mathcal{Ell}_q(\mathcal{O}, t)$ as

$$E \mapsto (\mathfrak{l}_1^{e_1} \ldots \mathfrak{l}_n^{e_n}) \star E.$$

Here, negative exponents $e_j < 0$ are interpreted as acting by the ideal $\mathfrak{l}_j^{-1}$ a total of $|e_j|$-times. Since the action of $\mathrm{Cl}(\mathcal{O})$ on $\mathcal{Ell}_q(\mathcal{O}, t)$ is commutative, the order in which the action of the various $\mathfrak{l}_j$ is computed is irrelevant. This leads to an interesting space for optimization, see Section 4.3.3.

From now on, the secret keys are given by the *exponent vectors* $(e_1, \ldots, e_n)$. If the starting curve E_0 is fixed, the curve $E_A = (\mathfrak{l}_1^{e_1} \ldots \mathfrak{l}_n^{e_n}) \star E_0$ corresponding to this secret key is referred to in the later chapters as the public key.

Remark 4.2.1. In the restricted group action framework, we no longer have a way of generating uniformly random elements $\mathfrak{a} \in \mathrm{Cl}(\mathcal{O})$. Even if we do know the relation lattice of the elements $\mathfrak{l}_i$, being able to evaluate the action by a uniformly random element currently requires solving (an approximate) closest vector problem in the relations lattice [BKV19, ADMP20], which is not known to be possible efficiently (in polynomial time) [Pan23].

This is not an issue for key exchange, however, other cryptographic protocols (such as digital signatures) do require uniform elements. A polynomial-time solution for digital signatures is the SeaSign protocol [DG19]; a polynomial-time algorithm to evaluate the group action is given by the Clapotis protocol [PR23],

" Of course, 'polynomial-time' and 'practical' are not the same thing." [Gal23]

Picking degrees. De Feo, Kieffer, and Smith [DKS18] note that computing the action by $\mathfrak{l}$ is significantly more efficient if $E[\mathfrak{l}]$ is contained in $E(\mathbb{F}_{q^k})$ for some small extension degree k. Instead of acting by *all* ideals of small norm, they

use primes ℓ_i for which the action is the *cheapest* to compute. The most efficient way to compute the isogeny steps is using Vélu's formulas or analogues (see Section 4.3.2), which requires computing the points in the kernel of the isogeny.

De Feo, Kieffer, and Smith select a suitable large prime characteristics p and spend thousands of (CPU) hours searching for curves E that admit many rational points over $\mathbb{F}_{p^k}$ for $k \in \{1, 2\}$. This makes the resulting CRS protocol efficient in the mathematician's sense: public-key generation can be computed in less than 10 minutes without optimization.

Example 4.2.2 ([DKS18, Secs. 4 and 6])**.** One parameter set is using ordinary elliptic curves over a prime field $\mathbb{F}_p$ and trace t such that $p+1\pm t$ (the cardinality of the curve or its twist) is divisible by the following primes

$$\ell \in \{3, 5, 7, 11, 13, 17, 103, 523, 821, 947, 1723\}. \tag{4.3}$$

There are further torsion points defined over small extensions ($k \leq 9$), producing a total of 25 primes for which the isogenies can be evaluated rather cheaply. The corresponding isogeny steps are called the *Vélu steps*.

To obtain a large enough keyspace, they include the computation of isogeny steps for a further 29 primes $23 \leq \ell_j \leq 359$, for which the isogeny steps need to be evaluated as the so-called *Elkies steps* (see [DKS18, Algs. 3 and 4]), which are orders of magnitude slower than Vélu's formulas.

The largest permissible exponent e_j varies per degree ℓ_j: cheaper steps are performed more often: $e_j \leq 409$ for the Vélu steps (4.3), and $e_j \leq 20$ for the Elkies steps (in fact, most of these satisfy $e_j \leq 2$). See Remark 4.3.10 for intuition as to why it is often better to include more expensive steps rather than increase exponents for the cheaper steps.

Remark 4.2.3. A different instantiation of the group action was given by the SCALLOP signature scheme [DFK+23]. The rough idea is to put oneself in a situation in which the structure of the class group is known and favorable for computation, bypassing the computational complexity of CSI-FiSh. This is achieved by using *oriented* supersingular elliptic curves over $\mathbb{F}_{p^2}$, as in Remark 2.5.8.

Since the class number of an order of conductor f is given by

$$h(\mathcal{O}) = \frac{h(\mathcal{O}_K) f}{[\mathcal{O}_K^* : \mathcal{O}^*]} \prod_{p \mid f} \left(1 - \left(\frac{\Delta_{\mathcal{O}_K}}{p}\right) \frac{1}{p}\right), \tag{4.4}$$

one can obtain favorable parameter sets by setting $\mathcal{O}_K = \mathbb{Z}[i]$, choosing a prime p such that $p^2 - 1$ has a large smooth factor (to be able to evaluate small-degree isogenies efficiently), and the conductor f a large prime such that $f - \left(\frac{-1}{f}\right)$ (determining the class number) is as smooth as possible.

4.2.2 CSIDH primes

To make the CRS protocol practical even in the cryptographic sense, Castryck, Langle, Martindale, Panny and Renes in CSIDH [CLM+18] note that for supersingular curves over $\mathbb{F}_p$, the group order is $p+1$. Therefore, searching for curves with a favorable number of points reduces to searching for a suitable prime. In CSIDH, one chooses primes of the form $p = 4 \cdot \ell_1 \cdot \cdots \cdot \ell_n - 1$ and the *non-maximal* order $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. Assuming further that $p \equiv 3 \bmod 8$ yields further benefits in simplicity and efficiency.

Example 4.2.4. CSIDH-512 [CLM+18] uses the $p \equiv 3 \pmod 8$ defined by:

$$p + 1 = 4 \cdot 3 \cdot 5 \cdot \ldots \cdot 373 \cdot 587.$$

Therefore, $\ell_1 = 3, \ldots, \ell_{74} = 587$, so we have 74 different small primes ℓ at our disposal. The original keyspace is chosen as $\mathcal{K} = \{-5, \ldots, 5\}^{74}$, so in expectation we have to compute about 3 isogenies per degree. In practice we use different keyspaces (Section 4.3.4).

One reason for the choice of $p \equiv 3 \bmod 4$ is that the curve $E_0 : y^2 = x^3 + x$ with j-invariant 1728 is supersingular over $\mathbb{F}_p$, and has endomorphism ring $\mathbb{Z}[\sqrt{-p}]$: There is only one non-trivial point of order 2 defined over $\mathbb{F}_p$, so $\frac{\pi - 1}{2} \notin \mathrm{End}(E_0)$.

Another reason is that all elliptic curves in the $\mathrm{Cl}(\mathbb{Z}[\sqrt{-p}])$-orbit of E_0 can be represented by Montgomery models:

Definition 4.2.5 (Montgomery form)**.** For any $A \in \mathbb{F}_q \backslash \{-2, 2\}$, the elliptic curve E_A in *Montgomery form* is given by the equation

$$E_A : y^2 = x^3 + Ax^2 + x. \tag{4.5}$$

We call A the *Montgomery coefficient* of E_A.

The condition $A \neq \pm 2$ guarantees that the cubic $x^3 + Ax^2 + x$ does not have repeated roots. Montgomery curves allow for computing with x-only arithmetic for scalar multiplication using the *Montgomery ladder*, computing isogenies, and compress the information about the curve E into a single element in $\mathbb{F}_q$. Details on the arithmetic of Montgomery curves can be found, for instance, in [CS18].

Clearly not all curves admit a Montgomery form (for instance, because the point $(0, 0)$ is a rational 2-torsion point on E_A). Fortunately, curves in the orbit of E_0 admit a *unique* Montgomery form:

Proposition 4.2.6 ([CLM+18, Proposition 8])**.** *Assume that $p \equiv 3 \pmod 8$ and $E/\mathbb{F}_p$ be supersingular. Then $\mathrm{End}_{\mathbb{F}_p} = \mathbb{Z}[\sqrt{-p}]$ if and only if E is isomorphic over $\mathbb{F}_p$ to the curve $E_A : y^2 = x^3 + Ax^2 + x$. If such an A exists, it is unique.*

From now on, we restrict to the case $p \equiv 3 \pmod 8$. Montgomery curves give us unique representatives for the classes in $\mathcal{E}\ell\ell_p(\mathbb{Z}[\sqrt{-p}], 0)$, and we write

$$\mathcal{E} = \mathcal{E}\ell\ell_p(\mathbb{Z}[\sqrt{-p}], 0) = \{E_A/\mathbb{F}_p : E_A \text{ supersingular}\}.$$

We also write $\mathcal{M} = \{A : E_A \in \mathcal{E}\}$ for the set of Montgomery coefficients (in particular, in Chapter 6).

Example 4.2.7 (Twists)**.** For $A \neq 0$, the quadratic twist of $E_A \in \mathcal{E}$ is $E_{-A} \in \mathcal{E}$. They are isomorphic over $\mathbb{F}_{p^2}$ via the map $(x, y) \mapsto (-x, iy)$ for any i with $i^2 = -1$.

For the curve $E_0 : y^2 = x^3 + x$ the quadratic twist $\tilde{E}_0 : y^2 = x^3 - x$ does *not* admit a Montgomery form. One reason is because it has full 2-torsion and therefore the endomorphism ring is $\mathrm{End}_{\mathbb{F}_p}(\tilde{E}_0) \cong \mathbb{Z}[\frac{\sqrt{-p}-1}{2}]$, which means it cannot admit a Montgomery form by Proposition 4.2.6.

Since E_0 (which is the curve with $j(E_0) = 1728$) is the only j-invariant for which the quadratic twist of E_A does not lie in $\mathcal{E}$. We conclude that $\#\mathcal{E}$ is odd. This result is classical; it is a consequence for instance of genus theory, which describes the 2-part of the class group (cf. Section 5.3.1).

Remark 4.2.8 (Other possible choices)**.** We note several other parameter choices for schemes that extend the CSIDH protocol.

CSURF [CD20] uses supersingular elliptic curves over $\mathbb{F}_p$ with $p \equiv 7 \bmod 8$, the *maximal* order $\mathcal{O} = \mathbb{Z}\left[\frac{1+\sqrt{-p}}{2}\right]$, with the smallest p chosen as

$$p + 1 = 8 \cdot 3^2 \cdot \ldots \cdot \widehat{347} \cdot \ldots \cdot \widehat{359} \cdot \ldots \cdot 389.$$

CSURF can no longer rely on Montgomery models, but the authors prove that even in this setting there is a convenient model for the curves.

Summary. In CSIDH, one chooses $\mathcal{E} = \mathcal{E}\ell\ell_p(\mathbb{Z}[\sqrt{-p}], 0)$ the set of supersingular elliptic curves with endomorphism ring $\mathbb{Z}[\sqrt{-p}]$. Taking a prime $p \equiv 3 \pmod 8$ allows for representing the curves in $\mathcal{E}$ by a Montgomery coefficient.

Further, the prime p is chosen such that $\#E(\mathbb{F}_p) = p + 1$ is divisible by many small primes, so there exist rational points of small order. Recall that this is desirable as a rational ℓ-torsion point generates the kernel of the isogeny corresponding to the action by a split ideal $\mathfrak{l} = (\ell, \pi - 1)$ (see Example 2.4.2).

Moreover, for $A \neq 0$, the quadratic twist of $E_A \in \mathcal{E}$ is $E_{-A} \in \mathcal{E}$. All these favorable properties are exploited when computing the action of $\mathrm{Cl}(\mathbb{Z}[\sqrt{-p}])$ on $\mathcal{E}$.

4.3 Algorithmic aspects

In this section, we discuss some of the algorithmic aspects of computing the CSIDH group action. From now on, we focus on the evaluation of the action

$$(\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}) * E$$

for a vector $(e_1, \ldots, e_n) \in \mathcal{K} \subset \mathbb{Z}^n$, and the related choices of $\mathcal{K}$.

First, we discuss how to determine the correct kernels of the isogenies corresponding to the action by a single $\mathfrak{l}^{\pm 1}$, then how to compute such an isogeny and amortize some of the costs by computing several isogenies from one torsion point, using *strategies*. Finally, we discuss how the choice of the keyspace influences the resulting efficiency of the scheme.

4.3.1 Determining the kernel of the isogenies

Assume that ℓ is an odd prime such that $\ell \mid p+1$ and $\ell^2 \nmid p+1$ (as is the case in CSIDH). First we compute how to evaluate the action by the ideals $\mathfrak{l} = (\ell, \pi - 1)$ and $\mathfrak{l}^{-1} = (\ell, \pi + 1)$. We call such action an *isogeny step*, taking us from the domain curve E_A to some new curve $E_{A'}$.

In Chapter 7, we call the action by $\mathfrak{l}$ a *positive* step and the action by $\mathfrak{l}^{-1}$ a *negative* step (leaving the degree of the isogeny implicit). As the notation suggests, a negative and a positive step of the same degree cancel each other, see Remark 2.4.5. Any point generating the kernel of a positive (negative) isogeny step is said to have *positive (negative) orientation*; we also say that the point is positive (resp. negative). In Chapter 7, we denote the orientation of a point by s, interpreted as ± 1. We stress that this terminology is CSIDH-specific: the discussion below crucially relies on the ideal (ℓ) factoring as $(\ell, \pi - 1)(\ell, \pi + 1)$. This means that the eigenvalues of Frobenius at ℓ are ± 1.

Positive steps. The action by $\mathfrak{l} = (\ell, \pi - 1)$ is easy. Example 2.4.2 and our assumptions on ℓ imply that $E_A(\mathbb{F}_p)[\ell]$ is cyclic of order ℓ, and so the kernel $E[\mathfrak{l}]$ is generated by any point $P \in E(\mathbb{F}_p)$ of order ℓ. Therefore $\mathfrak{l} \star E_A = E_{A'} = E_A/\langle P \rangle$.

Negative steps. To compute the action by $\mathfrak{l}^{-1} = (\ell, \pi + 1)$, we note that $E[\mathfrak{l}^{-1}]$ consists of points of order ℓ in $E(\overline{\mathbb{F}}_p)$ on which Frobenius acts as $\pi(P) = -P$.

Fix $i \in \mathbb{F}_{p^2}$ with $i^2 = -1$. For any generator P of $E[\mathfrak{l}^{-1}]$, write $P = (x, iy)$ for some $x, y \in \overline{\mathbb{F}}_p$. We can compute the action of π directly (using $p \equiv 3 \bmod 4$):

$$\pi(P) = (x^p, (iy)^p) = (x^p, -iy^p) \tag{4.6}$$

$$-P = (x, -iy). \tag{4.7}$$

Comparing (4.6) and (4.7), we conclude that $x, y \in \mathbb{F}_p$. So, the kernel $E[\mathfrak{l}^{-1}]$ of the negative step is generated by an ℓ-torsion point in $E(\mathbb{F}_{p^2})$ with x-coordinate in $\mathbb{F}_p$ and y-coordinate in $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$.

This is another reason why we use Montgomery curves and x-only arithmetic: even when computing negative isogeny steps, the x-coordinates are still in $\mathbb{F}_p$.

Sampling points of order ℓ. No efficient way is known to deterministically generate points of order ℓ in $E_A(\mathbb{F}_p)$. However, the following Las Vegas algorithm is popular for its simplicity and efficiency: sample a uniformly random point P, 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 $Q \neq \infty$.

To sample a point of order ℓ in $\tilde{E}_A(\mathbb{F}_p)$, we can use the following property:

Proposition 4.3.1. *Fix $i \in \mathbb{F}_{p^2}$ with $i^2 = -1$. Define the* twist group

$$\tilde{E}_A(\mathbb{F}_p) = \{(x, iy) \in E_A(\mathbb{F}_{p^2}) : x, y \in \mathbb{F}_p\} \cup \{\infty_{E_A}\}$$

Then $\tilde{E}_A(\mathbb{F}_p)$ is the image of $E_{-A}(\mathbb{F}_p)$ under the isomorphism $(x, y) \mapsto (-x, iy)$. In particular, $\tilde{E}_A(\mathbb{F}_p)$ is closed under addition on E_A.

We use the same notation $\tilde{E}$ for the twist of E and the image of the rational points of $\tilde{E}$ in $E(\mathbb{F}_{p^2})$. The points in $\tilde{E}(\mathbb{F}_p)$ are often thought of as lying on the twist (through the isomorphism above).

Corollary 4.3.2. *All positive points are in $E_A(\mathbb{F}_p)$ and all negative points lie in the twist group $\tilde{E}_A(\mathbb{F}_p)$.*

Therefore, to sample a negative point of order ℓ, we use an analogous Las Vegas algorithm as when sampling positive points: sample a uniformly random point $P \in \tilde{E}_A(\mathbb{F}_p)$, and compute $Q := [(p+1)/\ell]P$. The same analysis holds: since P is uniformly random in a cyclic group of order $p+1$, the point Q has order ℓ with probability $1 - 1/\ell$.

Remark 4.3.3 (Determining orientation)**.** On a Montgomery curve E_A, a point in E_A with x-coordinate $x \in \mathbb{F}_p^*$ is oriented positively if $x^3 + Ax^2 + x$ is a square in $\mathbb{F}_p$ and negatively otherwise; the orientation of points of order > 2 therefore agrees with the Legendre symbol of $x^3 + Ax^2 + x$ in $\mathbb{F}_p$.

Many state-of-the-art implementations (especially those using 2-point strategies, cf. Section 4.3.3) use the Elligator 2 map [BHKL13] to sample points both in $E_A(\mathbb{F}_p)$ and $\tilde{E}_A(\mathbb{F}_p)$.

Remark 4.3.4 (Mapping points)**.** The isogeny $\varphi : E_A \to \mathfrak{l} \star E_A$ maps the points of order ℓ on E_A to the point of ∞ on $\mathfrak{l} \star E_A$ (as those precisely generate the kernel of this isogeny). More interestingly, the points of order ℓ in $\tilde{E}_A(\mathbb{F}_p)$ are mapped to points of order ℓ in the twist group of $\mathfrak{l} \star E_A$.

This is easily seen from the definition of negative steps: the ℓ-torsion in the twist group $\mathfrak{l} \star E_A$ is the kernel of the negative step, but the ℓ-torsions in $E_A(\mathbb{F}_p)$ and $\tilde{E}_A(\mathbb{F}_p)$ jointly generate the ℓ-torsion in $E_A(\mathbb{F}_{p^2})$. To finish, use the explicit description of the kernel of the dual isogeny (proof of Proposition 2.2.8).

One can summarize this as: positive steps map negative points to negative points, and by symmetry, negative steps map positive points to positive points.

Remark 4.3.5 (Twisting)**.** Another way to compute negative isogenies is by using the following *quadratic twisting* trick, popularized in particular by [LGD21]:

$$\mathfrak{l}^{-1} \star E_A = (\widetilde{\mathfrak{l} \star E_{-A}}).$$

Therefore, to compute the negative step from E_A is the same as computing a positive step from its quadratic twist E_{-A}, and twisting.

4.3.2 Computing one isogeny

We have identified the kernels of the isogenies we want to compute (we have computed their kernels in Section 4.3.1), but we still need formulas to compute the target curve $E_{A'}$ of such an isogeny, and to evaluate the isogeny on points (which is essential to evaluating multiple isogeny steps efficiently, see Section 4.3.3).

As noted before, isogeny-based cryptography predominantly uses x-only arithmetic. The central computational task is therefore the following:

Definition 4.3.6 (xISOG)**.** Let $E_A/\mathbb{F}_q$ be an elliptic curve and let $P \in E$ be a point of order ℓ with x-coordinate in $\mathbb{F}_q$. Let $\varphi : E_A \to E_{A'} = E_A/\langle P \rangle$. The computational task xISOG is to:

1. compute the target curve $E_{A'}$, specifically its Montgomery coefficient;
2. for a possibly empty list of points $(Q_i) \in E$ with x-coordinates in $\mathbb{F}_q$, compute the x-coordinate of $\varphi(Q_i)$.

Typically, one computes the image of zero, one, or two points, see Section 4.3.3.

Vélu's formulas. Vélu's formulas [Vél71] are one of the main algorithms for xISOG: to compute both A' and evaluate $\varphi(Q)$ on any point. Vélu's formulas require $O(\ell)$ multiplications in $\mathbb{F}_q$.

The main computational task is to evaluate a *kernel polynomial*

$$h_S(X) = \prod_{s \in S} (X - x([s]P)) \tag{4.8}$$

for $S = \{1, 2, 3, \ldots, (\ell-1)/2\}$. All the arithmetic is done only using x-coordinates.

Algorithms to compute Vélu's formulas evaluate the product $h_S(X)$ by first computing $x(P), x([2]P), \ldots, x([(\ell-1)/2]P)$ and then evaluating the product (4.8). This costs $O(\ell)$ field multiplications: computing A' costs about 4ℓ field multiplications, computing the image of a point costs about 2ℓ multiplications.

√élu. For isogenies of degree $\ell \gtrsim 89$, √élu[1] [BDLS20] provide a significant speedup by using a baby-step-giant-step strategy. The main computational task is to evaluate a similar polynomial as in (4.8), for a different set S.

In √élu, the set $S = \{1, 3, 5, \ldots, \ell - 2\}$ is split into a "box" $U + V$ and "leftover" set W such that $S = (U+V) \cup W$. The product corresponding to $U+V$ is computed as the resultant of $h_U(X)$ and a polynomial related to $h_V(X)$, which leads to a quadratic speedup. Computing the whole $h_S(X)$ is then finished by multiplying by the product $h_W(X)$. See [BDLS20] for details. For each ℓ, one chooses U and V to minimize the total cost. Asymptotically, √élu uses $\tilde{O}(\sqrt{\ell})$ field multiplications.

Once can view √élu as a generalization of Vélu's formulas; indeed, these are recovered for U and V empty. This choice is optimal for small primes. For primes around $\ell = 89$, the √élu formulas typically achieve better performance, though the exact cutoff depends on lower-level arithmetic.

Remark 4.3.7. Both for Vélu's and √élu formulas, choosing a different kernel generator P' produces the same isogenies, not just isomorphic target curves.

Other formulas. For small degree isogenies, radical isogenies [CDV20] provide a significant speedup over Vélu's formulas. Other notable formulas for computing isogenies are Kohel's formulas [Koh96], which are very useful if the kernel points are defined over extension fields. In this case, one also can work with irrational factors of the kernel polynomials [EPSV23] or use the Frobenius action [BGDS23]. A novel way of computing isogenies of elliptic curves by embedding them in higher dimensional isogenies was given by Robert [Rob22a].

4.3.3 Strategies

In Section 4.3.1, we saw that kernel points can be probabilistically computed by sampling a random point T, and computing $[(p+1)/\ell]T$ to obtain a point of order ℓ. Once we obtain such a point, we use Vélu's formulas (or other means, see Section 4.3.2) to compute the isogeny.

However, this scalar multiplication is very costly. In practice, if p has 512 bits, so $(p+1)/\ell$ has almost 512 bits, so such a scalar multiplication costs thousands of field multiplications. We can amortize this cost by computing a series of isogenies after sampling one point on the initial curve, and mapping it through isogenies.

Example 4.3.8 (Pushing points through)**.** In the CSIDH setting, let Q be a point of order $\ell_1\ell_2$ on E_A, and set $R_1 = [\ell_2]Q$. Then R_1 is a point of order ℓ_1, and the image $R_2 = \varphi(Q)$ under $\phi : E_A \to E_A/\langle R_1 \rangle$ has order ℓ_2 on $E_A/\langle R_1 \rangle$.

So we can trade two large scalar multiplications for one large scalar multiplication (preparing the point Q as $[(p+1)/(\ell_1\ell_2)]P$), one small scalar multiplication (preparing $R_1 = [\ell_2]Q$) and one isogeny evaluation (computing $R_2 = \varphi(Q)$).

[1] Pronounced "square-root Vélu".

Definition 4.3.9 (Informal, following [DJP14])**.** A *strategy* is a method that computes a given series of isogenies. It specifies the degrees of the isogenies evaluated, the order in which they are evaluated, and, for each kernel point, whether it is obtained by a scalar multiplication of a random point (called *sampling*) or using an isogeny evaluation (called *pushing the point through*).

Multiplicative strategy. A simple *multiplicative* strategy was used in the algorithm of [CLM+18] and successors [BLMP19, MR18], see Algorithm 1.

Algorithm 1 Multiplicative strategy for evaluating 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 $\prod [\mathfrak{l}_i]^{e_i} * E_A = E_B$

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

The computation starts with a copy of the secret key $v = (e_1, \ldots, e_n)$. In each *round*, we sample a random point P and determine its orientation s. 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. The output is always interpreted as ± 1. Many implementations simply compute $s \leftarrow z^{\frac{p-1}{2}}$. Then we attempt to compute all isogeny steps with orientation s: steps with index $S = \{i : \text{sign}(e_i) = s\}$. Denote by $k = \prod_{i \in S} \ell_i$. We first compute one large scalar computation $Q = [\frac{p+1}{k}]P$ and, as in Example 4.3.8, use smaller scalar multiplications and isogeny evaluations to obtain kernel points at successive isogeny steps (possibly skipping some steps in Algorithm 1; we call this *missing torsion* in Chapter 7). At the end of each round, we update the vector v at each index that the isogeny computation succeeded. We proceed until $v = (0, \ldots, 0)$.

The scalar multiplications reduce in length at each step; processing the isogenies in decreasing order reduces the total cost [MR18].

SIMBA. In the multiplicative strategy for CSIDH (Algorithm 1), in every round we compute all isogeny steps we can. However, one can achieve a significant speedup with by *Splitting Isogenies into Multiple Batches* (SIMBA) [MCR19]. SIMBA partitions the set of isogeny degrees $\{\ell_1, \ldots, \ell_n\}$ into M batches, which

are called *prides* (the word *batch* we reserve for CTIDH, see Chapter 6). One round of computation computes the isogeny steps in one such pride.

In SIMBA-M, the i-th pride consists simply of all the isogeny steps with index $i, i+M, i+2M, \ldots$ Moreover, only keys with $e_i > 0$ are allowed, so we always evaluate isogenies in the positive direction and the prides are fixed during the computation. The scalar k in Algorithm 1 is replaced with $\ell_i \cdot \ell_{i+M} \cdot \ldots$, which is typically much smaller than the k for the multiplicative strategy. As a result, the first scalar multiplication is much more expensive, but the latter multiplications in Algorithm 1 are all significantly cheaper.

1-point, 2-point strategies. In Algorithm 1, we sample a point and compute a sequence isogenies in the *same* direction. This approach is called the *1-point approach*. One can also sample two points per round: one positive, and one negative, and compute an isogeny step for each i [OAYT19]. Both points need to be pushed through the isogeny in Algorithm 1. All constant-time implementations (see Section 4.3.4) use the *2-point approach*, e.g., [BBC+21, CCJR22]. Other strategies (such as [CR22]) require pushing through even more points.

Optimal strategies. For SIDH, it is possible to find *optimal strategies* which minimize the computational cost [DJP14]. However, in CSIDH, there are many more possibilites for which steps we combine into a strategy, the isogeny steps have pairwise different degrees, and some isogeny steps may fail because we sampled a random point of order not divisible by that degree. This leads to many more possible combinations [CR22]; imposing certain assumptions one can obtain conditionally optimal strategies [HLKA20]. For instance, one can generalize the SIMBA approach with point-pushing within prides as in [HLKA20]. However, the performance of such approaches for constant-time computations does not lead to significant speedups [CR22].

4.3.4 Key spaces and constant-time implementations

Finally, we discuss the choice of the key space $\mathcal{K} \subset \mathbb{Z}^n$. Recall that the secret key is a vector $(e_1, \ldots, e_n) \in \mathbb{Z}^n$, representing the action by $(\mathfrak{l}_1^{e_1} \ldots \mathfrak{l}_n^{e_n}) \star E$. The choice of $\mathcal{K}$ has a massive impact on efficiency, and different security properties. Next we discuss the concept of constant-time implementations, and discuss the key spaces that have been used in such implementations. This section is based on the introduction to [BBC+21].

First, one requirement is that the key space is large enough to prevent brute-force attacks, and, given the algebraic properties of the group action, to also prevent meet-in-the-middle attacks. Conventionally, we take $\#\mathcal{K} \geq 2^{2\lambda}$ for security 2^λ. Some authors have argued that smaller key spaces suffice, see [CCJR22].

Original CSIDH [CLM+18] uses the key space $\mathcal{K}_m = \{-m, \ldots, m\}^n \subset \mathbb{Z}^n$. For CSIDH-512, we have $n = 74$ and $m = 5$. Using different bounds m_i for each i

can improve speed (as suggesteed in [CLM+18, Rmk. 14] and shown in [MCR19]). The key space $\mathcal{K}_m := \prod_{i=1}^{n}\{-m_i, \ldots, m_i\}$ has size $\#\mathcal{K}_m = \prod_{i=1}^{n}(2m_i + 1)$; by small abuse of notation we denote by m the vector $(m_1, \ldots, m_n)$.

Remark 4.3.10 (Shape of the key space)**.** In general, it is more efficient to have *more* different isogeny degrees with *smaller* exponents: on average, we expect to compute fewer isogenies. Take as example the key space $\mathcal{K} = \{-2, -1, 0, -1, 2\}^3$. In expectation (as a rough approximation of variable-time computation), we would compute $6/5$ isogenies per index, so a total of $3 \cdot 6/5 = 3.6$ isogenies. A key space with the same size ($\#\mathcal{K} = 125$) with just one isogeny degree would be $\mathcal{K}' = \{-62, \ldots, 62\}$, for which we would expect to compute ≈ 31 isogenies.

For practical protocols, secure implementations need to be in *constant time*: not leaking information about the secret via the time the computation takes. In the CSIDH algorithm, a private key $(e_1, \ldots, e_n)$ requires us to compute $|e_i|$ isogenies of degree ℓ_i (regardless of the strategy), so the running time depends directly on the secret key. For CSIDH, various approaches have been proposed [MCR19, OAYT19, CCC+19, HLKA20, CR22, ACR23, CCJR22].

Defining constant time. Algorithms computing the isogeny group action are usually randomized (due to sampling points, see Section 4.3.1). In a randomized algorithm, each run depends on the random bits generated inside the algorithm. So a randomized algorithm computes a function from inputs to distributions over outputs. Similarly, the time taken by an algorithm is a function from inputs to distributions of times. *Constant time* means that the distribution of algorithm time for input i matches the distribution of algorithm time for input i'. If the input is a CSIDH curve and a private key, and the output is the result of the CSIDH action, then the algorithm time provides no information about the private key, and provides no information about the output.

Note that this does not mean that the time is deterministic. Deterministic time algorithms to evaluate the CSIDH group action (for instance [BLMP19], which was for quantum analysis and not for efficiency) are inefficient and not necesssary for stopping timing attacks.

To ensure the algorithm is constant time, it is necessary to avoid data flow from the secret inputs to branches and array indices. In practice, one must also take into account lower-level instructions such as variable-time division, or swaps.

Approaches to constant-time. The first to study constant-time implementations for CSIDH were Meyer, Campos and Reith [MCR19]. If we use strategies that evaluate positive and negative steps separately, the action by $(5, 5, 5..., 5)$ would require very different time to evaluate than the action by $(5, -5, \ldots, 5, -5)$, which is problematic for achieving constant-time behavior. They introduce the shifted key space $\mathcal{K}_m^+ = \{0, \ldots, 2m_i\}^n$, and always compute $2m_i$ isogenies, discarding the *dummy isogenies* beyond e_i steps. The dummy isogenies can be used

for computation that facilitates later steps in the algorithm. Nevertheless, this approach essentially doubles the cost of variable-time computing the group action for all keys. Finally, note that we have $\#\mathcal{K}_m^+ = \prod_{i=1}^n (m_i + 1)$.

This slowdown was partially mitigated by applying the "2-point strategy" to the key space $\mathcal{K}_m$ by [OAYT19]. They evaluate m_i isogeny steps for each i, but each iteration is about $1/3$ more expensive because of extra point evaluations. A dummy-free variant of the 2-point approach was proposed in [CCC+19], requiring roughly twice as many isogenies, but useful when protecting against fault attacks.

A different key space combining constant-time requirements and balancing cost was introduced in CTIDH [BBC+21], and will be studied in Chapter 6.

Remark 4.3.11. The dummy-free implementations [CCC+19, CR22, ACR23] replace pairs of dummy ℓ_i-isogenies by pairs of isogenies that effectively cancel each other. This approach requires fixing the parity of each exponent, so to reach the same key space size as before, it suffers a slowdown of factor 2.

Dummy-free implementations are interesting because they mitigate certain fault attacks (subject of study in Chapter 7), such as skipping isogenies: if the skipped isogeny was a dummy one, the result of the computation is still correct, which can be used to infer information about the secret key.

Chapter 5

Breaking DDH for class group actions

This chapter is based on the paper *Breaking the decisional Diffie-Hellman problem for class group actions using genus theory* [CSV20], and is joint work with Frederik Vercauteren and Wouter Castryck. The extended version of this work was published in [CSV22a].

The introduction and background sections have been shortened; the discussion of isogeny volcanoes has been mostly treated in Section 2.5. We also omit the discussion of higher dimensional abelian varieties [CSV22a, Sec. 6] and only mention it briefly in a short section on follow-up work in Section 5.8. The rest of the article has been edited mostly for typographical consistency.

5.1 Introduction

In this chapter, we turn our attention to the decisional Diffie–Hellman assumption (Definition 4.1.10), in particular for the group action of the class group $\mathrm{Cl}(\mathcal{O})$ of an imaginary quadratic order $\mathcal{O}$ on $\mathcal{E}\ell\ell_q(\mathcal{O}, t)$ for some q and t. Very little is known about the structure of $\mathrm{Cl}(\mathcal{O})$, and it was previously assumed that the structure is "sufficiently hidden" by the group action and cannot be exploited.

However, the 2-torsion part of $\mathrm{Cl}(\mathcal{O})$ can be described by genus theory [Cox89, I.§3 & II.§7]: for every odd prime $m_i \mid \Delta_\mathcal{O}$, there is an *assigned* quadratic character χ_i that can be computed in time polynomial in the size of m. Depending on $\Delta_\mathcal{O}$, there may be up to 2 additional assigned characters. Any non-trivial assigned character can be used to break the DDH assumption in the group $\mathrm{Cl}(\mathcal{O})$ similar to Example 4.1.5.

In Section 5.4, we show that if $\mathfrak{a} \subset \mathcal{O}$ has norm $\mathrm{N}(\mathfrak{a})$ coprime to m, the character χ_m is non-trivial, and the curves $E, E' \in \mathcal{E}\ell\ell_q(\mathcal{O}, t)$ satisfy $E' = \mathfrak{a} \star E$, we can compute $\chi_m([\mathfrak{a}])$ from the elliptic curves E and E'. Our approach relies on computing Tate pairings and walking to the floor of isogeny volcanoes, and so computes χ_i in time *exponential* in the size of m_i. However, heuristically, we expect that $\Delta_\mathcal{O}$ admits a small factor m_i such that χ_i is non-trivial and so heuristically, our attack works in polynomial time for a density 1 subset of quadratic orders. This is because $\mathrm{Cl}(\mathcal{O})[2]$ is only trivial in the case that $\Delta_\mathcal{O} = -q$ or $\Delta_\mathcal{O} = -4q$ with $q \equiv 3 \bmod 4$. We discuss a version of the attack that does not walk to the floor in Section 5.9.

For supersingular elliptic curves over $\mathbb{F}_p$ with $p \equiv 1 \bmod 4$ (Section 5.5), there is always a suitable non-trivial character. The resulting attack only requires a few exponentiations in $\mathbb{F}_p$. We stress that this case does *not* apply to CSIDH, which uses $p \equiv 3 \bmod 4$.

In Section 5.6 we analyze the impact on the DDH problem for class group actions, report on our implementation of the attack, and propose countermeasures. Section 5.7 concludes the main body and provides avenues for further research. In Section 5.8, we discuss some of the follow-up work that has been done since the publication of [CSV20].

5.2 High level overview of the attack

We first explain the attack in a simple (yet very general) setting. Fixing a base curve E, the class group action $\star$ gives us a representation of $\mathrm{Cl}(\mathcal{O})$ on the set $X = \mathcal{E}\ell\ell_q(\mathcal{O}, t)$ by mapping a class $[\mathfrak{a}]$ to $E' = [\mathfrak{a}] \star E$. For every odd prime divisor m of the discriminant $\Delta_\mathcal{O}$, genus theory provides a character

$$\chi : \mathrm{Cl}(\mathcal{O}) \to \{\pm 1\} : [\mathfrak{a}] \mapsto \left(\frac{\mathrm{N}(\mathfrak{a})}{m}\right),$$

where $\left(\frac{\cdot}{\cdot}\right)$ denotes the Legendre symbol and the representative $\mathfrak{a}$ of the class $[\mathfrak{a}]$ is chosen such that its norm $\mathrm{N}(\mathfrak{a})$ is coprime to m. The goal is to compute $\chi([\mathfrak{a}])$ given only the pair (E, E').

Let $\varphi : E \to E'$ denote the isogeny corresponding to $\mathfrak{a}$, then $\mathrm{N}(\mathfrak{a}) = \deg(\varphi)$, so to compute χ, it suffices to determine $\deg(\varphi) \bmod m$, up to non-zero squares in $\mathbb{Z}/m\mathbb{Z}$. Assume we know a tuple $(P, Q) \in E^2$ with $P \in E[m]$ and the corresponding tuple $(\varphi(P), \varphi(Q)) \in E'^2$, computing $\deg(\varphi) \bmod m$ is easy thanks to the compatibility of the reduced m-Tate pairing T_m:

$$T_m(\varphi(P), \varphi(Q)) = T_m(P, Q)^{\deg(\varphi)}\,.$$

If the pairing is non-trivial, both sides will be primitive m-th roots of unity, so computing discrete logs gives $\deg(\varphi) \bmod m$.

In practice, of course, we are not given such corresponding tuples (P, Q) and $(\varphi(P), \varphi(Q))$. The only information we really have about φ is that it is an $\mathbb{F}_q$-rational isogeny of degree coprime to m. However, there exist many other isogenies between E and E': namely one for every $\mathfrak{a}'$ representative of $[\mathfrak{a}]$. So even if we can recover information on $\deg(\varphi) = \mathrm{N}(\mathfrak{a})$, it needs to be information that is constant for the whole *class* $[\mathfrak{a}]$.

If $E(\mathbb{F}_q)$ has a *unique* subgroup of order m, then $E'(\mathbb{F}_q)$ similarly has such a unique subgroup, and furthermore, $\varphi(E(\mathbb{F}_q)[m]) = E'(\mathbb{F}_q)[m]$. If we let P be a generator of $E(\mathbb{F}_q)[m]$ and P' a generator of $E'(\mathbb{F}_q)[m]$, then we know there exists some $k \in [1, m-1]$ such that $\varphi(P) = kP'$. Note however, that if we assume we know a point Q and its image $\varphi(Q)$ (but not the image of P under φ), we do not learn anything since the values $T_m(kP', \varphi(Q)) = T_m(P', \varphi(Q))^k$ run through the whole of μ_m for $k = 1, \ldots, m-1$ and we do not know k.

The main insight now is that we do not need to recover $\deg(\varphi)$ exactly but only up to squares, so if we could recover $k^2 \deg(\varphi)$ then it is clear we can still compute $\chi([\mathfrak{a}])$. This hints at a possible solution as long as Q is somehow derived from P and that the *same* unknown scalar k can be used to compensate for the difference not only between $\varphi(P)$ and P', but also between $\varphi(Q)$ and Q'. Indeed, computing $T_m(P', Q')$ would then recover the correct value up to a square in the exponent, namely $T_m(P, Q)^{\deg(\varphi)k^2}$. The simplest choice clearly is to take $Q = P$ and $Q' = P'$.

We do not recover the exact value $\deg(\varphi) = \mathrm{N}(\mathfrak{a}) \bmod m$, but the value only up to squares. With the observation before that we can only recover a value that is constant among all the representatives of $[\mathfrak{a}]$, we see that we are computing a *quadratic character* $\mathrm{Cl}(\mathcal{O}) \to \{\pm 1\}$. All such *well-defined* characters are described by *genus theory*. Therefore, there is only hope of achieving this if $m \mid \Delta_{\mathcal{O}}$ is a ramified prime, and the resulting quadratic character is non-trivial.

If there is no $\mathbb{F}_q$-rational m^2-torsion, we show that the self-pairings $T_m(P, P)$ and $T_m(P', P')$ are non-trivial. This feature is specific to the Tate pairing. In the more general case of $v = \mathrm{val}_m(\#E(\mathbb{F}_q)) > 1$, we first walk down to the floor

of the m-isogeny volcano reaching a curve E_0 with $E_0(\mathbb{F}_q)[m^\infty] = \mathbb{Z}/(m^v)$, and then choose points P and P' of order m and corresponding points Q and Q' of order m^v satisfying $m^{v-1}Q = P$ and $m^{v-1}Q' = P'$. Note that also in this case, the same unknown scalar k will compensate for both differences.

To sum up, we use the Tate pairing of certain points to obtain information on $\deg\varphi$ (up to squares $\bmod m$). By genus theory, we see that we are actually computing the assigned characters χ of $\mathrm{Cl}(\mathcal{O})$ directly from curves E, E' in $\mathcal{E}\ell\ell_q(\mathcal{O}, t)$, denote it by $\chi(E, E')$.

Whenever the characters are non-trivial, their multiplicative property allows us to break DDH in $\mathcal{E}\ell\ell_q(\mathcal{O}, t)$: in the tuple $(E, [\mathfrak{a}]\star E, [\mathfrak{b}]\star E, E')$, if $E' = [\mathfrak{a}\mathfrak{b}]\star E$, then $\chi(E', [\mathfrak{b}]\star E) = \chi([\mathfrak{a}]) = \chi([\mathfrak{a}]\star E, E)$, which we can compute. If E' is a random curve and χ is non-trivial, this only happens half of the time.

5.3 Background

In this chapter, instead of ℓ we will use the letter m to denote a prime $m \neq p$. This is because the notation is more classical for character theory.

5.3.1 Genus theory

Genus theory studies which natural numbers arise as norms of ideals in a given ideal class of an imaginary quadratic order $\mathcal{O}$. It shows that this question is governed by the coset of $\mathrm{Cl}(\mathcal{O})^2$, the subgroup of squares inside $\mathrm{Cl}(\mathcal{O})$, to which the ideal class belongs. This section summarizes parts of [Cox89, I.§3 & II.§7].

Let $m_1 < m_2 < \ldots < m_r$ be the odd factors of $\Delta_\mathcal{O}$. If $\Delta_\mathcal{O} \equiv 1 \bmod 4$ then

$$\chi_i : (\mathbb{Z}/\Delta_\mathcal{O}\mathbb{Z})^* \to \{\pm 1\} : a \mapsto \left(\frac{a}{m_i}\right) \qquad (\text{for } i = 1, \ldots, r)$$

are called the *assigned characters* of $\mathcal{O}$. If $\Delta_\mathcal{O} = -4n \equiv 0 \bmod 4$, then we extend this list with δ if $n \equiv 1, 4, 5 \bmod 8$, with ϵ if $n \equiv 6 \bmod 8$, with $\delta\epsilon$ if $n \equiv 2 \bmod 8$, and with both δ and ϵ if $n \equiv 0 \bmod 8$. Here

$$\delta : a \mapsto (-1)^{(a-1)/2} \qquad \text{and} \qquad \epsilon : a \mapsto (-1)^{(a^2-1)/8}.$$

If $n \equiv 3, 7 \bmod 8$ then the list is not extended.

Let $\mu \in \{r, r+1, r+2\}$ denote the total number of assigned characters and consider the map $\Psi : (\mathbb{Z}/\Delta_\mathcal{O}\mathbb{Z})^* \to \{\pm 1\}^\mu$ having these assigned characters as its components. Then Ψ is surjective and its kernel H consists precisely of those integers that are coprime with (and that are considered modulo) $\Delta_\mathcal{O}$ and arise as norms of non-zero principal ideals of $\mathcal{O}$. This leads to a chain of maps

$$\Phi : \mathrm{Cl}(\mathcal{O}) \longrightarrow \frac{(\mathbb{Z}/\Delta_\mathcal{O}\mathbb{Z})^*}{H} \xrightarrow{\cong} \{\pm 1\}^\mu,$$

where the first map sends an ideal class $[\mathfrak{a}]$ to the norm of $\mathfrak{a}$ (it is always possible to choose a representant of norm coprime with $\Delta_{\mathcal{O}}$) and the second map is induced by Ψ. Genus theory tells us that $\ker \Phi = \mathrm{Cl}(\mathcal{O})^2$; the cosets of $\mathrm{Cl}(\mathcal{O})^2$ inside $\mathrm{Cl}(\mathcal{O})$ are called *genera*, with $\mathrm{Cl}(\mathcal{O})^2$ itself being referred to as the *principal genus*.

Remark 5.3.1. By abuse of notation, we can and will also view $\chi_1, \chi_2, \ldots, \chi_r, \delta, \epsilon$ as morphisms $\mathrm{Cl}(\mathcal{O}) \to \{\pm 1\}$, obtained by composing Φ with projection on the corresponding coordinate.

It can be shown that the image of Φ is a subgroup of $\{\pm 1\}^\mu$ having index 2, so that the cardinality of $\mathrm{Cl}(\mathcal{O})/\mathrm{Cl}(\mathcal{O})^2 \cong \mathrm{Cl}(\mathcal{O})[2]$ equals $2^{\mu-1}$. More precisely, if we write $\Delta_{\mathcal{O}} = -2^a b$ with $b = m_1^{e_1} m_2^{e_2} \cdots m_r^{e_r}$, then this is accounted for by

$$\chi_1^{e_1} \cdot \chi_2^{e_2} \cdots \chi_r^{e_r} \cdot \delta^{\frac{b+1}{2} \bmod 2} \cdot \epsilon^{a \bmod 2}, \tag{5.1}$$

which is non-trivial when viewed on $(\mathbb{Z}/\Delta_{\mathcal{O}}\mathbb{Z})^*$, but becomes trivial when viewed on $\mathrm{Cl}(\mathcal{O})$. For example, if $\Delta_{\mathcal{O}}$ is squarefree and congruent to 1 mod 4, then the image of Φ consists of those tuples in $\{\pm 1\}^r$ whose coordinates multiply to 1.

Our main goal is to break the DDH assumption in $\mathcal{Ell}_q(\mathcal{O}, t)$. To do this, we compute the assigned characters χ: on input two elliptic curves $E, E' \in \mathcal{Ell}_q(\mathcal{O}, t)$ that are connected by a secret ideal class $[\mathfrak{a}] \in \mathrm{Cl}(\mathcal{O})$, we will describe how to compute $\chi(E, E') := \chi([\mathfrak{a}])$. Note that to break the DDH assumption, it suffices to compute the most convenient choice of character χ.

Example 5.3.2. In Section 5.5, we will study supersingular elliptic curves defined over $\mathbb{F}_p$ with $p \equiv 1 \bmod 4$. Here $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ has discriminant $-4p$, thus there are two assigned characters: the character δ and the Legendre character χ associated with p. But Equation (5.1) tells us that $\chi([\mathfrak{a}]) = \delta([\mathfrak{a}])$ and also that χ and δ are necessarily non-trivial characters of $\mathrm{Cl}(\mathcal{O})$. So it suffices to compute $\delta([\mathfrak{a}])$, which as we will see can be done very efficiently.

5.3.2 The Tate pairing on elliptic curves

We recall the main properties of the Tate pairing. For more details see [BSS05, IX]. The (reduced) *Tate pairing* T_m is defined as

$$T_m : E(\mathbb{F}_{q^k})[m] \times E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k}) \to \mu_m : (P, Q) \mapsto f_{m,P}(D)^{(q^k-1)/m},$$

where k is the embedding degree (i.e. the smallest extension degree k with the property that $\mu_m \subset \mathbb{F}_{q^k}^*$); the function $f_{m,P}$ a so-called Miller function, i.e. an $\mathbb{F}_{q^k}$-rational function with divisor $(f_{m,P}) = m(P) - m(\infty)$; D an $\mathbb{F}_{q^k}$-rational divisor equivalent to $(Q) - (\infty)$ coprime to the support of $(f_{m,P})$. If the Miller function $f_{m,P}$ is normalized, then the pairing can be computed for $Q \neq P$ as $T_m(P, Q) = f_{m,P}(Q)^{(q^k-1)/m}$.

The reduced Tate pairing T_m has the following properties:

1. Bilinearity: $T_m(P, Q_1 + Q_2) = T_m(P, Q_1)T_m(P, Q_2)$ and $T_m(P_1 + P_2, Q) = T_m(P_1, Q)T_m(P_2, Q)$.

2. Non-degeneracy: for all $P \in E(\mathbb{F}_{q^k})[m]$ with $P \neq \infty$, there exists a point $Q \in E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k})$ such that $T_m(P, Q) \neq 1$. Similarly, for all $Q \in E(\mathbb{F}_{q^k})$ with $Q \notin mE(\mathbb{F}_{q^k})$, there exists a $P \in E(\mathbb{F}_{q^k})[m]$ with $T_m(P, Q) \neq 1$.

3. Compatibility: let φ be an $\mathbb{F}_q$-rational isogeny, then

$$T_m(\varphi(P), \varphi(Q)) = T_m(P, Q)^{\deg(\varphi)}.$$

4. Galois invariance: let $\sigma \in \mathrm{Gal}(\overline{\mathbb{F}}_q/\mathbb{F}_q)$ then $T_m(\sigma(P), \sigma(Q)) = \sigma(T_m(P, Q))$.

5.4 Characters for ordinary curves

Recall that $\mathcal{Ell}_q(t)$ is the set of $\mathbb{F}_q$-isomorphism classes of elliptic curves over $\mathbb{F}_q$ with trace of Frobenius t. We always assume that $\mathcal{Ell}_q(t)$ is non-empty. The m-isogeny graph $G_{q,m}(t)$ (see Definition 2.5.3) has as vertices $\mathcal{Ell}_q(t)$. The edges are m-isogenies, up to equivalence as in Section 2.2.3. The connected components of $G_{q,m}(t)$ are isogeny volcanoes. For more details, see Section 2.5.

Let $\mathcal{O}$ be an imaginary quadratic order with discriminant $\Delta_\mathcal{O}$. Then the set $\mathcal{Ell}_q(\mathcal{O}, t)$ consists of elliptic curves over $\mathbb{F}_q$ with CM by $\mathcal{O}$, up to $\mathbb{F}_q$-equivalence. The class group $\mathrm{Cl}(\mathcal{O})$ acts regularly on $\mathcal{Ell}_q(\mathcal{O}, t)$.

We want to compute the assigned character χ_i with modulus $m_i \mid \Delta_\mathcal{O}$ an odd prime. Suppose that we are in the ordinary case. Then we have $m_i \nmid q$, since otherwise $m_i \mid \Delta_\mathcal{O} \mid t^2 - 4q$ would imply that $\gcd(t, q) \neq 1$, contradicting that E is ordinary. By extending the base field if needed, we can assume without loss of generality that $\mathrm{val}_m(\#E(\mathbb{F}_q)) \geq 1$.

Theorem 5.4.1. *Consider an m-isogeny volcano of height h of ordinary elliptic curves over a finite field $\mathbb{F}_q$, and let N be their (common) number of $\mathbb{F}_q$-rational points. Assume $v = \mathrm{val}_m(N) \geq 1$. Let E be a curve on level i.*

- *If v is odd and $0 \leq i \leq h$, or if v is even and E is a curve on level $0 \leq i \leq v/2$, then $E(\mathbb{F}_q)[m^\infty] \cong \mathbb{Z}/m^{v-i}\mathbb{Z} \times \mathbb{Z}/m^i\mathbb{Z}$.*

- *If v is even $v/2 \leq i \leq h$, then $E(\mathbb{F}_q)[m^\infty] \cong \mathbb{Z}/m^{v/2}\mathbb{Z} \times \mathbb{Z}/m^{v/2}\mathbb{Z}$.*

(Note that the latter range may be empty, i.e. one may have $h < v/2$.)

Proof. This is implicitly contained in [Len96]; see also [MMS+06, Cor. 1] for $m = 2$ and [MST+07, Thm. 3] for m odd. □

It is easy to verify whether a given curve $E/\mathbb{F}_q$ is on the floor of its volcano. Indeed, for λ random points $P \in E(\mathbb{F}_q)$ one simply tests whether $(N/m)P = \infty_E$. As soon as one point fails the test, we know that E is on the floor. If all points pass the test, we are on the floor with probability $1/m^\lambda$. Given such a verification method, a few random walks allow one to find a shortest path down to the floor, see e.g. the algorithm FindShortestPathToFloor in [Sut13b]. Note that this is considerably easier than navigating the volcano in a fully controlled way, see again [Sut13b] and the references therein.[1]

Once we are on the floor, with a curve E_0, the natural generalization of the case $v = 1$ is to compute the m-Tate pairing $T_m(P, Q)$ with $\mathrm{ord}(P) = m$ and Q any point with $\mathrm{ord}(Q) = m^v$ satisfying $m^{v-1}Q = P$. The following theorem applied to $n = 1$ shows that the m-Tate pairing is non-trivial and, for a fixed P, independent of the choice of Q. Note that we indeed have $m \mid q - 1$: if we write $N = \#E_0(\mathbb{F}_q)$, then $m \mid t^2 - 4q = (q-1)^2 - 2(q+1)N + N^2$.

Theorem 5.4.2. *Let $E_0/\mathbb{F}_q$ be an ordinary elliptic curve and let m be a prime number. Assume that $m^n | (q-1)$ for $n \geq 1$ and that*

$$E_0(\mathbb{F}_q)[m^\infty] \cong \mathbb{Z}/m^v\mathbb{Z}$$

for some $v \geq n$. Then for any P, Q with $\mathrm{ord}(P) = m^n$ and $\mathrm{ord}(Q) = m^v$, the reduced Tate pairing $T_{m^n}(P, Q)$ is a primitive m^n-th root of unity. For a fixed P, the pairing $T_{m^n}(P, \cdot)$ is constant for all Q with $\mathrm{ord}(Q) = m^v$ and $m^{v-n}Q = P$.

Proof. Assume for the sake of contradiction that $T_{m^n}(P, Q)$ is not a primitive m^n-th root of unity, then $T_{m^n}(P, Q) \in \mu_{m^{n-1}}$, and in particular

$$1 = T_{m^n}(P, Q)^{m^{n-1}} = T_{m^n}(m^{n-1}P, Q)\,.$$

Since P has order m^n, the point $m^{n-1}P$ is not the identity element ∞. Since Q generates $E_0(\mathbb{F}_q)[m^\infty]$, we conclude that $T_{m^n}(m^{n-1}P, \cdot)$ is degenerate on the whole of $E_0(\mathbb{F}_q)/m^nE_0(\mathbb{F}_q)$, which contradicts the non-degeneracy of the Tate pairing. Thus we conclude that $T_{m^n}(P, Q)$ is a primitive m^n-th root of unity (alternatively and more directly, this follows from the *perfectness* of the Tate pairing, see [Bru11]). The solutions to $m^{v-n}X = P$ are given by $Q + R$ with $\mathrm{ord}(R) | m^{v-n}$. But then $R \in m^nE_0(\mathbb{F}_q)$ and so $T_{m^n}(P, R) = 1$, which shows that $T_{m^n}(P, Q)$ is independent of the choice of Q. □

5.4.1 Computing the characters χ_i

Let χ be an assigned character χ_i associated with an odd prime divisor $m = m_i$ of $\Delta_\mathcal{O}$. As before, we let $\varphi : E \to E'$ denote the isogeny corresponding to $\mathfrak{a}$ of degree $\deg(\varphi) = \mathrm{N}(\mathfrak{a})$. Recall that the goal is to compute $\chi([\mathfrak{a}]) = \left(\frac{\mathrm{N}(\mathfrak{a})}{m}\right)$.

[1] In the context of this paper, it is worth highlighting the work of Ionica and Joux [IJ13] on this topic, who use the Tate pairing as an auxiliary tool for travelling through the volcano.

Since $\mathrm{End}(E) = \mathrm{End}(E')$ (c.f. Remark 2.5.2), the curves E and E' are on the same level of their respective m-isogeny volcanoes. By taking the same number of steps down from E and E' to the floor on these volcanoes, we end up with two respective elliptic curves E_0, E_0' in $\mathcal{Ell}_q(\mathcal{O}_0, t)$, where $\mathcal{O}_0 \subset \mathcal{O}$ is a suborder having discriminant $\Delta_{\mathcal{O}_0} = m^{2s}\Delta_{\mathcal{O}}$, with s the number of steps taken to reach the floor.

Since both curves E_0 and E_0' are now on the floor, we can choose non-trivial points $P \in E_0[m](\mathbb{F}_q)$ and $P' \in E_0'[m](\mathbb{F}_q)$, and points Q, Q' of order exactly m^v satisfying $m^{v-1}Q = P$ and $m^{v-1}Q' = P'$. As the class group $\mathrm{Cl}(\mathcal{O}_0)$ acts transitively on $\mathcal{Ell}_q(\mathcal{O}_0, t)$, there exists an invertible ideal $\mathfrak{b} \subset \mathcal{O}_0$ such that

$$E_0' = [\mathfrak{b}] \star E_0,$$

where by [Cox89, Cor. 7.17] it can be assumed that $\mathrm{N}(\mathfrak{b})$ is coprime with $\Delta_{\mathcal{O}_0}$, hence coprime with m. Let $\varphi_0 : E_0 \to E_0'$ denote the corresponding degree $\mathrm{N}(\mathfrak{b})$ isogeny. Then there exists a $k \in \{1, \ldots, m-1\}$ with $k\varphi_0(P) = P'$. Clearly, the point $k\varphi_0(Q)$ also has order m^v and satisfies $m^{v-1}X = P'$. From Theorem 5.4.2 and the compatibility of the Tate pairing, it then follows:

$$T_m(P', Q') = T_m(k\varphi_0(P), k\varphi_0(Q)) = T_m(P, Q)^{k^2 \deg(\varphi_0)},$$

and thus

$$\left(\frac{\mathrm{N}(\mathfrak{b})}{m}\right) = \left(\frac{\deg(\varphi_0)}{m}\right) = \left(\frac{\log_{T_m(P,Q)} T_m(P', Q')}{m}\right).$$

We now show that this in fact equals $\chi([\mathfrak{a}])$. Indeed, since $\mathrm{N}(\mathfrak{b})$ is coprime with $\Delta_{\mathcal{O}_0}$, from [Cox89, Prop. 7.20] we see that the ideal $\mathfrak{b}\mathcal{O} \subset \mathcal{O}$ is invertible and again has norm $\mathrm{N}(\mathfrak{b})$. From the second paragraph of the proof of [Sut13b, Lem. 6] we see that $E' = [\mathfrak{b}\mathcal{O}] \star E$, and because the action of $\mathrm{Cl}(\mathcal{O})$ on $\mathcal{Ell}_q(\mathcal{O}, t)$ is free we conclude that $[\mathfrak{b}\mathcal{O}] = [\mathfrak{a}]$. Summing up, we can compute

$$\chi([\mathfrak{a}]) = \chi([\mathfrak{b}\mathcal{O}]) = \left(\frac{\mathrm{N}(\mathfrak{b}\mathcal{O})}{m}\right) = \left(\frac{\mathrm{N}(\mathfrak{b})}{m}\right) = \left(\frac{\log_{T_m(P,Q)} T_m(P', Q')}{m}\right).$$

Note that, in particular, this outcome is independent of the choice of the walks to the floor of the isogeny volcano.

Remark 5.4.3. In the appendix we provide an alternative (but more complex) proof that shows it is not needed to walk all the way down to the floor. However, since the height of the volcano is about $\frac{1}{2}\,\mathrm{val}_m(t^2 - 4q)$ (see Theorem 2.5.4), the volcanoes cannot be very high (in the worst case a logarithmic number of levels), so walking to the floor of the volcano is efficient. Furthemore, for odd m, the probability of the volcano having height zero is roughly $1 - 1/m$.

5.4.2 Computing the characters δ, $\delta\epsilon$ and ϵ

For $\Delta_{\mathcal{O}} = -4n$, genus theory (Section 5.3.1) may give extra characters δ, ϵ or $\delta\epsilon$ depending on n mod 8. Recall that these characters are defined as

$$\delta : [\mathfrak{a}] \mapsto (-1)^{(\mathrm{N}(\mathfrak{a})-1)/2} \qquad \text{and} \qquad \epsilon : [\mathfrak{a}] \mapsto (-1)^{(\mathrm{N}(\mathfrak{a})^2-1)/8},$$

where the ideal $\mathfrak{a}$ is chosen to have odd norm. Determining the value of δ is equivalent to computing $\mathrm{N}(\mathfrak{a})$ mod 4. If both δ and ϵ exist (i.e. $n \equiv 0 \bmod 8$), determining both character values is equivalent to computing $\mathrm{N}(\mathfrak{a})$ mod 8.

For $m = 2$, the previous approach using Theorem 5.4.2 with $n = 1$ remains valid, but only determines $\mathrm{N}(\mathfrak{a})$ mod 2, which is known beforehand since the norm is odd. The solution is to use a 4-pairing (i.e. $n = 2$) to derive δ and an 8-pairing (i.e. $n = 3$) in the case both δ and ϵ exist.

Character δ. The character δ exists when $n \equiv 0, 1, 4, 5 \bmod 8$. By taking a field extension if needed, we can assume without loss of generality that $4 \mid (q-1)$ and $v = \mathrm{val}_2(\#E(\mathbb{F}_q)) \geq 2$. As before, by walking down the volcano we reach a curve E_0 on the floor (and similarly E_0') satisfying $E_0(\mathbb{F}_q)[2^\infty] = \mathbb{Z}/2^v\mathbb{Z}$. We can use Theorem 5.4.2 for $m = 2$ and $n = 2$ along with the compatibility of the Tate pairing: if $\mathfrak{b}$ is an ideal connecting E_0 and E_0', we can compute the exact value

$$\mathrm{N}(\mathfrak{b}) \bmod 4 = \log_{T_4(P,Q)} T_4(P', Q') \tag{5.2}$$

for appropriately chosen points $P, Q \in E_0(\mathbb{F}_q)[2^\infty]$ and $P', Q' \in E_0'(\mathbb{F}_q)[2^\infty]$. Indeed, recall that the points P' and Q' are only determined by P and Q up to a scalar $k \in (\mathbb{Z}/(4))^*$, i.e. $k \equiv 1, 3 \bmod 4$, and so $k^2 \equiv 1 \bmod 4$.

As before, we can show that $[\mathfrak{b}\mathcal{O}] = [\mathfrak{a}]$, where we can assume $\mathrm{N}(\mathfrak{b}\mathcal{O}) = \mathrm{N}(\mathfrak{b})$, so we find that

$$\delta([\mathfrak{a}]) = \delta([\mathfrak{b}\mathcal{O}]) = (-1)^{(\mathrm{N}(\mathfrak{b}\mathcal{O})-1)/2} = (-1)^{(\log_{T_4(P,Q)} T_4(P',Q')-1)/2},$$

or, equivalently, we find that $\mathrm{N}(\mathfrak{a})$ mod 4 equals (5.2).

Characters $\delta\epsilon$ and ϵ. Recall that the character $\delta\epsilon$ exists when $n \equiv 0, 2 \bmod 8$ and the character ϵ exists when $n \equiv 0, 6 \bmod 8$. Again, by taking a field extension if needed, we can assume without loss of generality that $v = \mathrm{val}_2(\#E(\mathbb{F}_q)) \geq 3$ and that $8 \mid (q-1)$. Notice that, if δ and ϵ do not exist simultaneously, then we are necessarily on the surface of the 2-volcano, hence it takes at least one step to go to curves E_0 and E_0' on the floor. During this step the discriminant becomes multiplied by a factor of 4. Hence, on the floor, we are certain that both characters exist.

Applying Theorem 5.4.2 for $m = 2$ and $n = 3$ together with the compatibility of the Tate pairing, and using the fact that for k odd we have $k^2 \equiv 1 \bmod 8$, we know that the norm of an ideal $\mathfrak{b}$ connecting E_0 and E_0' satisfies

$$\mathrm{N}(\mathfrak{b}) \bmod 8 = \log_{T_8(P,Q)} T_8(P', Q'), \tag{5.3}$$

for appropriately chosen points $P, Q \in E_0(\mathbb{F}_q)[2^\infty]$ and $P', Q' \in E_0'(\mathbb{F}_q)[2^\infty]$. The same reasoning as before gives $[\mathfrak{b}\mathcal{O}] = [\mathfrak{a}]$, where $\mathrm{N}(\mathfrak{b}\mathcal{O}) = \mathrm{N}(\mathfrak{b})$, hence we find

$$\epsilon([\mathfrak{a}]) = \epsilon([\mathfrak{b}\mathcal{O}]) = (-1)^{(\mathrm{N}(\mathfrak{b}\mathcal{O})^2-1)/8} = (-1)^{((\log_{T_8(P,Q)} T_8(P',Q'))^2-1)/8},$$

and similarly for $\delta\epsilon$.

We stress that, in general, we cannot conclude that $\mathrm{N}(\mathfrak{a}) \bmod 8$ equals (5.3). E.g., if $n \equiv 6 \bmod 8$, in the presence of ϵ but in the absence of δ, an ideal class containing ideals of norm $1 \bmod 8$ will also contain ideals of norm $7 \bmod 8$. It is during the first step down the volcano that the congruence classes become separated.

5.5 Characters for supersingular curves

We now turn our attention to supersingular elliptic curves over prime fields $\mathbb{F}_p$ with $p > 3$. Recall that any such curve $E/\mathbb{F}_p$ has exactly $p+1$ rational points and its Frobenius satisfies $\pi^2 + p = 0$, therefore $\mathcal{O} = \mathrm{End}_{\mathbb{F}_p}(E)$ has discriminant

$$\Delta_{\mathcal{O}} = \begin{cases} -4p & \text{if } p \equiv 1 \bmod 4, \\ -p \text{ or } -4p & \text{if } p \equiv 3 \bmod 4. \end{cases}$$

From genus theory, we see that $\mathrm{Cl}(\mathcal{O})$ has non-trivial 2-torsion only in the former case. So we will restrict our attention to $p \equiv 1 \bmod 4$, in which case $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$. There are two assigned characters: the Legendre character associated with p, and δ. From the character relation (5.1) (see also Example 5.3.2), we see that these coincide on $\mathrm{Cl}(\mathcal{O})$, therefore it suffices to compute δ. As supersingular elliptic curves over $\mathbb{F}_{p^2}$ no longer have a volcano structure (see Section 2.6), we cannot apply the strategy of "extending the base field and going down the volcano".

Instead, we can compute δ directly on the input curves, i.e. not involving vertical isogenies. This is handled by the following theorem, which can be used to compute δ in many ordinary cases, too. The proof is entirely self-contained, although its flavour is similar to that of Section 5.4.

Theorem 5.5.1. *Let $q \equiv 1 \bmod 4$ be a prime power and let $E, E'/\mathbb{F}_q$ be elliptic curves with endomorphism ring $\mathcal{O}$ and trace $t \equiv 0 \bmod 4$, connected by an ideal class $[\mathfrak{a}] \in \mathrm{Cl}(\mathcal{O})$. Then δ is an assigned character of $\mathcal{O}$, and if we write*

$$E : y^2 = x^3 + ax^2 + bx \quad \textit{resp.} \quad E' : y^2 = x^3 + a'x^2 + b'x \tag{5.4}$$

then $\delta([\mathfrak{a}]) = (b'/b)^{(q-1)/4}$.

Proof. As $t \equiv 0 \bmod 4$, we have $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q) = q + 1 - t \equiv 2 \bmod 4$, and therefore both curves contain a unique rational point of order 2. When positioned at $(0,0)$, we indeed obtain models of the form Equation (5.4). Notice that $b^{(q-1)/4}$

does not depend on the specific choice of such a model: it is easy to check that the only freedom left is scaling a by u^2 and b by u^4 for some $u \in \mathbb{F}_q^*$. Of course, the same remark applies to $b'^{(q-1)/4}$.

On E, the points (x_0, y_0) doubling to $P = (0, 0)$ satisfy the condition

$$\frac{3x_0^2 + 2ax_0 + b}{2y_0} = \frac{y_0}{x_0},$$

which can be rewritten as $x_0(x_0^2 - b) = 0$. Therefore these points are

$$\left(\sqrt{b}, \pm\sqrt{b(a + 2\sqrt{b})}\right) \quad \text{and} \quad \left(-\sqrt{b}, \pm\sqrt{b(a - 2\sqrt{b})}\right), \tag{5.5}$$

from which we see that b is a non-square. Indeed, if we would have $\sqrt{b} \in \mathbb{F}_q$, then one of $a \pm 2\sqrt{b}$ would be a square in $\mathbb{F}_q$ because their product $a^2 - 4b$ is not (since there is only one $\mathbb{F}_q$-rational point of order 2). This would imply the existence of an $\mathbb{F}_q$-rational point of order 4, contradicting $\#E(\mathbb{F}_q) \equiv 2 \bmod 4$. The same reasoning shows that b' is a non-square.

Pick a representative $\mathfrak{a}$ of $[\mathfrak{a}]$ of norm coprime to $2q$. It suffices to show that

$$(-b')^{(q-1)/4} = \left((-b)^{(q-1)/4}\right)^{\mathrm{N}(\mathfrak{a})} \tag{5.6}$$

(the reason for including the minus signs, which cancel out, will become apparent soon). Indeed, both sides are primitive 4th roots of unity, whose ratio is either 1 or -1 depending on whether $\mathrm{N}(\mathfrak{a}) \equiv 1 \bmod 4$ or $\mathrm{N}(\mathfrak{a}) \equiv 3 \bmod 4$, as wanted.

Let $\varphi : E \to E'$ be the isogeny corresponding to $\mathfrak{a}$. Note that $\varphi(P) = P'$ because φ is defined over $\mathbb{F}_q$. From Equation (5.5), using that b is a non-square, we see that we can characterize $-b$ as $x(Q) \cdot x(\pi_q(Q))$, where Q denotes any of the four halves of P. Similarly, $-b'$ equals $x(Q') \cdot x(\pi_q(Q'))$, with Q' any of the four halves of $P' = (0, 0) \in E'$. In particular, since $\varphi(Q)$ is a half of $\varphi(P) = P'$, we have $-b' = x(\varphi(Q)) \cdot x(\pi_q(\varphi(Q)))$.

Remark 5.5.2. Observe that x is the normalized Miller function $f_{2,P}$, hence

$$(-b)^{(q-1)/4} = (x(Q) \cdot x(\pi_q(Q)))^{(q-1)/4} = \left(f_{2,P}(Q)^{1+q}\right)^{(q-1)/4} = f_{2,P}(Q)^{\frac{q^2-1}{4}},$$

and similarly for $(-b')^{(q-1)/4}$, so proving Equation (5.6) amounts to proving a compatibility rule for a non-fully reduced 2-Tate pairing.

Denote by $\pm K_1, \pm K_2, \ldots, \pm K_{(\mathrm{N}(\mathfrak{a})-1)/2}$ the non-trivial points in $\ker \varphi$, say with x-coordinates $x_1, x_2, \ldots, x_{(\mathrm{N}(\mathfrak{a})-1)/2} \in \overline{\mathbb{F}}_q$. Besides P itself, the points mapping to P' are $P \pm K_1, P \pm K_2, \ldots, P \pm K_{(\mathrm{N}(\mathfrak{a})-1)/2}$, and an easy calculation shows that the x-coordinates of these points are $b/x_1, b/x_2, \ldots, b/x_{(\mathrm{N}(\mathfrak{a})-1)/2}$. This implies that the function

$$x \left(\prod_{i=1}^{(\mathrm{N}(\mathfrak{a})-1)/2} \frac{x - \frac{b}{x_i}}{x - x_i} \right)^2$$

viewed on E has the same divisor as $x \circ \varphi$, therefore both functions are proportional. To determine the constant involved, we can assume that our curve E' is obtained through an application of Vélu's formulae [Vél71], composed with a translation along the x-axis that positions P' at $(0,0)$. From [DF10, Rmk. 8.1] we then see that $x \circ \varphi = g(x)/h(x)$ with

$$h(x) = \prod_{i=1}^{(\mathrm{N}(\mathfrak{a})-1)/2} (x - x_i)^2$$

and with $g(x)$ a polynomial of degree $\mathrm{N}(\mathfrak{a})$ with leading coefficient

$$\mathrm{N}(\mathfrak{a}) - 3(\mathrm{N}(\mathfrak{a}) - 1) + 2(\mathrm{N}(\mathfrak{a}) - 1) = 1.$$

So the involved constant is just 1, i.e. equality holds. We then compute

$$\begin{aligned} -b' &= x(\varphi(Q)) \cdot x(\pi_q(\varphi(Q))) \\ &= (x \circ \varphi)(Q) \cdot (x \circ \varphi)(\pi_q(Q)) \\ &= -b \left(\prod_{i=1}^{(\mathrm{N}(\mathfrak{a})-1)/2} \frac{(\sqrt{b} - \frac{b}{x_i})(-\sqrt{b} - \frac{b}{x_i})}{(\sqrt{b} - x_i)(-\sqrt{b} - x_i)} \right)^2 \\ &= \frac{(-b)^{\mathrm{N}(\mathfrak{a})}}{\left(\prod_{i=1}^{(\mathrm{N}(\mathfrak{a})-1)/2} x_i \right)^4}, \end{aligned}$$

and Equation (5.6) follows by raising both sides to the power $(q-1)/4$. □

5.6 Impact on DDH and countermeasures

5.6.1 Impact on DDH for class group actions

It is clear that any non-trivial character χ (or δ, ϵ, $\delta\epsilon$) can be used to probabilistically determine whether a sample $(E^{(1)} = [\mathfrak{a}] \star E, E^{(2)} = [\mathfrak{b}] \star E, E^{(3)})$ is a true Diffie-Hellman sample, i.e. whether $E^{(3)} = [\mathfrak{a} \cdot \mathfrak{b}] \star E$ or not. For instance, one could compute $\chi([\mathfrak{a}])$ in two different ways, namely as $\chi(E, E^{(1)})$ and compare with $\chi(E^{(2)}, E^{(3)})$. Similarly, one could compute $\chi([\mathfrak{b}])$ in two ways, as $\chi(E, E^{(2)})$ as well as $\chi(E^{(1)}, E^{(3)})$. If the sample is not a true Diffie–Hellman sample this will be detected with probability $1/2$. In many cases we have more than one character available, so if we assume that $s < \mu$ linearly independent characters are computable (see below for the complexity of a single character), this probability increases to $1 - 1/2^s$.

Supersingular curves. For supersingular curves over $\mathbb{F}_p$ with $p \equiv 1 \bmod 4$, the character δ exists and is always non-trivial (see Example 5.3.2). As shown in Section 5.5, computing this character requires computing a 2-torsion point, one inversion and one exponentiation in $\mathbb{F}_p$, so in this case, DDH can be broken in time $O(\log p \cdot M_p)$ with M_p the cost of a multiplication in $\mathbb{F}_p$.

Ordinary curves. For ordinary curves, we will order the characters (if they exist) according to complexity: δ, ϵ, $\delta\epsilon$, χ_{m_i} for $i = 1, \ldots, r$. From genus theory, it follows that at most one of the μ characters is trivial (since $\#\operatorname{Cl}(\mathcal{O})[2] = 2^{\mu-1}$), so if the easiest-to-compute character is trivial, we immediately conclude that the second easiest to compute character is non-trivial. To determine the complexity, assume that m is an odd prime divisor of $\Delta_{\mathcal{O}}$. To apply our attack, we first find the smallest extension $\mathbb{F}_{q^k}$ such that $\operatorname{val}_m(\#E(\mathbb{F}_{q^k})) \geq 1$. Since $m \mid \Delta_{\mathcal{O}} \mid t^2 - 4q$, the matrix of Frobenius on $E[m]$ is of the form

$$\begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix},$$

with $\lambda^2 \equiv q \bmod m$. In both cases, for $k = \operatorname{ord}(\lambda) \in (\mathbb{Z}/m\mathbb{Z})^*$, we conclude that $\operatorname{val}_m(\#E(\mathbb{F}_{q^k})) \geq 1$. Furthermore, since the determinant of the k-th power equals $q^k \equiv \lambda^{2k} \equiv 1 \bmod m$, we conclude that $\mu_m \subset \mathbb{F}_{q^k}$ and thus the m-Tate pairing is defined over $\mathbb{F}_{q^k}$. We see that in the worst case, we have $k = m - 1$. Computing the m-Tate pairing requires $O(\log m \cdot M_{q^k})$ which is $O(m^{1+\varepsilon} \cdot M_q)$ assuming fast polynomial arithmetic and using $k < m$. The cost of walking down the volcano [Sut13b] over $\mathbb{F}_{q^k}$ in the worst case is given by $O(h \cdot (m^{3+\varepsilon} \cdot \log q) \cdot M_q)$ assuming fast polynomial arithmetic (and $k < m - 1$), with h a bound on the height of the volcano. Once we reached the floor of the volcano, we need to solve the equation $m^{v-1}Q = P$, with P an m-torsion point, and $v = \operatorname{val}_m(\#E(\mathbb{F}_{q^k}))$. This can be computed deterministically using division polynomials, or probabilistically as follows: generate a point Q_1 of order m^v, and compute $P_1 = m^{v-1}Q_1$. Since we are on the floor, $E(\mathbb{F}_q)[m]$ is cyclic, so there exists a k with $P = kP_1$. Then $Q = kQ_1$ is a solution. This randomized approach can be done in expected time $O(m^{3+\varepsilon} \cdot \log q \cdot M_q)$.

As remarked before, we note that in the majority of cases (with probability roughly $1 - 1/m$), the height of the m-volcano is zero and the complexity of the attack is solely determined by the computation of the Tate pairing.

Computing the exact coset modulo $\operatorname{Cl}(\mathcal{O})^2$. Genus theory shows that the intersection of the kernels of the assigned characters is exactly $\operatorname{Cl}(\mathcal{O})^2$. Thanks to the class group relation (5.1), we are allowed to omit one character. If all remaining characters have a manageable complexity then, given two elliptic curves E and $[\mathfrak{a}] \star E$, this allows to determine completely the coset of $\operatorname{Cl}(\mathcal{O})^2$ inside $\operatorname{Cl}(\mathcal{O})$ to which the connecting ideal class $[\mathfrak{a}]$ belongs. In general, we can determine

which coset of $C \supset \mathrm{Cl}(\mathcal{O})^2$ contains $[\mathfrak{a}]$, where C denotes the intersection of the kernels of the characters whose computation is feasible.

As an application, one can reduce the vectorization problem for $\mathrm{Cl}(\mathcal{O})$ to that for C. Indeed, one simply chooses an ideal class $[\mathfrak{b}]$ in the same coset as $[\mathfrak{a}]$, so that $[\mathfrak{a} \cdot \mathfrak{b}] \in C$, and one considers the vectorization problem associated with E and $[\mathfrak{a}\cdot\mathfrak{b}]\star E = [\mathfrak{b}]\star([\mathfrak{a}]\star E)$. After finding $[\mathfrak{a}\cdot\mathfrak{b}]$, one recovers $[\mathfrak{a}]$ as $[\mathfrak{b}]^{-1}\cdot[\mathfrak{a}\cdot\mathfrak{b}]$. In the optimal case where $C = \mathrm{Cl}(\mathcal{O})^2$, this reduces the group size by a factor $2^{\mu-1}$. We emphasize that this reduction is classical; quantumly, such a reduction follows from earlier work due to Friedl, Ivanyos, Magniez, Santha and Sen [FIM+14], see [CDEL21, §2] for a more detailed discussion.

5.6.2 Implementation results

We implemented our attack in the Magma computer algebra system [BCP97] and the resulting code is given in the GitHub repository [CSV22b].

The main functions are `ComputeEvenCharacters`, `ComputeOddCharacter` and `ComputeSupersingularDelta`. We also use a very simple randomized method to walk to the floor of the volcano in the function `ToFloor`. A more efficient approach can be found in [Sut13b].

To illustrate the code, we apply it to an example from [DKS18, Section 4], which we discussed in Example 4.2.2. In particular, let $p = 7\left(\prod_{\substack{2\leq \ell \leq 380 \\ \ell \text{ prime}}} \ell\right) - 1$ and consider the elliptic curve $E : y^2 = x^3 + Ax^2 + x$ with

$$\begin{aligned} A =& 10861338504649280383859950140772947007703646408372 8 \\ & 31934324660566888732797778932142488253565145603672 5 \\ & 91944602210571423767689240032829444439469242521864171\,, \end{aligned}$$

then $\mathrm{End}(E)$ is the maximal order and E is on the surface of a volcano of height 2. The discriminant is of the form $-4n$ with $n \equiv 2 \bmod 8$, so we will be able to compute the character $\delta\epsilon$.

The code first computes a random isogeny of degree 523 (easy to compute since it is rational), to obtain the "challenge" $E' = [\mathfrak{a}] \star E$. After going to a degree 2 extension, it then descends the volcano to the floor, and on the floor, it computes both δ as well as ϵ, from which it derives that $\delta\epsilon(E, E') = 1$, which is consistent with the fact that $\delta\epsilon([\mathfrak{a}]) = \delta\epsilon(523) = 1$.

5.6.3 Countermeasures

Since the attack crucially relies on the existence of 2-torsion in $\mathrm{Cl}(\mathcal{O})$, the simplest countermeasure is to restrict to a setting where $\mathrm{Cl}(\mathcal{O})[2]$ is trivial, e.g. supersingular elliptic curves over $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$. This corresponds precisely to the CSIDH setting [CLM+18], so our attack does not impact CSIDH.

Another standard approach is to work with co-factors: since all characters become trivial on $\mathrm{Cl}(\mathcal{O})^2$ we can simply restrict to elements which are squares, i.e. in the Diffie-Hellman protocol one would sample $[\mathfrak{a}]^2$ and $[\mathfrak{b}]^2$.

Warning. We advise to be much more cautious than simply squaring. Genus theory gives the structure of $\mathrm{Cl}(\mathcal{O})[2]$, but one can also derive the structure of the 2-Sylow subgroup $\mathrm{Cl}(\mathcal{O})[2^\infty]$ using an algorithm going back to Gauss and analyzed in detail by Bosma and Stevenhagen [BS96]. Although our attack is currently not refined enough to also exploit this extra information, we expect that a generalization of our attack will be able to do so. As such, instead of simply squaring, we advise to use as co-factor an upper bound on the exponent of the 2-Sylow subgroup.

5.7 Conclusion

We showed how the characters defined by genus theory for the class group $\mathrm{Cl}(\mathcal{O})$ can be computed from the group action of $\mathrm{Cl}(\mathcal{O})$ on $\mathcal{E}\ell\ell_q(\mathcal{O}, t)$, knowing only the equations of two elliptic curves E and $E' = [\mathfrak{a}] \star E$, for an unknown ideal class $[\mathfrak{a}]$. For a character χ associated to the prime divisor $m \mid \Delta_\mathcal{O}$, the complexity is exponential in the size of m, and it is thus efficiently computable only for smallish m. However, since only one such character is required to break DDH for class group actions, we conclude that for a subset of density 1 of ordinary curves, and for all supersingular curves over $\mathbb{F}_p$ with $p \equiv 1 \bmod 4$, DDH (without appropriate countermeasures) is broken. Note that CSIDH [CLM+18] is not affected as it relies on supersingular elliptic curves over $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$. We have also shown that the main ideas behind these results can be used to tackle related questions on abelian varieties of arbitrary dimension. Our current results however, are only the first tiny steps towards a proper and full generalization, which is the subject of future research.

The main, quite surprising, insight of this paper is that the structure of the class group $\mathrm{Cl}(\mathcal{O})$ does actually matter, and cannot be assumed to be fully hidden when represented as $\mathcal{E}\ell\ell_q(\mathcal{O}, t)$ under the class group action $\star$, not even classically. Philosophically, one might argue that this is inherently caused by the fact that the structure of $\mathrm{Cl}(\mathcal{O})[2]$ is easily computable. As such, it is imperative to analyze two cases which also give partial information about the class group $\mathrm{Cl}(\mathcal{O})$:

- As already mentioned in Section 7.9, the algorithm by Bosma and Stevenhagen [BS96] determines the structure of the 2-Sylow group $\mathrm{Cl}(\mathcal{O})[2^\infty]$. Can our attack be extended to take this extra information into account?

- The class number formula expressing the class number of a suborder $\mathcal{O}$ in terms of the class number of the maximal order $\mathcal{O}_K$ and the conductor f (see Equation (4.4)) can be used to derive certain prime factors of $h(\mathcal{O})$ without

knowing $h(\mathcal{O}_K)$. For instance, in the case of CSIDH with $p \equiv 3 \bmod 8$ where $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$, the above formula implies that $h(\mathcal{O})$ is divisible by 3. Can an attack be devised where such factors are exploited?

Finally, we note that in most settings the exact structure of $\mathrm{Cl}(\mathcal{O})$ is unknown, so the usual approach of restricting to a large prime order subgroup does not apply. As a precaution, we therefore advise to work with supersingular curves $E/\mathbb{F}_p$ with $p \equiv 3 \bmod 4$, such that $\mathrm{End}(E) = \mathcal{O}_K$, i.e. restrict to curves on the surface as was done in the recent CSURF construction [CD20].

5.8 Follow-up work

This attack has since been generalized to several new situations.

Higher dimensions. The extended version [CSV22a] shows that the central idea of the attack naturally generalizes to principally polarized abelian varieties of any dimension. However, it is unclear what the correct generalization of the DDH problem should be in the higher dimensions. [CSV22a] shows that one can still use the reduced Tate pairing to determine the length of the chain of secret Richelot isogenies between Jacobians of genus-2 curves (with appropriate torsion).

Oriented curves. The recent results of Castryck, Houben, Vercauteren, and Wesolowski [CHVW22] showed that the attack extends to supersingular elliptic curves *oriented* by an imaginary quadratic order $\mathcal{O}$.

Their approach uses the Weil pairing (which is always self-trivial), and so instead pair the point P with its image under a *distortion map*, i.e. an endomorphism σ such that $\sigma(P)$ is not a multiple of P. This approach works for almost all imaginary quadratic orders with *even* class number.

Breaking weak instances of CDH Further work by Castryck, Houben, Merz, Mula, van Buuren, and Vercauteren [CHM+23] use generalizations of the Weil and Tate pairings to identify weak instances of the CDH assumption. In the best case, their attack is polynomial time. Their approach also generalizes and simplifies the alternative approach in Section 5.9.

5.9 Appendix: Not walking to the floor

In Section 5.4, our approach to computing $\chi(E, E')$ was to take an arbitrary walk to the floor of the respective m-isogeny volcanoes of E and E'. In fact, one can stop walking down as soon as one reaches a level where the m^∞-torsion is

sufficiently unbalanced. For this, we modify Theorem 5.4.2 (for $n = 1$), which is likely to admit further generalizations. Here, we should mention recent follow-up work [CHVW22], which shows that one can avoid walking to the floor by resorting to the Weil pairing instead of the Tate pairing (although this approach may come with extra costs [CHVW22, §4]).

Theorem 5.9.1. *Let $E/\mathbb{F}_q$ be an ordinary elliptic curve and let $m \mid q-1$ be a prime. Assume that E is not located on the crater of its m-volcano and that*

$$E(\mathbb{F}_q)[m^\infty] \cong \mathbb{Z}/m^r\mathbb{Z} \times \mathbb{Z}/m^s\mathbb{Z}$$

for some $r > s+1$. Let $P \in E(\mathbb{F}_q)[m] \setminus \{\infty\}$ be such that there exists $Q \in E(\mathbb{F}_q)$ for which $m^{r-1}Q = P$. Then the reduced Tate pairing

$$T_m(P, \cdot) : E(\mathbb{F}_q)/mE(\mathbb{F}_q) \to \mu_m : X \mapsto T_m(P, X) \tag{5.7}$$

is trivial if and only if X belongs to $E[m^s] \bmod mE(\mathbb{F}_q)$. In particular, $T_m(P, Q)$ is a primitive m-th root of unity which, for a fixed P, does not depend on the choice of Q.

Proof. The assumption $m \mid (q-1)$ implies that $\mu_m \subset \mathbb{F}_q$. As explained in [BSS05, IX.7.1], the kernel of $T_m(P, \cdot)$ is a codimension 1 subspace of $E(\mathbb{F}_q)/mE(\mathbb{F}_q)$, when viewed as a vector space over $\mathbb{F}_m$. Therefore it suffices to prove that $T_m(P, \cdot)$ is trivial on $E[m^s] \bmod mE(\mathbb{F}_q)$, because the latter space indeed has codimension 1. More precisely, it has dimension 0 if $s = 0$ and dimension 1 if $s \geq 1$.

Since we are not on the crater, we know from Theorem 5.4.1 that there exists an elliptic curve $E'/\mathbb{F}_q$ and an $\mathbb{F}_q$-rational m-isogeny $\varphi : E' \to E$ such that $E'(\mathbb{F}_q)[m^\infty] \cong \mathbb{Z}/(m^{r-1}) \times \mathbb{Z}/(m^{s+1})$. We note:

- we have $E[m^s] \subset \varphi(E'[m^{s+1}]) \subset \varphi(E'(\mathbb{F}_q))$, hence each $X \in E[m^s]$ can be written as $\varphi(X')$ for some $X' \in E'(\mathbb{F}_q)$.
- The kernel of the dual isogeny $\hat{\varphi} : E \to E'$ equals $\langle P \rangle$; otherwise E' would admit $\mathbb{F}_q$-rational m^r-torsion. So P is the image of a $P' \in E'[m] \subset E'(\mathbb{F}_q)$.

We conclude that

$$T_m(P, X) = T_m(\varphi(P'), \varphi(X')) = T_m(P', X')^{\deg(\varphi)} = T_m(P', X')^m = 1,$$

as wanted. □

Chapter 6

CTIDH: constant-time CSIDH

This chapter is based on the paper *CTIDH: faster constant-time CSIDH* [BBC+21], co-authored with Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, Tung Chou, Tanja Lange, Michael Meyer, and Benjamin Smith (CHES 2021).

The introduction is new, and the background section is omitted as the relevant details have been treated in Chapters 2 and 4. Two original sections [BBC+21, Sec. 7.4 and 8] have been shortened and the appendices have been omitted. The remaining sections have been edited for style and typographical consistency.

6.1 Contributions of CTIDH

This paper introduces a new key space for CSIDH, and a new constant-time algorithm to evaluate the CSIDH group action. The new key space is not useful by itself—it slows down previous constant-time algorithms—and similarly the new constant-time algorithm is not useful for previous key spaces; but there is a synergy between the key space and the algorithm, and using both of them together produces a large improvement in the performance of constant-time CSIDH.

Assume that one is using 6 primes and allows at most 6 isogeny computations, with each $\mathfrak{l}_i$ exponent being nonnegative. The standard key space (Section 4.3.4) chooses $(e_1, e_2, \ldots, e_6) \in \mathcal{K}_1 = \{0, 1\}^6$, giving $2^6 = 64$ keys.

The new key space groups the primes into *batches* of three primes, and allows up to 3 isogenies per batch. More concereteley, choose $(e_1, e_2, \ldots, e_6) \in [0, 3]^6$ with the condition that $e_1 + e_2 + e_3 \leq 3$ and $e_4 + e_5 + e_6 \leq 3$, giving $20^2 = 400$ keys.

Putting each prime into its own batch is the standard key space. The opposite extreme of all primes in 1 giant batch is also not new: putting a bound on the 1-norm of the key vector is the best tradeoff between the number of isogenies and the size of the key space [NOTT23]. In the above example, 1 giant batch of six primes under the condition $e_1 + \ldots e_6 \leq 6$ gives 924 keys for 6 isogenies.

However, we also want to make the implementation constant-time. Because the key spaces above have different sizes, we will estimate the cost as the number of isogenies per bit of the key space. One giant batch would require 6 isogenies per degree, that is, 36 isogenies per less than 10 bits. The intermediate example would require 18 isogenies for 8.6 bits of key space. The standard key space is the clear winner in this (small) example, requiring 6 isogenies per 6 keys (requiring larger-sized key spaces reduces the benefits).

In CTIDH, we give a new algorithm for computing the isogeny steps for all primes in the batch in *constant time*, using the same sequence of operations. In the example above, CTIDH only computes 3 isogenies per batch, and a total of 6 isogenies per 8.64 bits of key space. We achieve this by generalizing the Matryoshka-doll structure of Vélu's formulas [BLMP19] to $\sqrt{\text{élu}}$ formulas from [BDLS20]. In the process, we also resolve the issue of isogeny computation failing with probability dependent on the degree (because of sampling torsion).

Moreover, we optimize the batches such that the new constant-time isogeny evaluation does not incur such a massive overhead as in [BLMP19]. For comparability we report CSIDH-512 speeds, setting records in multiplications and in cycles for complete constant-time software. For applications that want higher security levels (see Remark 4.1.12), our software also supports larger sizes.

We formalize the concept of *rounds* in computing isogenies as *atomic blocks* (Section 6.3), and discuss how to compute them in constant time (Section 6.4). The question of optimally selecting batches and bounds for a fixed key space size is discussed in Section 6.5. Our software `high-ctidh` is discussed in Section 6.6 and its performance is compared to other implementations in Section 6.7.

6.2 Batching and key spaces

The main conceptual novelty in CTIDH is the organization of primes and isogenies in batches. We generalize the example from the introduction to define a new batch-oriented key space.

Batching primes. In CTIDH, the sequence of primes $(\ell_1, \ldots, \ell_n)$ is partitioned into a series of *batches*: subsequences of consecutive primes. Let $0 < B \leq n$ be the number of batches; we represent the sequence of the batch sizes by an integer vector $N = (N_1, \ldots, N_B) \in \mathbb{Z}_{>0}^B$ with $\sum_{i=1}^B N_i = n$. We relabel the primes in batches as: $(\ell_{1,1}, \ldots, \ell_{1,N_1}) := (\ell_1, \ldots, \ell_{N_1})$, $(\ell_{2,1}, \ldots, \ell_{2,N_2}) := (\ell_{N_1+1}, \ldots, \ell_{N_1+N_2}), \ldots,$ $(\ell_{B,1}, \ldots, \ell_{B,N_B}) := (\ell_{n-N_B+1}, \ldots, \ell_n)$. If $\ell_{i,j}$ corresponds to ℓ_k, then we also write $\mathfrak{l}_{i,j}$ for $\mathfrak{l}_k$ and $e_{i,j}$ for e_k.

Example 6.2.1. Say we have $n = 6$ primes, $(\ell_1, \ldots, \ell_6)$. If we take $B = 3$ and $N = (3, 2, 1)$, then $(\ell_{1,1}, \ell_{1,2}, \ell_{1,3}) = (\ell_1, \ell_2, \ell_3)$, $(\ell_{2,1}, \ell_{2,2}) = (\ell_4, \ell_5)$, and finally $(\ell_{3,1}) = (\ell_6)$.

Batching isogenies. Consider the i-th batch of primes $(\ell_{i,1}, \ldots, \ell_{i,N_i})$. Rather than setting a bound $m_{i,j} \geq |e_{i,j}|$ for the number of $\ell_{i,j}$-isogenies for each individual index $1 \leq j \leq N_i$, we set a bound $m_i \geq \sum_{j=1}^{N_i} |e_{i,j}|$ and compute m_i isogenies from the batch $(\ell_{i,1}, \ldots, \ell_{i,N_i})$. This looks analogous to the use of dummy operations in the previous constant-time algorithms, but it gives a larger key space per isogeny computed because of the ambiguity between the degrees in a batch. Moreover, we will show in Section 6.4.2 that it is possible to evaluate any isogeny within one batch in the same constant time.

Extreme batching choices correspond to well-known approaches to the group action evaluation: one prime per batch ($B = n$ and $N = (1, \ldots, 1)$) was considered in [CLM+18]; one n-prime batch ($B = 1$ and $N = (n)$) is considered in [BLMP19] for the quantum oracle evaluation and in [NOTT23] as a speedup for CSIDH. The intermediate cases are new, and, as we will show, faster.

The new key space. For $N \in \mathbb{Z}_{>0}^B$ and $m \in \mathbb{Z}_{\geq 0}^B$, we define

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

We may see $\mathcal{K}_{N,m}$ as a generalization of $\mathcal{K}_m$.

Lemma 6.2.2. *We have*

$$\#\mathcal{K}_{N,m} = \prod_{i=1}^{B} \Phi(N_i, m_i), \quad \textit{where} \quad \Phi(x, y) = \sum_{k=0}^{\min\{x,y\}} \binom{x}{k} 2^k \binom{y}{k}$$

counts the vectors in $\mathbb{Z}^x$ with 1-norm at most y.

Proof. The size of the key space is the product of the sizes for each batch. In $\Phi(x, y)$ there are k nonzero entries in the x positions and there are $\binom{x}{k}$ ways to determine which entries are nonzero. For each of the nonzero entries there are 2 ways to choose the sign. The vector of partial sums over these k nonzero entries has k different integers in $[1, y]$ and each vector uniquely matches one assignment of partial sums. There are $\binom{y}{k}$ ways to pick k different integers in $[1, y]$. □

6.3 Isogeny atomic blocks

In this section we define *atomic blocks* (AB) as formalizations of the concept of *rounds* in isogeny computations, subroutines that have been widely used before but never formally defined. Algorithms computing the isogeny group action typically proceed as follows: in the first round, the algorithm chooses a series of degrees for which isogenies still need to be computed, and then uses a strategy (Section 4.3.3) to compute a sequence of isogenies of those degrees. The next round chooses a possibly different series of degrees, and computes another sequence of steps. *Atomic blocks* formalize rounds to constant-time computation.

Square-free ABs generalize the approach we take when evaluating the CSIDH group action with the traditional key spaces $\mathcal{K}_m$ and $\mathcal{K}_m^+$ as in Algorithm 2, as we always evaluate one isogeny per degree. *Restricted square-free ABs* are used to evaluate the group action using the batching idea with keys in $\mathcal{K}_{N,m}$; with details in Algorithm 3. We postpone the explicit construction of ABs to Section 6.4.

6.3.1 Square-free atomic blocks

In this section we formalize how constant-time implementations combine isogeny computations after sampling one, respectively two points, and explain why this top layer of the algorithms is independent of the secret key. The lower layer of efficiently computing these blocks in constant-time will be described in Section 6.4.

Definition 6.3.1. Let $R \subseteq \{-1, 0, 1\}$ and let $I = (I_1, \ldots, I_k) \in \mathbb{Z}^k$ such that $1 \leq I_1 < I_2 < \cdots < I_k \leq n$. A *square-free atomic block* of length k is a probabilistic algorithm $\alpha_{R,I}$ with inputs $A \in \mathcal{M}$ and $\epsilon \in R^k$ output $A' \in \mathcal{M}$ and $f \in \{0, 1\}^k$ such that $E_{A'} = (\prod_i \mathfrak{l}_{I_i}^{f_i \cdot \epsilon_i}) \star E_A$, satisfying the following two properties:

1. there is a function σ such that, for each (A, ϵ), the distribution of f, given that (A', f) is returned by $\alpha_{R,I}$ on input (A, ϵ), is $\sigma(R, I)$, and

2. there is a function τ such that, for each (A, ϵ) and each f, the distribution of the time taken by $\alpha_{R,I}$, given that (A', f) is returned by $\alpha_{R,I}$ on input (A, ϵ), is $\tau(R, I, f)$.

That is, a square-free AB outputs a Montgomery coefficient of the new curve, together with a vector f describing which isogeny computations succeeded.

We want constant-time algorithms. If the atomic blocks are already constant-time, this is easier to achieve: we can evaluate the group action on input $e \in \mathcal{K}$ and $A \in \mathcal{M}$ using a sequence of square-free AB calls $(A', f) \leftarrow \alpha_{R,I}(A, \epsilon)$. If in each step the choice of R and I are independent of e, the algorithm does not leak information about e through timing.

This is illustrated by Algorithm 2, which expresses the constant-time group action from [OAYT19] using a sequence of square-free ABs with $R = \{-1, 0, 1\}$ to evaluate the action for keys in $\mathcal{K}_m$. The choices of R and I are independent of e for each AB $\alpha_{R,I}$, and all other steps can be easily made constant-time. The choice of I in Line 3 may vary between different executions, due to the varying failure vectors f of previously evaluated ABs. However this only depends on the initial choice of m_i, and is independent of e.

Algorithm 2 Replacing the inner loop of [OAYT19, Algorithm 3] with any square-free AB with $R = \{-1, 0, 1\}$. Keys are in $\mathcal{K}_m$ for $m = (m_1, \ldots, m_n)$.

Input: $A \in \mathcal{M}$, $e = (e_1, \ldots, e_n) \in \mathcal{K}_m$
Output: A' with $E_{A'} = (\prod_i \mathfrak{l}_i^{e_i}) \star E_A$

1: $(\mu_1, \ldots, \mu_n) \leftarrow (m_1, \ldots, m_n)$
2: **while** $(\mu_1, \ldots, \mu_n) \neq (0, \ldots, 0)$ **do**
3: Set $I = (I_1, \ldots, I_k)$ s.t. $I_1 < \cdots < I_k$ and $\{I_1, \ldots, I_k\} = \{i \mid \mu_i > 0\}$
4: **for** $1 \leq i \leq k$ **do**
5: $\epsilon_i \leftarrow \texttt{Sign}(e_{I_i})$ ▷ 1 if $e_{I_i} > 0$; 0 if $e_{I_i} = 0$; -1 if $e_{I_i} < 0$
6: $(A, f) \leftarrow \alpha_{R,I}(A, (\epsilon_1, \ldots, \epsilon_k))$ ▷ Square-free AB
7: **for** $1 \leq i \leq k$ **do**
8: $(\mu_{I_i}, e_{I_i}) \leftarrow (\mu_{I_i} - f_i, e_{I_i} - \epsilon_i \cdot f_i)$
9: **return** A.

Remark 6.3.2. The constant-time group action from [MCR19] can also be expressed simply in terms of ABs. The algorithm is extremely similar to Algorithm 2, using $\mathcal{K}_m^+$ in place of $\mathcal{K}_m$ and $R = \{0, 1\}$ in place of $\{-1, 0, 1\}$. Line 5 can be simplified to $\epsilon_i \leftarrow 1$ if $e_{I_i} \neq 0$, or 0 if $e_{I_i} = 0$.

Remark 6.3.3. The distribution of the vector f depends on how the atomic blocks are constructed. In [MCR19] and [OAYT19], $\Pr(f_i = 0) = 1/\ell_{I_i}$ for all i. In [CCJR22], f is always $(1, 1, \ldots, 1)$.

6.3.2 Restricted square-free atomic blocks

Restricted square-free ABs are generalizations of square-free ABs to the context of batching. We further require that the that the new atomic blocks do not leak information on which of the primes from a batch we have chosen.

Definition 6.3.4. Let $R \subseteq \{-1, 0, 1\}$, $B \geq 1$, and $I = (I_1, \ldots, I_k) \in \mathbb{Z}^k$ such that $1 \leq I_1 < I_2 < \cdots < I_k \leq B$. A *restricted square-free atomic block* of length k is a probabilistic algorithm $\beta_{R,I}$ taking as input $A \in \mathcal{M}$, $\epsilon \in R^k$, and $J \in \mathbb{Z}^k$ with $1 \leq J_i \leq N_{I_i}$ for all $1 \leq i \leq k$, and returning $A' \in \mathcal{M}$ and $f \in \{0, 1\}^k$ such that $E_{A'} = (\prod_i \mathfrak{l}_{I_i,J_i}^{f_i \cdot \epsilon_i}) \star E_A$, satisfying the following two properties:

1. there is a function σ such that, for each (A, ϵ, J), the distribution of f, given that (A', f) is returned by $\beta_{R,I}$ on input (A, ϵ, J), is $\sigma(R, I)$; and
2. there is a function τ such that, for each (A, ϵ, J) and each f, the distribution of the time taken by $\beta_{R,I}$, given that (A', f) is returned by $\beta_{R,I}$ on input (A, ϵ, J), is $\tau(R, I, f)$.

Algorithm 3 uses restricted square-free ABs with $R = \{-1, 0, 1\}$ to compute group actions for keys in $\mathcal{K}_{N,m}$, generalizing Algorithm 2.

Algorithm 3 A constant-time group action for keys in $\mathcal{K}_{N,m}$ based on restricted square-free ABs with $R = \{-1, 0, 1\}$ with parameters N, m, B.

Input: $A \in \mathcal{M}$, $e = (e_1, \ldots, e_n) \in \mathcal{K}_{N,m}$
Output: A' with $E_{A'} = (\prod_i \mathfrak{l}_i^{e_i}) \star E_A$

$(\mu_1, \ldots, \mu_B) \leftarrow (m_1, \ldots, m_B)$
while $(\mu_1, \ldots, \mu_B) \neq (0, \ldots, 0)$ **do**
  Let $I = (I_1, \ldots, I_k)$ s.t. $I_1 < \cdots < I_k$ and $\{I_1, \ldots, I_k\} = \{i \mid \mu_i > 0\}$
  for $1 \leq i \leq k$ **do**
    if there exists j such that $e_{I_i,j} \neq 0$ **then**
      $J_i \leftarrow$ some such j
    else
      $J_i \leftarrow$ any element of $\{1, \ldots, N_{I_i}\}$
    $\epsilon_i \leftarrow \mathtt{Sign}(e_{I_i,J_i})$ ▷ 1 if $e_{I_i,J_i} > 0$; 0 if $e_{I_i,J_i} = 0$; -1 if $e_{I_i,J_i} < 0$
  $(A, f) \leftarrow \beta_{R,I}(A, (\epsilon_1, \ldots, \epsilon_k), J)$ ▷ Restricted square-free AB
  for $1 \leq i \leq k$ **do**
    $(\mu_{I_i}, e_{I_i,J_i}) \leftarrow (\mu_{I_i} - f_i, e_{I_i,J_i} - \epsilon_i \cdot f_i)$
return A

6.4 Evaluating atomic blocks in constant time

Now we introduce the algorithm used in CTIDH to realize the restricted square-free atomic blocks $\beta_{R,I}$ from Section 6.3. Throughout the section, $R = \{-1, 0, 1\}$.

We start with recasting the inner loop of [OAYT19, Algorithm 3] as a realization of the square-free atomic block $\alpha_{R,I}$. We first present the algorithm in a simpler variable-time form (Algorithm 4) and then explain the small changes needed to eliminate timing leaks, obtaining $\alpha_{R,I}$.

Section 6.4.2 presents our new algorithm to realize $\beta_{R,I}$. The extra challenge here is to hide which prime is being used within each batch. Again we begin by presenting a simpler variable-time algorithm (Algorithm 5) and then explain how to eliminate timing leaks.

6.4.1 Square-free atomic blocks for isogeny evaluation

Algorithm 4 translates the inner loop of [OAYT19, Algorithm 3] to the AB framework. The inputs are $A \in \mathcal{M}$ and $\epsilon \in \{-1, 0, 1\}^k$. The goal is to compute k isogenies of degrees $\ell_{I_1}, \ldots, \ell_{I_k}$, but some of these computations may fail. The outputs are a vector $f \in \{0, 1\}^k$ recording which of the computations succeeded, and A' such that $(\prod_i [\ell_{I_i}^{f_i \cdot \epsilon_i}]) \star E_A = E_{A'}$.

The algorithm uses the 2-point approach with dummy isogenies. It uses two subroutines:

- `UniformRandomPoints` takes $A \in \mathcal{M}$, and returns a uniform random pair of points (T_0, T_1), with $T_0 \in E_A(\mathbb{F}_p)$ and $T_1 \in \tilde{E}_A(\mathbb{F}_p)$; i.e., T_0 is a uniform random element of $E_A(\mathbb{F}_p)$, and T_1, independent of T_0, is a uniform random element of $\tilde{E}_A(\mathbb{F}_p)$.
- `Isogeny` takes $A \in \mathcal{M}$, a point P in $E_A(\mathbb{F}_{p^2})$ with x-coordinate in $\mathbb{F}_p$ generating the kernel of an ℓ_{I_j}-isogeny $\varphi : E_A \to E_{A'} = E_A/\langle P \rangle$, and a tuple of points $(Q_1, \ldots, Q_t)$, and returns A' and $(\varphi(Q_1), \ldots, \varphi(Q_t))$.

Remark 6.4.1. Algorithm 4 uses a multiplicative strategy, but it can easily be modified to use a SIMBA or point-pushing strategy, which is much more efficient in general [OAYT19, CR22]. The isogeny algorithm can be Vélu or $\sqrt{}$élu, whichever is more efficient for the given degree.

Eliminating timing leaks. The following standard modifications to Algorithm 4 produce an algorithm meeting the definition of a square-free atomic block (Definition 6.3.1).

Observe first that $f_j = 1$ if and only if the prime ℓ_{I_j} divides the order of the current T_s. This is equivalent to ℓ_{I_j} dividing the order of the initially sampled point T_s (since T_s has been modified only by multiplication by scalars that are not divisible by ℓ_{I_j}, and by isogenies of degrees not divisible by ℓ_{I_j}). This has probability $1 - 1/\ell_{I_j}$, since the initial T_s is a uniform random point in a cyclic group of size $p+1$. These probabilities are independent across j, since (T_0, T_1) is a uniform random pair of points.

So the distribution of the f vector has position j set with probability $1-1/\ell_{I_j}$, independently across j. This is a function purely of I, independent of (A, ϵ), as required. What follows are algorithm modifications to ensure that the time distribution is a function purely of (I, f); these modifications do not affect f.

Algorithm 4 Inner loop of [OAYT19, Algorithm 3].
Parameters $k \in \mathbb{Z}$, $R = \{-1, 0, 1\}$, $I \in \mathbb{Z}^k_{\geq 0}$

Input: $A \in \mathcal{M}, \epsilon \in \{-1, 0, 1\}^k$
Output: $A' \in \mathcal{M}, f \in \{0, 1\}^k$

1: $(T_0, T_1) \leftarrow \texttt{UniformRandomPoints}(A)$ ▷ $T_0 \in E_A$, $T_1 \in E_{-A}$
2: $(T_0, T_1) \leftarrow ([r]T_0, [r]T_1)$ **where** $r = 4\prod_{i \notin I} \ell_i$
3: $r' \leftarrow \prod_{i \in I} \ell_i$
4: **for** $j = k$ **down to** 1 **do**
5: $\quad r' \leftarrow r'/\ell_{I_j}$
6: $\quad s \leftarrow \texttt{SignBit}(\epsilon_j)$ ▷ 1 if $\epsilon_j < 0$, otherwise 0
7: $\quad P \leftarrow [r']T_s$
8: $\quad$ **if** $P \neq \infty$ **then** ▷ branch without secret information
9: $\quad\quad f_j \leftarrow 1$
10: $\quad\quad (A', (T_0', T_1')) \leftarrow \texttt{Isogeny}(A, P, (T_0, T_1), I_j)$
11: $\quad\quad$ **if** $\epsilon_j \neq 0$ **then** ▷ branch with secret information
12: $\quad\quad\quad (A, T_0, T_1) \leftarrow (A', T_0', T_1')$
13: $\quad\quad$ **else**
14: $\quad\quad\quad T_s \leftarrow [\ell_{I_j}]T_s$
15: $\quad$ **else**
16: $\quad\quad f_j \leftarrow 0$
17: $\quad T_{1-s} \leftarrow [\ell_{I_j}]T_{1-s}$
18: **return** A, f

Step 7, taking T_0 if $s = 0$ or T_1 if $s = 1$, is replaced with a constant-time point selection: e.g., taking the bitwise XOR $T_0 \oplus T_1$, then ANDing each bit with s, and then XORing the result with T_0. Similar comments apply to the subsequent uses of T_s and T_{1-s}. It is simplest to merge all of these selections into a constant-time swap of T_0, T_1 when $s = 1$, followed by a constant-time swap back at the bottom of the loop. The adjacent swaps at the bottom of one loop and the top of the next loop can be merged, analogous merging is standard in constant-time versions of the Montgomery ladder for scalar multiplication.

Step 11 determines whether an actual isogeny or a dummy isogeny has to be computed. The conditional assignment to (A, T_0, T_1) in the first case is replaced with unconditional constant-time point selection. The conditional operation in the second case is replaced with an unconditional operation, multiplying T_s by ℓ_{I_j} in both cases. This changes the point T_s in the first case, but does not change the order of T_s (since the isogeny has already removed ℓ_{I_j} from the order of T_s in the first case), and all that matters for the algorithm is the order. One can perform the multiplication by ℓ_{I_j} within a dummy isogeny computation [MCR19, OAYT19].

The branch in Step 8 is determined by public information f_j. The isogeny computation inside `Isogeny` takes constant time with standard algorithms; at a

lower level, arithmetic in $\mathbb{F}_p$ is handled by constant-time subroutines. The computation of `UniformRandomPoints` takes variable time with standard algorithms, but the time distribution is independent of the input curve.

The total time is the sum for initialization (`UniformRandomPoints`, computation of r and r', initial scalar multiplication), f_j computation (division, scalar multiplications, selection), and computations when $f_j = 1$ (`Isogeny`, scalar multiplication, selection). This sum is a function of (I, f), independent of (A, ϵ).

6.4.2 Restricted square-free atomic blocks

We now consider the more difficult goal of hiding which isogeny is being computed within each batch. We present first the high-level algorithm (Algorithm 5), then discuss the `PointAccept` and `MatryoshkaIsogeny` subroutines, and finally the algorithm modifications to meet Definition 6.3.4.

The inputs to Algorithm 5 are $A \in \mathcal{M}$, $\epsilon \in \{-1, 0, 1\}^k$, and $J \in \mathbb{Z}^k$. The goal is to compute k isogenies of degrees $\ell_{I_1,J_1}, \ldots, \ell_{I_k,J_k}$. The outputs are $A' \in \mathcal{M}$ and $f \in \{0, 1\}^k$ such that $(\prod_i \mathfrak{l}_{I_i,J_i}^{f_i \cdot \epsilon_i}) \star E_A = E_{A'}$.

Like Algorithm 4, Algorithm 5 uses a 2-point approach and dummy isogenies. It uses the following subroutines:

- `UniformRandomPoints` is as before.
- `PointAccept` replaces the check $P \neq \infty$ to prevent timing leakage. It takes a point P and $I_j, J_j \in \mathbb{Z}$ such that P either has order ℓ_{I_j,J_j} or 1, and outputs either 0 or 1, under the condition that the output is 0 whenever $P = \infty$.
- `MatryoshkaIsogeny` replaces `Isogeny` from Algorithm 4. There is an extra input J_j indicating the secret position within a batch.

Note that the output of `PointAccept` can be 0 when $P \neq \infty$, so we add a multiplication by ℓ_{I_j,J_j} in Step 14 to make sure we continue the loop with points of expected order.

`PointAccept`. Step 8 of Algorithm 5 computes a potential kernel point P. The probability that $P = \infty$ is $1/\ell_{I_j,J_j}$, which depends on J_j. For the batch $(\ell_{I_j,1}, \ldots, \ell_{I_j,N_{I_j}})$, `PointAccept` artificially increases this probability to $1/\ell_{I_j,1}$, independent of ℓ_{I_j,J_j} by tossing a coin with success probability

$$\gamma = \frac{\ell_{I_j,J_j} \cdot (\ell_{I_j,1} - 1)}{\ell_{I_j,1} \cdot (\ell_{I_j,J_j} - 1)}$$

and only returns $f_j = 1$ if $P \neq \infty$ *and* the coin toss is successful. The probability that the output is 1 is then $\gamma \cdot (1 - 1/\ell_{I_j,J_j}) = 1 - 1/\ell_{I_j,1}$ which is independent of J_j (note that $\ell_{I_j,1}$ is the smallest prime in the batch).

Algorithm 5 The CTIDH inner loop. Parameters $k \in \mathbb{Z}, R = \{-1, 0, 1\}, I \in \mathbb{Z}_{\geq 0}^k$

Input: $A \in \mathcal{M}, \epsilon \in \{-1, 0, 1\}^k, J \in \mathbb{Z}_{>0}^k$
Output: $A' \in \mathcal{M}, f \in \{0, 1\}^k$

$(T_0, T_1) \leftarrow$ `UniformRandomPoints`(A) ▷ $T_0 \in E_A$, $T_1 \in E_{-A}$
$(T_0, T_1) \leftarrow ([r]T_0, [r]T_1)$ **where** $r = 4 \prod_{i \notin I} \prod_{1 \leq j \leq N_i} \ell_{i,j}$
$(T_0, T_1) \leftarrow ([\tilde{r}]T_0, [\tilde{r}]T_1)$ **where** $\tilde{r} = \prod_{i \in I} \prod_{1 \leq j \leq N_i, j \neq J_i} \ell_{i,j}$ ▷ hide selection
$r' \leftarrow \prod_{i \in I} \ell_{i,J_i}$ ▷ hide selection
for **do**$j = k$ **down to** 1
    $r' \leftarrow r'/\ell_{I_j,J_j}$ ▷ hide ℓ_{I_j,J_j}, batch is public
    $s \leftarrow$ `SignBit`(ϵ_j) ▷ 1 if $\epsilon_j < 0$, otherwise 0
    $P \leftarrow [r']T_s$ ▷ hide ℓ_{I_j,J_j}, batch is public
    $f_j \leftarrow$ `PointAccept`(P, I_j, J_j)
    if $f_j = 1$ **then** ▷ this branch is on public information
        $(A', (T_0', T_1')) \leftarrow$ `MatryoshkaIsogeny`$A, P, T_0, T_1, I_j, J_j)$
        if $\epsilon_j \neq 0$ **then** ▷ branch with secret information
            $(A, T_0, T_1) \leftarrow (A', T_0', T_1')$
    $(T_0, T_1) \leftarrow ([\ell_{I_j,J_j}]T_0, [\ell_{I_j,J_j}]T_1)$ ▷ hide selection
return A, f

`MatryoshkaIsogeny`. The function `MatryoshkaIsogeny` replaces the `Isogeny` computation. It takes as input the Montgomery coefficient of a curve E_A, a batch $(\ell_{i,1}, \ldots, \ell_{i,N_i})$, an isogeny index j within the batch, a point P of order $\ell_{i,j}$ generating the kernel of an isogeny $\varphi : E_A \to E_A/\langle P \rangle = E_{A'}$, and a tuple of points $(Q_1, \ldots, Q_t)$, and returns A' and $(\varphi(Q_1), \ldots, \varphi(Q_t))$. `MatryoshkaIsogeny` is computed with cost independent of j.

From Vélu's formulas, [BLMP19] computed any ℓ_i-isogeny for $\ell_i \leq \ell$ using the computation of an ℓ-isogeny and masking. Recall that the first step of computing Vélu's formulas is to compute $x(P), x([2]P), \ldots, x([(\ell-1)/2]P)$, and their product (Equation (4.8)). This includes the computation for all $\ell_i < \ell$, reminiscent of a Matryoshka-doll [BLMP19].

In CTIDH, we specialize the √élu formulas so as to obtain a Matryoshka-doll structure. We define the sets U and V (Section 4.3.2), as the optimal choices for the *smallest* degree in the batch: i.e., $\ell_{i,1}$. The leftover set W is chosen to make the formulas work even for the *largest* prime ℓ_{i,N_i} in the batch. The baby-step giant-step part of the algorithm stays unchanged; while we iterate through W we save the intermediate results corresponding to *all degrees* $\ell_{i,j}$ in the batch. In the final step, we select the result for the degree that we wanted to compute.

The sets U and V have size $\approx \sqrt{\ell_{i,1}}$. If all the primes in the batch are of similar size, the optimal choices of U_i and V_i for each i are close to U and V and the Matryoshka-like formulas are close in performance to the optimal formulas.

Eliminating timing leaks. We now describe how to modify Algorithm 5 to meet the definition of a restricted square-free atomic block (Definition 6.3.4).

We begin with the distribution of f. For each input (A, ϵ, J), the distribution has f_j set with probability $1 - 1/\ell_{I_j,1}$ independently across j. This distribution is a function purely of I, independent of (A, ϵ, J), as required. What remains is to ensure that the time distribution is a function purely of (I, f).

There are secret scalars $\tilde{r}$, r', and ℓ_{I_j,J_j} used in various scalar multiplications in Steps 3, 8, and 14. Standard integer-arithmetic algorithms that dynamically suppress leading zero bits are replaced by constant-time algorithms that always use the maximum number of bits, and variable-time scalar-multiplication algorithms are replaced by a constant-time Montgomery ladder, as in [BLMP19]. It is straightforward to compute an upper bound on each scalar in Algorithm 5. See Section 6.6 for faster alternatives.

Isogenies using `MatryoshkaIsogeny` can be computed in time that depends only on the batch, not on the selection of a prime within the batch. Everything else is as in Section 6.4.1: the distribution of `UniformRandomPoints` timings is independent of the inputs, Step 8 uses constant-time selection, the branch in Step 12 is replaced by constant-time selection, and the branch in Step 10 does not need to be modified.

6.5 Strategies and parameters for CTIDH

The optimization process for previous constant-time algorithms for CSIDH has two levels. The bottom level tries to minimize the cost of each AB, for example by optimizing $\sqrt{}$élu parameters and searching for a choice of strategy. The top level searches for exponent bounds $m = (m_1, \ldots, m_n)$ that minimize the total cost subject to the key space size, modeling the cost of ABs by a cost function.

Optimizing CTIDH is more complicated. There is a new top level, searching for a choice of batch sizes $N = (N_1, \ldots, N_B)$. Batch sizes influence the success chance and cost of an AB at the bottom level as the discussion in Section 6.4.2 shows. Batches also influence the total cost of any particular choice of 1-norm bounds $m = (m_1, \ldots, m_B)$ at the middle level. The size of the key space depends on both N and m; see Lemma 6.2.2.

We describe a reasonably efficient method to search for CTIDH parameters.

Strategies for CTIDH. We save time at the lowest level of the search by using multiplicative strategies. As in previous papers, it would be easy to adapt Algorithm 5 to use other strategies, but this is unlikely to produce large benefits.

Seen from a high level, evaluating ABs multiplicatively in CTIDH has a similar effect to SIMBA strategies for previous algorithms. For example, SIMBA-N_1 for traditional batch sizes $(1, \ldots, 1)$ limits each AB to at most n/N_1 isogenies (if n is divisible by N_1), in order to save multiplicative effort. Now consider CTIDH

where all B batches have size N_1, i.e., $N = (N_1, \ldots, N_1)$. Each CTIDH AB then computes at most $B = n/N_1$ isogenies, again saving multiplicative effort.

One could split a CTIDH AB into further SIMBA prides, but [MCR19] already shows that most of the benefit of SIMBA comes from putting some cap on the number of isogenies in an AB; the exact choice of cap is relatively unimportant. One could also try to optimize point-pushing strategies as an alternative to multiplicative strategies, as an alternative to SIMBA, or within each SIMBA pride, but the searches in [HLKA20] and [CR22] suggest that optimizing these strategies saves at most a small percentage in the number of multiplications, while incurring overhead for managing additional points.

Cost functions for CTIDH. The search through various CTIDH batching configuration vectors N and 1-norm bound vectors m tries to minimize a cost function $C(N, m)$, a model of the cost of a group-action evaluation. The numerical search examples later in this section use the following cost function: the average number of multiplications (counting squarings as multiplications) used by the CTIDH algorithms, including the speedups described in Section 6.6.

One way to compute this function is to statistically approximate it: run the software from Section 6.6 many times, inspect the multiplication counter built into the software, and take the average over many experiments. A more efficient way to compute the same function with the same accuracy is with a simulator that skips the multiplications but still counts how many there are. Our simulator, written in Python, is about 50 times faster than the software from Section 6.6.

However, using a statistical approximation raises concerns about the impact of statistical variations. So, instead of using the software or the simulator, we directly compute the average cost of the first AB, the average cost of the second AB, etc., stopping when the probability of needing any further AB is below 10^{-9}.

Batch b, with smallest prime $\ell_{b,1}$, has success probability $1 - 1/\ell_{b,1}$ from each AB, so the chance q_b of reaching m_b successes within R ABs is the sum of the coefficients of $x^{m_b}, x^{m_b+1}, \ldots$ in the polynomial $(1/\ell_{b,1} + (1-1/\ell_{b,1})x)^R$. Batches are independent, so $q_1 q_2 \cdots q_B$ is the probability of not needing any further AB. Note that multiplying the polynomial $(1/\ell_{b,1} + (1 - 1/\ell_{b,1})x)^R$ by $1/\ell_{b,1} + (1 - 1/\ell_{b,1})x$ for each increase in R is more efficient than computing binomial coefficients.

Computing the cost of an AB (times the probability that the AB occurs) is more complicated. Splitting the analysis into 2^B cases—e.g., one case, occurring with probability $(1 - q_1)(1 - q_2) \cdots (1 - q_B)$, is that all B batches still remain to be done—might be workable, since B is not very large and one can skip cases that occur with very low probability. We instead take the following approach. Fix b. The probability that batch b is in the AB is $1 - q_b$; the probability that batch a is in the AB for exactly j values $a < b$ is the coefficient of x^j in the polynomial $\prod_{a<b}(q_a + (1 - q_a)x)$; the probability that batch c is in the AB for exactly k values $c > b$ is the coefficient of x^k in $\prod_{c>b}(q_c + (1 - q_c)x)$. There

are $O(B^2)$ possibilities for (j, k); each possibility determines the total number of batches in the AB and the position of b in the AB, assuming b is in the AB. For the AB algorithms considered here, this is enough information to determine the contribution of batch b to the cost of the AB. Our Python implementation of this approach has similar cost to 100 runs of the simulator, depending on B.

We also explored various simpler possibilities for cost functions. A deterministic model of ABs is easier to compute and simulates real costs reasonably well, leading to parameters whose observed costs were consistently within 10% of the best costs we found via the cost function defined above.

Optimizing the 1-norm bounds. Given a fixed configuration N of B batches, we use a greedy algorithm as in [CR22] to search for a 1-norm bound vector m:

1. Choose an initial $m = (m_1, \ldots, m_B)$ such that $\mathcal{K}_{N,m}$ is large enough, and set $C_{\min} \leftarrow C(N, m)$.

2. For each i in $\{1, \ldots, B\}$, do the following:

   (a) Set $\tilde{m} \leftarrow (m_1, \ldots, m_{i-1}, m_i - 1, m_{i+1}, \ldots, m_B)$.

   (b) If $\mathcal{K}_{N,\tilde{m}}$ is large enough, set $(m, C_{\min}) \leftarrow (\tilde{m}, C(N, \tilde{m}))$.

   (c) Else, set $\tilde{m}' \leftarrow \tilde{m}$, and for each $j \neq i$ in $\{1, \ldots, B\}$ do the following:

      i. Set $\tilde{m} \leftarrow (\tilde{m}'_1, \ldots, \tilde{m}'_{j-1}, \tilde{m}'_j + 1, \tilde{m}'_{j+1}, \ldots, \tilde{m}'_B)$.

      ii. If $\mathcal{K}_{N,\tilde{m}}$ is too small, recursively go to Step 2(c).

      iii. Else, if $C(N, \tilde{m}) < C_{\min}$, set $(m, C_{\min}) \leftarrow (\tilde{m}, C(N, \tilde{m}))$.

3. If $C_{\min}$ was updated in Step 2, then repeat Step 2.

4. Return $(m, C_{\min})$.

This algorithm applies small changes to the bound vector m at each step, reducing one entry while possibly increasing others. Obviously, this finds only a *locally* optimal m with respect to these changes and the initial choice of m in Step 1; different choices generally produce different results.

One way to choose an initial m is to try $(1, 1, \ldots, 1)$, then $(2, 2, \ldots, 2)$, etc., stopping when $\mathcal{K}_{N,m}$ is large enough. Another approach, is to start from the best m found for the parent N below, and merely increase the first component of m until $\mathcal{K}_{N,m}$ is large enough; usually at most one increase is needed.

The algorithm involves at least $B(B-1)$ evaluations of the cost function for the final pass through Step 2, and possibly many more if there are many recursive calls or many improvements to m, but usually these are small effects.

Optimizing the prime batches. We optimize N via a similar greedy algorithm, using the algorithm above as a subroutine. For a *fixed* number of batches B, we proceed as follows:

1. Choose an initial $N = (N_1, \ldots, N_B)$ with $\sum_i N_i = n$, and let $(m, C_{\min})$ be the output of the algorithm above applied to N.
2. For each $i \in \{1, \ldots, B\}$, do the following:
   (a) Set $\tilde{N}^i \leftarrow (N_1, \ldots, N_{i-1}, N_i - 1, N_{i+1}, \ldots, N_B)$.
   (b) For each $j \neq i$ in $\{1, \ldots, B\}$,
      i. Set $\tilde{N}^{i,j} \leftarrow (\tilde{N}^i_1, \ldots, \tilde{N}^i_{j-1}, \tilde{N}^i_j + 1, \tilde{N}^i_{j+1}, \ldots, \tilde{N}^i_B)$.
      ii. Let $(\tilde{m}, \tilde{C})$ be the output of the algorithm above applied to $N^{i,j}$.
      iii. If $\tilde{C} < C_{\min}$, then update $(N, m, C_{\min}) \leftarrow (\tilde{N}^{i,j}, \tilde{m}, \tilde{C})$.
3. If $C_{\min}$ was updated in Step 2, then repeat Step 2.
4. Return N, m, and $C_{\min}$.

This algorithm also finds only a local optimum with respect to these changes, and with respect to the initial choice of N in Step 1. Our experiments took an initial choice for N such that $z \leq N_1 \leq \cdots \leq N_B \leq z + 1$ for some $z \in \mathbb{Z}$. One can also omit one or more large primes ℓ_j by taking each $N_j = 1$ and $m_j = 0$.

Within the full two-layer greedy algorithm, each N considered at the upper layer involves $B(B-1)$ calls to the lower layer, the optimization of 1-norm bounds. Recall that each call to the lower layer involves at least $B(B-1)$ evaluations of the cost function. Overall there are nearly B^4 evaluations of the cost function.

Numerical examples. Table 6.1 shows examples of outputs of the above search. For each B, the "N"/"m" column shows the final (N, m) found, and the "cost" column shows the cost function for that (N, m), to two digits after the decimal point. The full table can be found in [BBC+21, Table 1]

For the search, we used a server with two 64-core AMD EPYC 7742 CPUs, but limited each search to 32 cores running in parallel. We parallelized only the upper layer of the search; often fewer than 32 cores were used since some calls to the lower layer were slower than others. For each B, "wall" shows the seconds of real time used for the search, and "CPU" shows the total seconds of CPU time (across all cores, user time plus system time) used for the search.

6.6 Constant-time software for the action

We have built a self-contained high-performance software package, `high-ctidh`, that includes implementations of all of the operations needed by CSIDH users for

B	wall CPU	cost	N m
6	13.29 150.29	583256.02	10 11 12 12 15 14 19 31 31 32 32 30
9	138.65 2763.28	485052.29	5 7 8 8 8 7 10 12 9 14 22 23 23 23 23 24 24 13
13	2301.94 55252.51	445054.10	3 4 4 6 6 6 7 7 8 7 7 8 1 10 16 17 18 18 18 18 18 18 18 17 14 1
14	**6509** **161371.00**	**437985.55**	**2 3 4 4 5 5 6 7 7 8 8 6 8 1** **10 14 16 17 17 17 18 18 18 18 18 13 13 1**
15	8341 211336.80	440201.56	3 4 3 4 4 5 5 5 6 6 6 7 7 8 1 9 14 15 15 16 16 16 16 16 16 16 16 15 13 1

Table 6.1: Results of searches for CTIDH parameters with at least 2^{256} keys for the CSIDH-512 prime. Details in text. Batching with smallest cost is in bold.

whichever parameter set is selected: constant-time key generation, constant-time computation of the CSIDH action, and validation of (claimed) public keys. The package uses the CTIDH key space and CTIDH algorithms to set new cycle-count records for constant-time CSIDH.

The `high-ctidh` source code is in C, with assembly language for field arithmetic. Beyond the performance benefits, using low-level languages is helpful for ensuring constant-time behavior, as explained below. Measuring the performance of a full C implementation also resolves the concerns raised by using multiplications as a predictor of performance, such as concerns that some subroutines could be difficult to handle in constant time and that improved multiplication counts could be outweighed by overhead.

The software is freely available at `http://ctidh.isogeny.org/`. This section describes the software. Section 6.7 reports the software speeds and compares with previous implementations.

Processor selection and field arithmetic. The original CSIDH paper reported clock cycles for variable-time CSIDH-512 software on an Intel Skylake CPU core. Skylake is also the most common CPU choice in followup papers on CSIDH software speed. We similarly focus on Skylake to maximize comparability.

The original `csidh-20180826` software [CLM+18] included a small assembly-language library for Intel chips (Broadwell and newer) to perform arithmetic modulo the CSIDH-512 prime. The same library has been copied, with minor tweaks and generalizations to other primes, into various subsequent software packages, including `high-ctidh`. Code above the field-arithmetic level, decomposing isogenies into multiplications etc., are written in C, so porting the software to another CPU is mainly a matter of writing an efficient Montgomery multiplier for that

CPU. Beware that each CPU will have different cycle counts, and possibly a different ranking of algorithmic choices.

The `velusqrt-asm` software from [BDLS20] includes an adaptation of the same library to CSIDH-1024. The `sqale-csidh-velusqrt` software [CCJR22] includes adaptations to larger sizes, all automatically generated by a code generator that takes p as input. The `high-ctidh` package includes a similar code generator, with some small improvements in the details: for example, we use less arithmetic for conditional subtraction, and we avoid `cmov` instructions with memory operands out of concern that they could have data-dependent timings.

6.6.1 Computing one isogeny

The middle layer of `high-ctidh` computes an ℓ-isogeny for one prime ℓ; it also includes auxiliary functions such as multiplying by the scalar ℓ. We built this layer as follows. We started with the `xISOG` function in `velusqrt-asm`. As in `csidh-20180826`, this function takes a curve and a point P of order ℓ, and returns the corresponding ℓ-isogenous curve. It also takes a point T, and returns the image of that point under the isogeny.

We extended the function interface to take lower and upper bounds on ℓ—the smallest and largest prime in the batch containing ℓ—and we modified the software to take time depending only on these bounds, not on the secret ℓ. The Matryoshka-doll structure of the computation (see Section 6.4.2) meant that very little code had to change. Each loop to ℓ is replaced by a loop to the upper bound, with constant-time conditional selection of the results relevant to ℓ; and ℓ is replaced by the lower bound as input to the $\sqrt{\text{élu}}$ parameter selection. An upper bound was used the same way in [BLMP19]; the use of the lower bound for a Matryoshka-doll $\sqrt{\text{élu}}$ is new here.

We reused the $\sqrt{\text{élu}}$ parameter-tuning mechanisms from `velusqrt-asm`. These automatic mechanisms offer the option of tuning for multiplication counts or cycles. Since most CSIDH-related papers report multiplication counts while fewer report cycles, we chose to tune for multiplication counts for comparability.

We made more changes to incorporate known optimizations, including an observation from [BLMP19] regarding the applicability of multiexponentiation, and an observation from [ACR23] regarding reciprocal polynomials. Computing a 587-isogeny and pushing a point through takes 2108 multiplications in this software (counting squarings as multiplications); for comparison, [ACR23] took 3.4% more, and `velusqrt-asm` took 8.9% more.

More importantly for the high-level algorithms, we extended the interface to allow an array of points T to be pushed through the isogeny—e.g., two or zero points rather than one. We also incorporated shorter differential addition chains, as in [CCC+19], for scalar multiplications, and standard addition chains for the constant-time exponentiation inside Legendre-symbol computation.

There would be marginal speedups from tuning the $\sqrt{\text{élu}}$ parameters for each

number of points separately. The parameters $(6, 3)$ for 0 points (instead of $(0, 0)$) saves 2 out of 328 multiplications for $\ell = 79$; further 2 out of 344 multiplications for $\ell = 83$; and 8 out of 368 multiplications for $\ell = 89$. Parameter adjustments also save 3 multiplications for 0 points for each $\ell \in \{557, 587, 613\}$. However, we did not find such speedups for most primes, and we did not find such speedups for the much more common case of 2 points.

6.6.2 Computing the action

The top layer of `high-ctidh` is new, and includes the core CTIDH algorithms. The key space is $\mathcal{K}_{N,m}$, allowing any vector with 1-norm at most m_1 for the first N_1 primes, 1-norm at most m_2 for the first N_2 primes, etc. Constant-time generation of a length-N_i vector of 1-norm at most m_i works as follows:

- Generate $N_i + m_i$ uniform random b-bit integers.
- Set the bottom bit of each of the first N_i integers, and clear the bottom bit of each of the last m_i integers.
- Sort the integers. (We reused constant-time sorting software from [Ber18].)
- If any adjacent integers are the same outside the bottom bit, start over. (Otherwise the integers were distinct outside the bottom bit, so sorting them applies a uniform random permutation.)
- Extract the bottom bit at each position. (This is a uniform random bit string of length $N_i + m_i$ with exactly N_i bits set.)
- Consider the entries as integers. Add the first entry to the second, then add the resulting second entry to the third, etc. (Now there are maybe some 0s, then at least one 1, then at least one 2, and so on through at least one N_i.)
- Count, in constant time, the number e_0 of 0, the number e_1 of 1, and so on through the number e_{N_i-1} of $N_i - 1$. (These add up to at most $N_i + m_i - 1$, since the number of N_i was not included. Each of $e_1, \ldots, e_{N_i-1}$ is positive, and e_0 is nonnegative.)
- Subtract 1 from each of $e_1, \ldots, e_{N_i-1}$. (Now $e_0, \ldots, e_{N_i-1}$ is a uniform random string of N_i nonnegative integers with sum at most m_i.)
- Generate a uniform random N_i-bit string $s_0, \ldots, s_{N_i-1}$.
- Compute, in constant time, whether any j has $s_j = 1$ and $e_j = 0$. If so, start over.
- Replace each e_j with $-e_j$ if $s_j = 1$.

The two rejection steps in this algorithm are independent of the secrets produced as output. The first rejection step is very unlikely to occur when b is chosen so that 2^b is larger than $(N_i+m_i)^2$. The second rejection step occurs more frequently. Sign variations for vectors of Hamming weight k contribute 2^k by Lemma 6.2.2 and thus the rejection correctly happens more frequently for smaller k.

In the case $N_i > m_i$, `high-ctidh` saves time by skipping (in constant time) the $s_j = 1$ rejection test for the first $N_i - m_i$ values of j having $e_j = 0$. There are always at least $N_i - m_i$ such values of j. This increases each acceptance chance by a factor $2^{N_i-m_i}$, preserving uniformity of the final output.

Once a private key is generated, the action is computed by restricted square-free ABs. As in Section 6.3, the first AB handles one prime per batch, the next AB handles one prime per batch that might have something left to do, etc.

Within each AB, Elligator 2 [BHKL13] (see also [BBC+21, Appendix B]) is used twice to generate two independent points. We first generate points with Elligator on the first curve E_A and select the point with the same orientation as the first isogeny. Then we push the point through and use Elligator again to generate two points with Elligator on the second curve $E_{A'}$ and select the point of opposite orientation. Both choices are secret. Both points (T_0, T_1) are pushed through subsequent isogenies as in Algorithm 5; no points are pushed through the last isogeny and only one point is pushed through the isogeny before that. The AB thus pushes through $1, 2, 2, 2, \ldots, 2, 2, 2, 1, 0$ points.

The software permutes the $b \le B$ batches in every AB to use primes in the following order: $\ell_{b-1}, \ell_{b-3}, \ell_{b-4}, \ldots, \ell_1, \ell_{b-2}, \ell_b$. Each AB selects one prime from each batch in the block and tries to compute an isogeny of total degree D, the product of the selected primes; $D = r'$ in Algorithm 5. Each point is multiplied by 4 and all primes outside D immediately after being generated by Elligator, so that the order of the point divides D. There are two types of primes outside D (compare Steps 2 and 3 of Algorithm 5):

- The batches in the AB are public. Primes outside these batches are publicly outside D.
- Primes that are inside the batches in the AB, but that are not the secretly selected prime per batch, are secretly outside D.

For scalar multiplication by a product of secret primes, [BLMP19] uses a Montgomery ladder, with the number of ladder steps determined by the maximum possible product. For public primes, [CCC+19] does better using a precomputed differential addition chain for each prime. Our `high-ctidh` software also uses these chains for secret primes, taking care to handle the incompleteness of differential-addition formulas and to do everything in constant time. The primes in a batch usually vary slightly in chain length, so the software always runs to the maximum length.

Each ℓ-isogeny then clears ℓ from the order of the point that was used to compute the isogeny. As in line 14 of Algorithm 5, the software multiplies the point by ℓ anyway (again using a constant-time differential addition chain), just in case this was a dummy isogeny, i.e., there was secretly nothing left to do in the batch. This extra scalar multiplication could be merged with the isogeny computation, but the $\sqrt{}$élu structure seems to make this somewhat more complicated than in [MCR19], and the extra scalar multiplication accounts for only about 3% of the CSIDH-512 computation. The other point is also multiplied by ℓ.

An AB successfully handling a batch is a public event, visible in timing: it means that a (real or dummy) ℓ-isogeny is computed for some ℓ in the batch, publicly decreasing the maximum 1-norm of the batch. This event occurs with probability $1 - 1/\ell_{i,1}$, where $\ell_{i,1}$ is the smallest prime in the batch containing ℓ. As in Section 6.4.2, this is achieved artificially inflating the failure probability of computing the ℓ-isogeny by using a γ-biased coin toss for $\gamma = (1-1/\ell_{i,1})/(1-1/\ell)$.

One obvious way to generate a γ-biased coin is to (1) generate a uniform random integer modulo $\ell_{i,1}(\ell-1)$ and (2) compute whether the integer is smaller than $\ell(\ell_{i,1}-1)$. The second step is easy to do in constant time. For the first step, the software generates a uniform random 256-bit integer and, in constant time, reduces that modulo $\ell_{i,1}(\ell-1)$; the resulting distribution is indistinguishable from uniform. One could instead use rejection sampling to compute a uniform random integer modulo $\ell_{i,1}M$, where M is the least common multiple of $\ell-1$ across primes ℓ in the batch, and then reduce the integer modulo $\ell_{i,1}(\ell-1)$, to obtain an exactly uniform distribution; the reason to use M here rather than just one $\ell-1$ is to avoid having the secret ℓ influence the rejection probability.

6.6.3 Automated constant-time verification

We claim that the CTIDH algorithm is constant time, and base our claim on reviewing the mathematical properties of the new algorithm, reviewing each new line of the code in `high-ctidh`, and every line of the reused code. These analyses were done entirely by hand. For extra assurance, we designated an internal auditor to use the automated tool `valgrind` to verify the constant-time claims.

The standard tool `valgrind` [NS07] runs a specified binary, watching each instruction for memory errors—in particular, branches and array indices derived from undefined data. If secret data in cryptographic software is marked as undefined then simply running `valgrind` will automatically check whether there is any data flow from secrets to branches and array indices; see, e.g., [Lan10]. See also [Jan21] for a survey of related tools.

Because `valgrind` works at the binary level, this analysis includes any optimizations that might have been introduced by the compiler. A compiler change could generate a different binary with timing leaks, but `valgrind` is fast enough to be systematically run on all compiled cryptographic software before deployment.

The auditor wrote a simple `checkct` program using `high-ctidh` to perform

a full CSIDH key exchange; this is included in the `high-ctidh` package. For example, running `valgrind ./checkct512default` (performing a full CSIDH-512 key exchange) takes under 30 seconds on a 3GHz Skylake core, The underlying `randombytes` function marks all of its output as undefined, so `valgrind` is checking for any possible data flow from randomness to branches or to array indices. For each size, `valgrind` completes successfully, indicating that there is no such data flow. For more details, see [BBC+21, Sec. 7.4].

6.7 Software speeds

This section reports various measurements of the `high-ctidh` software from Section 6.6, and compares with previous speeds for constant-time CSIDH. For a full discussion of the performance of CTIDH, see [BBC+21, Sec. 8]

We use the standard notation **M** for multiplications not including squarings, **S** for squarings, and **a** for additions including subtractions. One common metric in the literature is $(\mathbf{M}, \mathbf{S}, \mathbf{a}) = (1, 1, 0)$, counting the total number of multiplications while ignoring the costs of addition and ignoring possible squaring speedups. Another common metric is $(\mathbf{M}, \mathbf{S}, \mathbf{a}) = (1, 0.8, 0.05)$.

We performed $a = 16383$ group action computations for each of $c = 65$ private keys (as in [BDLS20]) and recorded the cycle count, a total multiplication count, and the total number of squarings, and of additions (including subtractions). Experiments were run on a single core on a 3GHz Intel Xeon E3-1220 v5 (Skylake) CPU with Turbo Boost disabled. This CPU does not support hyperthreading.

6.7.1 CTIDH parameters

We discuss the parameters CTIDH-512, CTIDH-1024, and CTIDH-511, the performance of which can be found in Table 6.2.

Example 6.7.1 (CTIDH-512)**.** This parameter set uses the CSIDH-512 prime, see Example 4.2.4, and we change the key space to $\mathcal{K}_{N,m}$. The best parameters we have found were (N, m) shown for $B = 14$ in Table 6.1. For this (N, m), the keyspace $\mathcal{K}_{N,m}$ has $\approx 2^{256.009}$ keys. It uses $B = 14$ batches and the largest m_i in m is $m_i = 18$.

Our cost calculator claimed that one group action for CTIDH-512 would require approximately 437986 multiplications on average; Table 6.2 reports costs as 438006 (resp. 438762) multiplications, depending on the metric for $(\mathbf{M}, \mathbf{S}, \mathbf{a})$.

Example 6.7.2 (CTIDH-1024)**.** We use the CSIDH-1024 prime, and define a new key space $\mathcal{K}_{N,m}$. The best parameters we found are batch sizes (N, m) with 23 batches of size $2, 3, 5, 4, 6, 6, 6, 6, 6, 7, 7, 7, 6, 7, 7, 5, 6, 5, 10, 3, 10, 5, 1$, with the following bounds $m = (2, 4, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 3, 6, 2, 6, 2, 0)$. Note

that the last 0 indicates that it is the most efficient to not include the computation by the largest prime. There are approximately $2^{256.066}$ keys.

We see from Table 6.2 that the average number of multiplications 375683, resp. 382432 is smaller for CTIDH-1024 than CTIDH-512. Note that CTIDH-1024 uses a large prime, which, unsurprisingly, makes each mutliplication slower. This is similar to [ACR23, Tables 1 and 2]. The reason is that even though the cofactors are larger, the larger list of primes $\ell_1, \dots \ell_n$ also means that we can use smaller exponents for the same key space.

Example 6.7.3 (CTIDH-511)**.** To understand the effect of changing the size of the keyspace, we define a key space for the CSIDH-512 prime with key space of size 2^{220}, as in the software from [CCJR22].

The key space $\mathcal{K}_{N,m}$ uses 15 batches of size $2, 3, 4, 4, 5, 5, 5, 5, 5, 7, 7, 8, 7, 6, 1$, with bounds $m = (6, 9, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 8, 6, 1)$. There are approximately $2^{220.004}$ keys.

This results in a significant speedup of requiring 310945 (resp. 311852) multiplications, see Table 6.2.

We also note that generating the secret key, as in Section 6.6.2 typically costs under 1 million cycles, which is a negligible cost compared to generating the corresponding public key.

6.7.2 Comparisons

There have been several previous speed reports for constant-time CSIDH [MCR19, OAYT19, CCC+19, HLKA20, CR22, ACR23, CCJR22]. A parameter set of the CSIDH-512 prime with a key space of 2^{256} vectors is almost always included, and for this size the lowest multiplication count we have found in the literature is 789000: this is from [ACR23, Table 1, "hvelu", "OAYT-style"], which reports $624000\mathbf{M} + 165000\mathbf{S} + 893000\mathbf{a}$. All Skylake cycle counts we have found are above 200 million. The `high-ctidh` speeds are much faster, and have the added feature of constant-time verification using `valgrind`.

Sometimes the latest software speeds are not fully reflected in the relevant papers, so we downloaded the latest versions of `csidh_withstrategies` (dated July 2020) from [CR22] and `sqale-csidh-velusqrt` (dated December 2020) from [CCJR22], and ran their benchmarking tools—with one modification to change the number of experiments ("`its`") from 1024 to 65536—on the same Skylake machine that we had used for measuring `high-ctidh`. Some care is required in comparisons, for at least three reasons: first, some tools report the time for an action *plus* key validation; second, different benchmarking frameworks could be measuring different things (e.g., our impression is that the costs of Elligator were omitted from the multiplication counts reported in [ACR23]); third, the 512-bit parameters in `sqale-csidh-velusqrt` use a key space of size

only 2^{220}, as noted above. Note that for CSIDH-1024 there is even more variation in the literature in the size of key space; e.g., the original CSIDH-1024 software from [CLM+18] used $5^{130} > 2^{300}$ keys.

The `csidh_withstrategies` tools, using parameters `BITLENGTH_OF_P=512 TYPE=WITHDUMMY_2 APPROACH=STRATEGY`, reported averages of 218.42 million clock cycles (standard deviation $\sigma = 3.39$ million), 691231**a** ($\sigma = 12554$), 189377**S** ($\sigma = 4450$), and 665876**M** ($\sigma = 7888$); in other words, 855253 multiplications, or 851939 counting $(\mathbf{M}, \mathbf{S}, \mathbf{a}) = (1, 0.8, 0.05)$.

The `sqale-csidh-velusqrt` tools, using `BITS=512 STYLE=wd2`, reported averages of 190.921 million cycles ($\sigma = 4.32$ million), 626000**a** ($\sigma = 13000$), 128000**S** ($\sigma = 5000$), and 447000**M** ($\sigma = 9000$); i.e., 575000 multiplications. For comparison, `high-ctidh` takes 89.11 million cycles (310945 multiplications) as noted above, plus 4.09 million cycles for validation.

Finally, Table 6.2 summarizes the measurements listed above for `high-ctidh`, for the software from [CR22], and for the software from [CCJR22]; the measurements stated in [OAYT19, CCC+19, HLKA20, ACR23] for the software in those papers; and the measurements in [CCC+19] for the software in [MCR19]. For [HLKA20] the reported processor is an Intel Core i7-7500k, which is Kaby Lake rather than Skylake, but Kaby Lake cycle counts generally match Skylake cycle counts. The table omits cycle counts for [ACR23], which used Python, and [OAYT19], which used C but had measurements affected by an unknown amount of Turbo Boost.

pub	priv	DH	Mcyc	**M**	**S**	**a**	$1,1,0$	$1,0.8,0.05$	
512	220	1	89.11	228780	82165	346798	310945	311852	new
512	220	1	190.92	447000	128000	626000	575000	580700	[CCJR22]
512	220	2	93.23	238538	87154	361964	325692	326359	new
512	256	1	125.53	321207	116798	482311	438006	438762	new
512	256	1	—	624000	165000	893000	789000	800650	[ACR23]
512	256	2	129.64	330966	121787	497476	452752	453269	new
512	256	2	218.42	665876	189377	691231	855253	851939	[CR22]
512	256	2	238.51	632444	209310	704576	841754	835121	[HLKA20]
512	256	2	239.00	657000	210000	691000	867000	859550	[CCC+19]
512	256	2	—	732966	243838	680801	976804	962076	[OAYT19]
512	256	2	395.00	1054000	410000	1053000	1464000	1434650	[MCR19]
1024	256	1	469.52	287739	87944	486764	375683	382432	new
1024	256	1	—	552000	133000	924000	685000	704600	[ACR23]
1024	256	2	511.19	310154	99371	521400	409525	415721	new

Table 6.2: Comparison of speed reports for constant-time CSIDH actions. The CSIDH size is specified by "pub" (512 for the CSIDH-512 prime, 1024 for the CSIDH-1024 prime) and "priv" (k where private keys are chosen from a space of approximately 2^k vectors). "DH" is the Diffie–Hellman stage: "1" for computing a public key (computing the CSIDH action), "2" for computing a shared secret (validating a public key and then computing the CSIDH action). "Mcyc" is millions of Skylake cycles (not shown for Python software); "**M**" is the number of multiplications not including squarings; "**S**" is the number of squarings; "**a**" is the number of additions including subtractions; "$1,1,0$" and "$1,0.8,0.05$" are combinations of **M**, **S**, and **a**. See text for measurement details and standard deviations.

Chapter 7

Disorientation faults in CSIDH

This chapter is based on the paper *Disorientation faults in CSIDH* [BKL+23], joint work with Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, and Monika Trimoska.

The introduction has been modified, and the background section is omitted as the relevant details have been treated in Chapters 2 and 4. The remaining sections have been edited for style and typographical consistency.

7.1 Introduction

Recall that in CSIDH, the secrets are the number and orientation of isogeny steps we need to perform. Thus, the control flow of a straightforward implementation is directly related to the secret key, which complicates side-channel resistant implementations [BLMP19, MR18, MCR19, HLKA20, BBC+21, CMRS22].

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 alters the *orientation* of a point used during the computation of $\mathfrak{a} \star E_0$. 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, *faulty* curve.

Any faulty curve differs from the correct public key by isogeny steps that depend on the *secret key*. Therefore, by finding paths between faulty curves and the public key leaks information on the key, and having enough faulty curves allows us to recover the full secret key.

To simplify exposition in Section 7.4, we first assume access to a device (e.g., a hardware security module providing a CSIDH accelerator) that applies a secret key to a given public key (i.e., computing the shared key in CSIDH) and returns the result. We also discuss in Section 7.7 a *hashed* version of the attack 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 (message authentication code) 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` (Section 7.6). 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. We also use the *quadratic twisting* trick to reduce this cost in Section 7.8.

Our attack is general enough that it applies to most implementations of CSIDH, and we give details in Section 7.5. We also provide a set of general lightweight *countermeasures* in Section 7.9.

7.2 Fault attacks

Fault attacks. In a side-channel attack, an attacker uses passive observations of physical leakage (such as timing differences, electromagnetic emissions, or power consumption) during the computation to infer secret information. Another class of physical attacks are *fault-injection (fault) attacks*: By actively manipulating the execution environment of a secure device (e.g. 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 now incorrect, *faulty* outputs.

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 (especially in a cryptographic context) may lead to exploitable behavior. In practice, the difficulty of injecting a particular kind of fault (or combination of multiple faults) depends on various parameters; generally speaking, less targeted faults are easier.

Fault attacks on isogeny schemes. Prior works investigating fault attacks on isogeny-based cryptography mostly target specific variants or implementations of schemes. *Loop-abort* faults on the SIDH cryptosystem [GW17], discussed for CSIDH in [CKM+20], lead to leakage of an intermediate value of the computation rather than the final result. Replacing torsion points with other points in SIDH [TDEP21, Ti17] can be used to recover the secret keys; faulting intermediate curves in SIDH [ACMR22] to learn if secret isogeny paths lead over subfield curves can also leak information on secret keys. But the two latter attacks cannot be mounted against CSIDH due to the structural and mathematical differences between SIDH and CSIDH.

Specifically for CSIDH, one can modify memory locations and observe if this changes the resulting shared secret [CKM21]. A different attack avenue is to target dummy computations in CSIDH [CKM+20, LH21].

7.3 Attack scenario and fault model

Throughout, 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 input an elliptic curve E_A and receive back the result $E_B = \mathfrak{a} \star E_A$ (the result of applying $\mathfrak{a}$ to the public key E_A as in the second step of the key exchange). For concreteness, we mostly refer to the computation of the CSIDH group action as described in Algorithm 1.

We use the notation of Section 4.2, and all the elliptic curves we encounter are assumed to be *CSIDH curves*: supersingular elliptic curves over $\mathbb{F}_p$ in Montgomery form.

Remark 7.3.1. 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 7.4 will increase significantly. To simplify exposition, we postpone this discussion to Section 7.7.

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 7.4.

Square check. Recall that the orientation of a point P is computed in Algorithm 1 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.

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 (i.e. the shared key) so can discard those outputs and continue with those where the orientation has been flipped.

Remark 7.3.2. There are other ways to flip the orientation s. For example, one can inject a fault into x after s has been computed. The analysis and attack of Sections 7.4 and 7.5 apply to all possible ways to flip s, independent of the actual fault injection. The countermeasures introduced in Section 7.9 prevent all possible ways to flip s that we know of.

In general, faulting the Legendre symbol computation in `IsSquare` leads to a random $\mathbb{F}_p$-value as output instead of ± 1. The interpretation of this result depends on the implementation. E.g., the CSIDH implementation from [CLM+18] interprets the output as boolean value by setting $s = 1$ if the result is $+1$, and -1 otherwise. In this case, faults mostly flip from positive to negative orientation. Thus, faulting the computation of z is superior in our attack setting.

Elligator. On input of a random value, Elligator 2 [BHKL13] 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 $P_- \leftarrow P$. A fault to this `IsSquare` check flips the assignments to P_+ and P_-; hence, the orientation of *both* points is flipped.

So all steps computed using either of these points are in the wrong direction. A notable exception is CTIDH, which uses two independent calls to Elligator to produce points. Our attack nonetheless extends to this case (Section 7.5.2).

7.4 Exploiting orientation flips

In Section 7.3, we defined an attack scenario that allows us to flip the orientation s in Algorithm 1 in Line 3, *without* changing the point P. Then, in Line 4, the

incorrect set S' is chosen: steps with direction s, which is the *opposite* direction to the point P. This means that all the steps in S' will be performed in the opposite direction. It is more convenient to select the set S first and talk about the orientation of the point, or just the point P being flipped.

We assume that we can always successfully flip the orientation in a specific round r, while the rest of the computation is performed correctly. We also assume access to the result of the faulty evaluation, which is a *faulty curve* $E_t \neq \mathfrak{a} \star E_A$.

Organization of the section. We first study the effect of orientation flips for full-order points in Sections 7.4.1 and 7.4.2, and then discuss the effects of failing to generate points of full order in Sections 7.4.3 and 7.4.4. We organize the faulty curves into components according to their orientation and round in Section 7.4.5 and study the distance of components from different rounds in Section 7.4.6. In Section 7.4.7, we combine the information to recover the secret key $\mathfrak{a}$.

7.4.1 Implications of flipping the orientation of a point

Assume first that all points have full order, so no steps are skipped in Line 7.

In the round we fault, we are evaluating the group action by an element $\prod_{i \in S} \mathfrak{l}_i$. Suppose we generate a negatively oriented point P, but flip its orientation. This does not change the point (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: the action by $\prod_{i \in S} \mathfrak{l}_i^{-1}$ instead. Similarly for flipping positive orientation to negative.

If the rest of the computation is performed correctly and we only faulted the computation of the steps in S with direction s, then the resulting curve differs from the correct output E_B by twice as many steps in the opposite direction $-s$ (if we step left by accident, we need to take two steps right to correct for it). We call S the *missing set* of E_t, and the faulted curve satisfies

$$E_t = \prod_{i \in S} \mathfrak{l}_i^{-2s} \star E_B.$$

Definition 7.4.1. We define the *distance* between E and E' to be the minimal size $d = |S|$ of $S \subset \{1, \ldots, n\}$ such that $E' = \prod_{i \in S} \mathfrak{l}_i^{\pm 2} \star E$ (and, for completeness, set distance to ∞ if no such S exists).

If E_t is a faulty curve, then the distance E_t and E_B counts the number of flipped isogeny steps.

Positive and negative primes. Suppose $\mathfrak{a}$ is given by the exponent vector (e_i). We define the set of *positive* primes $L_+ := \{i \mid e_i > 0\}$ to be the set of positive isogeny steps, *negative* primes $L_- := \{i \mid e_i < 0\}$ to be the set of negative isogeny

steps, and neutral primes $L_0 := \{i \mid e_i = 0\}$. In 1-point strategies, in each round we compute steps with the same orientation, so any missing set S satisfies $S \subset L_+$ or $S \subset L_-$. However, in 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 7.4.2. Assume we flip the orientation $s \mapsto -s$ of the first point P in Algorithm 1. Then S contains exactly those i such that $|e_i| \geq 1$ and $\text{sign}(e_i) = -s$. Therefore, we have $S = L_{-s}$.

7.4.2 Faulty curves and full-order points

We continue to assume that all points have full order and analyze which faulty curves we obtain by flipping the orientation in round r.

When using 1-point strategies (cf. Section 4.3.3), the computation in round r depends on the previous rounds. This is because we sample 1 point per round, and only perform isogenies in the direction of that point. So the set S in round r depends on the previous rounds with *the same orientation*.

We will call a round *positive* (resp. *negative*) if steps were performed for *positive* (resp. *negative*) primes. Note that if the round was faulted, the steps were performed in the opposite direction, if not, the steps were computed correctly.

In a 2-point strategy, we sample points of both orientations and compute isogenies in both directions. If the points have 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.

For 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 the positive direction.

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

$-(-)$. Two negative rounds, the second negative 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 the same as $(-)$ and $++(-)$: in all three, 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 7.4.3 (CSIDH)**.** Suppose we use 4 primes $L = \{3, 5, 7, 11\}$ and a secret key $(1, -2, -1, 3)$. 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.

Definition 7.4.4. Let $E^{r,+}$ be the faulty curve produced in by a sequence of rounds $+\cdots+(+)$ 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, the last sign in the sequence must be $(+)$, and the sequence must contain a total of r pluses.

Lemma 7.4.1. *Assume we use a 1-point strategy. The probability to get an effective round-r sample if we successfully flip in round n is equal to* $\binom{n-1}{r-1} \cdot \frac{1}{2^{n-1}}$.

Definition 7.4.5. We define the set $S^{r,s}$ as the missing set of the effective round-r curve with orientation s, i.e., $E_B = \prod_{i \in S^{r,s}} \mathfrak{l}_i^{2s} \star E^{r,s}$.

For example in CSIDH, the sets $S^{1,\pm}$ were already discussed in Example 7.4.2 and in general, $S^{r,+} = \{i \mid e_i \geq r\}$ and $S^{r,-} = \{i \mid e_i \leq -r\}$.

7.4.3 Missing torsion

In Section 7.4.2, 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 had full order (and hence all isogenies were computed) up until round r, but the point P generated in round r potentially has smaller order. We call this the *missing torsion* case. The remaining case of non-full order points in earlier rounds will be concluded in Section 7.4.4.

For concreteness, we assume that we are computing the CSIDH group action using Algorithm 1 and derive concrete expressions for the probability of obtaining specific faulty curves. For other implementations, the analysis is analogous.

Missing torsion. Let P be a point of order $N \mid p+1$. We say that P is *missing* ℓ (in its order) if $\ell \nmid N$. The *missing torsion* of a point P is the collection of primes that are missing from its order. As such, it describes which isogeny steps cannot be computed from (suitable powers, or isogeny images of) P.

We say that the computation of an isogeny step *failed* in a given round if the computation of that step was skipped, e.g. in Line 7 in Algorithm 1.

Round-r faulty curves. For simplicity, assume that we are in round r, in the case $+\cdots+(+)$, and that none of the isogeny computations in the previous rounds failed. In round r, we want to compute the positive steps, sampling a point P. (As this is the faulted round, P is a negative point.)

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 $S_t \subset S^{r,+}$ 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 $E_B = \prod_{i \in S_t} \mathfrak{l}_i^2 \star E_t$. We see that S_t differs frm $S^{r,+}$ exactly by the primes in $S^{r,+}$ missing in the order of P.

Assume we repeat this scenario in T runs, leading to different sampled points P and possibly different faulty curves E_t. Let $n(E_t)$ be the number of times the curve E_t occurs among the T samples. As P_t is a randomly sampled point, its order is divisible by ℓ_i with probability $\frac{\ell_i - 1}{\ell_i}$. Same analysis works for negative rounds. So we can calculate the probability we obtain a given E_t.

Proposition 7.4.2. *Let P_t be a random point of orientation s. The probability that we obtain a curve $E_t = \prod_{i \in S_t} \mathfrak{l}_i^{-2s} \star E_B$ is exactly*

$$p_t = \prod_{i \in S_t} \frac{\ell_i - 1}{\ell_i} \cdot \prod_{i \in S^{r,s} \setminus S_t} \frac{1}{\ell_i}. \tag{7.1}$$

Proof. The probability of obtaining E_t is equal to the probability that the order of the point P_t is divisible by all the primes in S_t and not divisible by all the primes in $S^{r,s} \setminus S_t$. The first happens with probability $\prod_{i \in S_t} \frac{\ell_i - 1}{\ell_i}$; the second is an independent event happening with probability $\prod_{i \in S^{r,s} \setminus S_t} \frac{1}{\ell}$. □

The expected number of appearances $n(E_t)$ of E_t in T runs is $n(E_t) \approx p_t \cdot T$. Since $\frac{\ell_i - 1}{\ell_i} \geq \frac{1}{\ell_i}$ for all ℓ_i, the probability p_t is maximal when $S_t = S^{r,s}$. We denote this probability by $p^{r,s}$. Hence, the curve that is likely to appear the most in this scenario over enough samples, is the curve $E^{r,s}$ which we defined as precisely that curve with missing set $S^{r,s}$.

Corollary 7.4.3. *The curves $E^{r,+}$ 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 7.4.6 (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 approximated 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 1. For instance, if e_1 is positive, then $p^{1,-}$ is larger than $p^{1,+}$ and 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 $n(E^{1,+})$; the 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 $p_t := \frac{1}{\ell_1} \cdot \frac{\ell_1}{\ell_1 - 1} \cdot p^{1,+} = \frac{1}{2} \cdot p^{1,+}$.

We can generalize Example 7.4.6 for any two faulty curves E_t and $E_{t'}$ that are effective round-r samples of the same orientation:

Corollary 7.4.4. *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 $E_t = \left(\prod_{i \in S_{t'} \setminus S_t} \mathfrak{l}_i^{2s} \cdot \prod_{i \in S_t \setminus S_{t'}} \mathfrak{l}_i^{-2s}\right) \star E_{t'}$. In particular, any effective round-r curve E_t with orientation s is close to $E^{r,s}$: since $S_t \subset S^{r,s}$, S_Δ is small.*

Example 7.4.7. Consider CSIDH with primes $L = \{3, 5, \dots\}$. Assume that in one faulted run, the first positive point P_{t_1} misses 3-torison, while in another run the first positive point P_{t_2} misses 5-torsion. The faulty curves E_{t_1} and E_{t_2} differ from $E^{1,+}$ by two 3-isogenies and two 5-isogenies, respectively, and are at distance 2 from each other. 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 $e_2 \geq 1$.

7.4.4 Torsion noise

Orthogonally to Section 7.4.3, we now examine the case that missing torsion occurred in an earlier round than the round we are faulting.

Example 7.4.8 (CSIDH)**.** Suppose that $e_1 = 1$ and that in the first positive round, the point generated in Line 2 of Algorithm 1 had order not divisible by ℓ_1, but all other points have full order. Thus, the ℓ_1-isogeny attempt fails in the first positive step. Consider now the second positive round. From Section 7.4.2, we would expect to be computing steps in $S^{2,+} = \{i \mid e_i \geq 2\}$. Since no ℓ_1-isogeny was computed in the first round, 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, $E_t = \mathfrak{l}_1^{-2} \star E^{2,+}$. Unlike the faulty curves from Section 7.4.3, the positively oriented isogeny goes from E_t *towards* $E^{2,+}$. Note that if we had $e_1 = 2$, 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 call the phenomenon observed in Example 7.4.8 *torsion noise*. Torsion noise happens when we fault the computation in round r and flip the orientation of an ℓ_i-isogeny for which $|e_i| < r$.

Torsion noise is rarer than missing torsion but can still happen: 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 unlikely 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 7.4.3, 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.

Remark 7.4.9 (Orientation of torsion noise)**.** 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.

7.4.5 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 7.4.4, we show that such curves are close to each other. In particular, for most curves E_t, the path from E_t to $E^{r,s}$ uses only degrees contained in the set $S^{r,s}$, and the only exception comes from torsion noise. Finding short paths among faulty curves gives us information about $S^{r,s}$, and hence about the secret key.

Definition 7.4.10. Let $\{E_t\}$ be a set of of effective round-r faulty curves with orientation s. The *component* $G^{r,s}$ is a graph with vertices $\{E_t\}$. There is and edge between two curves if they have distance 1. We label the edges by the ℓ_i corresponding to the two ℓ_i-isogeny steps.

We sparsify the graph $G^{r,s}$ and think of it as a tree with $E^{r,s}$ as the *root*. If we do not have enough faulty curves $\{E_t\}$, it may not be possible to connect all the curves with only one double-step. For convenience, we assume that we have enough curves. In practice, we find short paths from E_t to $E^{r,s}$ and include in the graph $G^{r,s}$ all the intermediate curves.

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 $E^{r,s}$ as the most frequent faulty curves (Corollary 7.4.3). The distance from the root to any round-r faulty curve with the same orientation is small (cf. Corollary 7.4.4). Therefore, we can find the edges by applying short walks in the isogeny graph.

Secret information. The edges of $G^{r,s}$ give information on $S^{r,s}$: An effective round-r faulty curve E_t with torsion set $S_t \subset S^{r,s}$ is connected to the root by a path with labels $S^{r,s} \setminus S_t$. Therefore, any labels in in the graph $G^{r,s}$ determine primes with orientation s. The direction of the edge matters: the orientation of $E^{r,+} \to E_t$ is positive; and that of $E^{r,-} \to E_t$ is negative. Torsion noise has precisely the opposite direction of the edges (see Remark 7.4.9), and so any such label ℓ_i is included in $S^{r',s}$ for some $r' < r$.

Sorting round-r samples. Suppose we have a set of round-r faulty curves $\{E_t\}$, but we do not have information about the orientation yet. We can again use Corollary 7.4.3 to find the root of the graph; then we take small isogeny steps until 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 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 $S^{r,-}$. Figure 7.1 summarizes the process, where, e.g., $E^{r,+} \to E_7$ shows missing torsion and $E_8 \to E^{r,+}$ is an example of torsion noise.

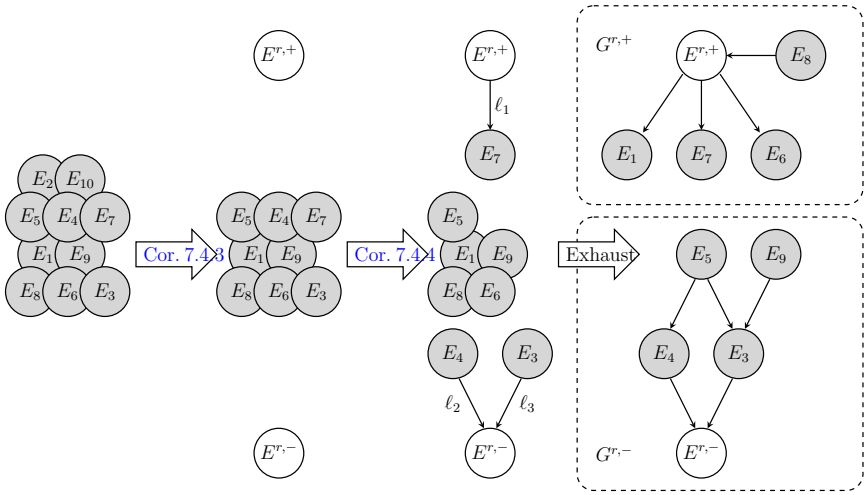


Figure 7.1: Building up the component graphs of faulty curves.

7.4.6 Connecting the components $G^{r,s}$

Now, we explain how to connect the components $G^{r,s}$ for different rounds r. The distance of these components is related to the sets $S^{r,\pm}$. We show that it is

computationally feasible to connect the components via a meet-in-the-middle approach. Connecting two components reveals significant knowledge on the sets $S^{r,+}$ and $S^{r,-}$, and connecting all components is enough to infer $\mathfrak{a}$ (Section 7.4.7).

Notation. We defined the distance of two faulty curves in Definition 7.4.1. The distance between the graphs $G^{r,s}$ and $G^{r',s'}$ is the minimum of distances of $E_t \in G^{r,s}$ and $E_{t'} \in G^{r',s'}$. This distance is always at most the distance of the root curves $E^{r,s}$ and $E^{r',s'}$.

We will try to find isogeny paths between two components via meet-in-the-middle approach (more details in Section 7.6). Roughly, we start performing short sequences of isogeny steps (two steps per degree) from each component, until we reach a collision. We call *support* the set of isogeny degres we use in this neighborhood search. The MITM search is infeasible for large distances and so the support will be changed adaptively. If we decide that a prime ℓ_i cannot be used in a path connecting G and G' (for instance, because of orientation, or because it is connecting other components), we *filter* ℓ_i out of the support.

Connecting two components. We start with the usual example of CSIDH, computed using Algorithm 1. The difference between the sets $S^{1,+}$ and $S^{2,+}$ is precisely $S_\Delta = \{i : e_i = 1\}$. So, we have $E^{2,+} = \prod_{i \in S_\Delta} \mathfrak{l}_i^2 \star E^{1,+}$. Clearly this gives a path between the components $G^{1,+}$ and $G^{2,+}$ (for illustration, see the example of CSIDH-512 in Figure 7.2).

In the general case, if we find an isogeny between two components, sa $G^{r,+}$ and $G^{r',+}$, we can compute the isogeny between the two roots $E^{r,+}$ and $E^{r',+}$. The degree of this isogeny $E^{r,+} \to E^{r',+}$ describes precisely the *difference* between the sets $S^{r,+}$ and $S^{r',+}$. In other CSIDH-variants, such sets are not necessarily nested, but connecting all components still reveals e_i as Section 7.4.7 will show. In general, we connect two subgraphs by a distributed meet-in-the-middle search which finds the shortest connection first.

Distance between connected components. The distance between any two components $G^{r,+}$ and $G^{r',+}$ is equivalent to finding the set difference of $S^{r,+}$ and $S^{r',+}$. It depends heavily on the implementation, as these sets are determined by the key $\mathfrak{a}$ and the evaluation of this key. In CSIDH-512, the difference between $S^{r,+}$ and $S^{(r+1),+}$ is the set of indices with $e_i = r$, which on average is of size $\frac{74}{11} \approx 6.7$. In practice, this distance roughly varies between 0 and 15. Note that upon finding any connection, we reduce the support for further connection finding, so it is possible to find connections between components with larger distances as well. See Section 7.6 for more details on how we connect these components in practice.

7.4.7 Revealing the private key

So far, we showed 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 7.4: 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 together with the curve E_B, we learn the difference between the sets $S^{r,+}$ and $S^{(r+1),+}$ for all rounds r (as well as for $S^{r,-}$ and $S^{(r+1),-}$). A single isogeny from some $G^{r,+}$ to $E_B = \mathfrak{a} \star E_A$ then recovers $S^{r,+}$ for this round r: Such an isogeny gives us an isogeny from $E^{r,+} = \prod_{i \in S^{r,+}} \mathfrak{l}_i^{-2} \star E_B$ to E_B, whose degree shows us exactly those $\ell_i \in S^{r,+}$. From a connection between the components $G^{r,+}$ and $G^{r',+}$, we learn the difference in sets $S^{r,+}$ and $S^{r',+}$. From $S^{r,+}$, we can then deduce $S^{r',+}$. Therefore, if all graphs $G^{r,+}$ for different r are connected, and we have at least one isogeny from a node to E_B, we learn the sets $S^{r,+}$ for all rounds r (and equivalently for $S^{r,-}$). From the knowledge of all sets $S^{r,+}$ and $S^{r,-}$ 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 *correct* 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 set $\tilde{S}^{r,+}$ derived from our attack (leading to some $\mathfrak{a}'$) always have a small distance. A simple meet-in-the-middle search between $\mathfrak{a}' \star E_A$ and $\mathfrak{a} \star E_A$ then quickly reveals the errors caused by missing torsion and torsion noise.

7.4.8 Complexity of recovering the secret $\mathfrak{a}$

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

1. Gather enough effective round-r samples E_t per round r, using Lemma 7.4.1.
2. Build up the components $G^{r,+}$ and $G^{r,-}$ using Corollaries 7.4.3 and 7.4.4.
3. Connect components to learn the difference in sets $S^{r,+}$ and $S^{r',+}$.
4. Compute the sets $S^{r,+}$ and $S^{r,-}$ for every round and recover $\mathfrak{a}$.

The overall complexity depends on the number of samples per round, but is in general dominated by Step 3. For Step 2, nodes are in most cases relatively close to the root $E^{r,+}$ or to an already connected node E_t, as shown in Corollary 7.4.4.

For Step 3, components are usually further apart than nodes from Step 2. In general, the distance between components $G^{r,+}$ and $G^{r',+}$ depends heavily on the specific design choices of an implementation. In a usual meet-in-the-middle

approach, where n is size of the support and d is the distance between $G^{r,+}$ and $G^{r',+}$, the complexity of finding a connection is $\mathcal{O}(\binom{n}{d/2})$ isogeny-step computations. 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 7.5.

7.5 Case studies: CSIDH and CTIDH

We described the general strategy in four steps in Section 7.4.8, but the execution depends on the actual implementation. We discuss the case of CSIDH-512 in Section 7.5.1, CTIDH-512 in Section 7.5.2, and we analyze other implementations in Section 7.5.3.

For concreteness, we assume we are faulting the computation of $E_B = \mathfrak{a} \star E_0$.

7.5.1 Breaking CSIDH-512

We recalled the parameters of CSIDH-512 in Example 4.2.4. It uses 74 primes ℓ_i, and the secret keys are sampled from $[-5, 5]^{74}$. For any $k \in [-5, 5]$, we expect about $\frac{1}{11} \cdot 74$ exponents $e_i = k$; this count obeys a binomial distribution with parameters $(74, 1/11)$. We expect to see about $\frac{5}{11} \cdot 74 \approx 33.6$ positive and negative primes each, and about $\frac{1}{11} \cdot 74 \approx 6.7$ neutral primes. The group action is evaluated as in Algorithm 1, using a 1-point strategy. Upon sampling a point with orientation s, we set $S = \{i \mid e_i \neq 0,\ \mathrm{sign}(e_i) = s\}$.

Now, we specialize the four steps to secret-key recover defined in Section 7.4.8.

Building components $G^{r,+}$ and $G^{r,-}$. Step 2 of the attack works exactly as described in Section 7.4.5. If E_t and $E_{t'}$ are effective samples from the same round with the same orientation, their distance is small (Corollary 7.4.4). Using round information and frequency of faulty curves, we identify the 10 *root* curves $E^{r,\pm}$ for $r = 1, \ldots, 5$, and explore all paths of small length from those 10 curves, or grow the neighborhoods via a meet-in-the-middle approach, e.g., using `pubcrawl` (Section 7.6). We perform this neighborhood search on all of the sampled curves until we have 10 connected graphs $G^{r,\pm}$ for $r \in \{1, \ldots, 5\}$, as in Figure 7.1. This step is almost effortless: most curves are distance 1 or 2 away from the root $E^{r,s}$.

The degrees of the isogenies corresponding to the new edges in $G^{r,\pm}$ reveal information on the sets $S^{r,\pm}$, in particular on the orientation of primes, which can be used to reduce the search space when connecting the components $G^{r,\pm}$.

Filter-and-break it, until you make it. Step 3 is the most computationally intensive step, as it connects the 10 components $G^{r,\pm}$ and E_B into a single large connected component. We argue that it is practical for CSIDH-512.

For this, we examine the gaps between $G^{r,s}$ and $G^{r+1,s}$ (setting $G^{6,\pm} = E_B$) as in Figure 7.2. Clearly they are given by $S^{r,s} \setminus S^{r+1,s}$ (note that $S^{6,\pm} = \emptyset$). Since there are 10 gaps and 74 primes, at least one pair $E^{r,s}$ and $E^{r+1,s}$ has distance at most 7, and the shortest paths between components are at most that long. As we expect about 7 exponent $e_i = 0$, we expect to find curves of distance at most 6. Such gaps are easily found using a meet-in-the-middle search, see Section 7.6.

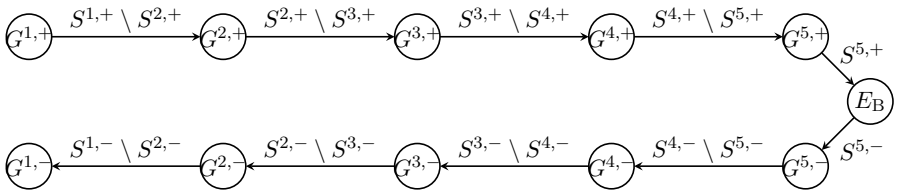


Figure 7.2: Large connected component associated to an attack on CSIDH-512.

We start by learning the orientation of the components. First we identify $G^{1,\pm}$ and look at 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, and all components with a matching label are also positive.

Next, all the labels that appear as degrees of edges in positive components $G^{r,+}$ for any r are necessarily positive. Finally, positively oriented components can only be connected by positive primes, so we can remove from the support all the primes that we know are negative. Similarly for negative orientations.

Then we start the meet-in-the-middle search connecting components with the same orientation. Any label i appears in at most *one* connection, so every time we find a connection, we filter the connecting primes from the support. This reduces the support and we can perform the MITM search for larger distances.

Recovering the secret key. From the connected components, we recover all of the sets $S^{r,\pm}$ and we compute the secret key as described in Section 7.4.7.

Example 7.5.1 (Toy CSIDH-103)**.** Figure 7.3 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 $\{-3, \dots, +3\}^n$. 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

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

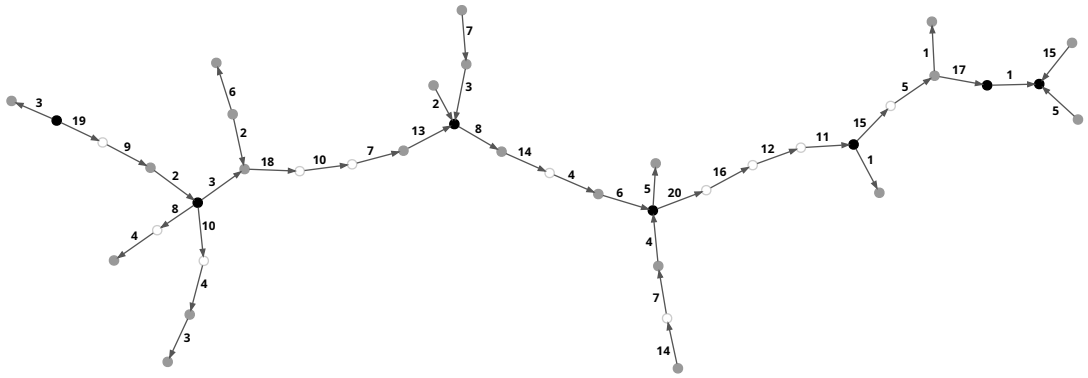


Figure 7.3: Example isogeny graph of faulty curves obtained from attacking the fictitious CSIDH-103 implementation from Example 7.5.1. An edge labeled i denotes the isogeny step $\mathfrak{l}_i$. The E_B curve and the root faulty curves $E^{r,s}$ are rendered in black (from left to right: $E^{1,+}$, $E^{2,+}$, $E^{3,+}$, E_B, $E^{3,-}$, $E^{2,-}$, $E^{1,-}$), other faulty curves appearing in the dataset are gray, and white circles are "intermediate" curves discovered while connecting the components. The primes appearing on the connecting path between $E^{i,\pm}$ and $E^{i+1,\pm}$ are exactly the primes appearing i times with orientation $\pm$. For example, the primes indexed by 2, 9, 19 appearing between $E^{1,+}$ and $E^{2,+}$ have exponent $+1$ in the secret key.

Required number of samples. Recovering the full secret exponent vector in CSIDH-512 equates to computing the sets $S^{r,+}$ and $S^{r,-}$ for $r \in \{1, \ldots, 5\}$. Recall that to compute these sets we need to build a connected component including subcomponents $G^{r,+}$ and $G^{r,-}$ for $r \in \{1, \ldots, 5\}$, and E_B (the one-node-graph consisting of just the public key). We build the components $G^{r,+}$ and $G^{r,-}$ by acquiring enough effective round-r samples.

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 $T_5 = 128$, so that $T = 256$. From Lemma 7.4.1, we then 4 round-5 samples per orientation and the probability that we do not get any of the elements of $G^{5,\pm}$ is about 1.7%.

This strategy can be improved upon. Notice that we 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 such samples. But gathering samples from round 5 already gives us many samples from rounds before. Using Lemma 7.4.1 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 benefits that all faulty curves are

effective round-1 curves, making them easy to detect in later rounds; that no torsion noise appears; and that missing torsion quickly allows to determine the orientation of the small primes, reducing the search space for connecting the components. Finally, note that gathering T faulty samples requires approximately $2T$ fault injections, since, half of the faults are expected to flip the orientation.

7.5.2 Breaking CTIDH-512

CTIDH-512 [BBC+21] uses 14 batches with bounds $m_i \leq 18$, requiring at least 18 rounds (see Example 6.7.1). Due to some computations failing, in practice we may need up to around 22 rounds. In every round, we compute one isogeny per batch; using a 2-point strategy, we compute isogenies in both positive and negative direction. So, all round-r samples are effective round-r samples.

CTIDH uses two independent calls to Elligator-2 map to sample two points of opposite orientation. We will always assume that we inject a fault into only one of these two Elligator calls (as in Section 7.3). Hence, 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 $(2, -4)$, 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 can visualize this as a queue $[3+, 3+, 5-, 5-, 5-, 5-]$ (padded on the right with dummy isogenies for the remaining rounds up to m_1).

Therefore, the sets $S^{r,\pm}$ contain precisely the r-th prime in the queue for the batch $\mathcal{B}_i$. With 14 batches and an equal chance for either orientation, we expect that each $S^{r,\pm}$ will contain about 7 primes, and only one prime per batch.

The small number of batches and the ordering of primes within the batches make CTIDH especially easy to break using our disorientation attack.

Components for CTIDH-512. We again construct the graphs $G^{r,s}$ from enough samples. In CTIDH, the failure probability of each isogeny computation is constant per batch and equal to the probability of failure for the *smallest* prime in the batch (to hide which $\ell_{i,j}$ step was computed); this slightly increases the chance of missing torsion and torsion noise.

The distance of the root curves $E^{r,s}$ to the curve E_B is bounded by the number of batches. Per round r, the sum of the distances of $E^{r,\pm}$ to E_B is at most 14, so we again expeect the distance to be about 7.

The distance between two graphs $G^{r,s}$ and $G^{(r+1),s}$ is often much smaller. 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 we expect significant overlap: The set difference contains the 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 can be used to quickly find other sets $S^{r',+}$. 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 $E^{r,\pm}$ (or components $G^{r,\pm}$) of distance 4. This connects E_B to all the curves at distance at most 8. Disregarding orientation and information on batches, if we have N curves that we want to connect, the naive search requires about $2 \cdot \binom{74}{4} \cdot N \approx 2^{21} \cdot N$ isogenies. The actual search space is much smaller as we never use two primes from the same batch.

Moreover, isogenies in batches are in ascending order. So, if in round r we see that the 3rd prime from batch $\mathcal{B}_i$ was used, none of the rounds $r' > r$ involves the first two prime, and none of the rounds $r' < r$ can use the later primes from the batch for that direction. Late rounds typically contain many dummy isogenies and the faulty curves are especially close to the public key. We expect to rapidly recover $S^{r,\pm}$ for the late round curves.

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. Assume we perform c successful faults per round per Elligator call, expecting to get samples for both orientations per round. As CTIDH-512 performs 18 rounds (in practice typically up to 22 because of isogeny steps failing), we require $T = 18 \cdot 2 \cdot c$ successful flips. It seems possible to take $c = 1$ and hence $T = 36$ (or up to $T = 44$) samples.

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. Let $\tilde{S}^{r,s}$ denote the index set recovered for round r and sign s. We can correct for some of these errors, looking at $\tilde{S}^{r',\pm}$ for rounds r' close to r. Consider only primes from the same batch $\mathcal{B}$, then the following can happen:

- *No* prime from $\mathcal{B}$ is contained in either $\tilde{S}^{r,+}$ or $\tilde{S}^{r,-}$: all primes from $\mathcal{B}$ are done or *missing torsion* must have happened. We can examine the primes from the batch $\mathcal{B}$ which occur in neighboring rounds $\tilde{S}^{(r\pm1),\pm}$ and use the

ordering in the batch to obtain guesses on which steps should have been computed if any.

- *One* prime from $\mathcal{B}$ is contained in $\tilde{S}^{r,+} \cup \tilde{S}^{r,-}$: we fix no errors.

- *Two* primes from $\mathcal{B}$ are contained in $\tilde{S}^{r,+} \cup \tilde{S}^{r,-}$: the smaller one must have come from torsion noise in a previous round and can be removed.

Remark 7.5.2. 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, we feel that already two successful faults per round are low enough.

Even 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 reduce the search space for the secret key significantly. One may also select the rounds to attack adaptively, based on the information recovered from $S^{r,s}$.

7.5.3 Other variants of CSIDH

Finally, we discuss some of the other implementations of CSIDH. We discuss SIMBA [MCR19], dummy-free implementations [CCC+19, CR22, ACR23], and SQALE [CCJR22]. All of these use `IsSquare` checks for point sampling and are vulnerable to our attack.

SIMBA. Implementations using SIMBA [MCR19] can be attacked similarly to CSIDH (cf. Section 7.5.1). SIMBA-M divides the primes into M *prides* (batches), so in each round we only compute $\lceil n/M \rceil$. So, as in CTIDH, fewer isogenies are computed per round, and the distances between the components $G^{r,s}$ are smaller.

Dummy-free CSIDH. The dummy-free implementations [CCC+19, CR22, ACR23] replace pairs of dummy ℓ_i-isogenies by pairs of isogeny steps that cancel each other, c.f. Remark 4.3.11. However, in these implementations, the isogenies start in the correct direction, and we learn $|e_i|$ from disorientation faults if we identify the first round in which $\mathfrak{l}_i$ is applied in the opposite direction. Therefore, once we learn the sets $S^{r,+}$ and $S^{r,-}$, we can determine e_i precisely.

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. Due to the bounds used in [ACR23], large degree ℓ_i do not show up in later rounds, which decreases the meet-in-the-middle complexity of connecting the components $G^{r,+}$ and $G^{(r+1),+}$ for later rounds r.

SQALE. SQALE [CCJR22] 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 primes. SQALE uses a 2-point strategy and only requires one round (keeping in mind 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),
- or the curve $E^+ = (\prod_{S^+} \mathfrak{l}_i^{-2}) \star E_B$, if the positive point was flipped,
- or the curve $E^- = (\prod_{S^-} \mathfrak{l}_i^{2}) \star E_B$, 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 points randomly, and some of the isogeny computation will fail, producing faulty curves close to $E^\pm$ (and curves with the same orientation will be close to each other, as in Section 7.4.5). Getting enough faulty curves 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 [CR22] in another 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.

7.6 The `pubcrawl` tool

The post-processing stage of our attack requires reconstructing 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. Here we report on implementation details and performance results for our `pubcrawl` software.[1]

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 for other applications requiring neighborhood searches in CSIDH in the future.

Algorithm. `pubcrawl` 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

[1] The `pubcrawl` software is available at https://yx7.cc/code/pubcrawl/pubcrawl-latest.tar.xz. The name refers to *crawl*ing the graph of *pub*lic keys, and a tour taking in several pubs or drinking places, with one or more drinks at each.

isogeny steps ("sign", denoting orientation of the steps), 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 $\leq m$ can be computed [CS97, § 3] as $G_n(m) = \sum_{k=0}^{m} \binom{n}{k}\binom{m-k+n}{n}$. Similarly, the number of vectors in $\mathbb{Z}^n_{\geq 0}$ of 1-norm $\leq m$ equals $H_n(m) = \sum_{k=0}^{m} \binom{k+n-1}{n-1}$.

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.

The overwhelming majority of the cost comes from computing isogeny steps in a breadth-first manner, which parallelizes perfectly. Hence, both time and memory 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 ≈ 250 bytes per node visited, although this quantity can vary significantly with the data structure internals. Example estimates are given in Table 7.1. The `pubcrawl` approach could be undoubtedly sped up, for instance by using strategies to compute multiple steps.

Table 7.1: Example cost estimates per target curve for various `pubcrawl` instances, assuming each isogeny step takes 0.7 milliseconds and consumes 250 bytes. For example, an isogeny walk of length up to 10 between two given curves can be recovered using approximately 10 core days and 300 gigabytes of RAM.

sign	$\lvert\text{support}\rvert$	distance	cardinality of search space	core time	memory
both	74	≤ 4	$20{,}549{,}801 \approx 2^{24.29}$	4.0 h	5.1 GB
both	74	≤ 5	$612{,}825{,}229 \approx 2^{29.19}$	5.0 d	153.2 GB
both	74	≤ 6	$15{,}235{,}618{,}021 \approx 2^{33.83}$	123.4 d	3.8 TB
both	74	≤ 7	$324{,}826{,}290{,}929 \approx 2^{38.24}$	7.2 y	81.2 TB
one	74	≤ 4	$1{,}426{,}425 \approx 2^{20.44}$	16.6 min	356.6 MB
one	74	≤ 5	$22{,}537{,}515 \approx 2^{24.43}$	4.4 h	5.6 GB
one	74	≤ 6	$300{,}500{,}200 \approx 2^{28.16}$	2.4 d	75.1 GB
one	74	≤ 7	$3{,}477{,}216{,}600 \approx 2^{31.70}$	28.2 d	869.3 GB

7.7 Hashed version

The attacker-observable output in Diffie–Hellman-style key agreements is not the shared elliptic curve, but a certain derived value (cf. Remark 7.3.1). 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. So 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 $\text{MAC}_k(\texttt{str})$ for some known fixed string `str`.

The attack strategies from Section 7.4 and Section 7.5 exploit the connections between the various faulty curves, but when we are only given a derived value, we are unable to apply isogenies. Our attack 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 call the observable values of the faulty curves E the *hash* $H(E)$. We assume that we can derive $H(E)$ for a given E, but that we cannot recover E given only $H(E)$.

As we cannot apply isogenies to the hashes, we must adapt the strategy from Section 7.4. 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 reduce the support when applying `pubcrawl`. But from the frequency analysis (Corollary 7.4.3), we can still identify the two most frequent new hashes h_1, h_2 per round as the probable hashes of $H(E^{r,\pm})$.

Example 7.7.1 (CSIDH)**.** When faulting the first point, the two most common hashed values are our best guesses for the hashes of $E^{1,\pm}$. Considering faults in the second point, we guess $H(E^{2,\pm})$ to be the most common hashes that have not appeared in round 1. Similarly for later points.

To recover E given a hash $H(E)$, 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 $H(E)$. In practice, we run `pubcrawl` with one orientation (or both, in parallel) until we recognize $H(E^{r,\pm})$. Having identified $E^{r,\pm}$, we can then run a small neighborhood search around $E^{r,\pm}$ to identify the hashes of the faulty curves E_t close to $E^{r,\pm}$. 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 E_B.

Example 7.7.2 (CSIDH-512)**.** The curves $E^{5,\pm}$ have the smallest distance to E_B (cf. Section 7.5.1). Starting from the public key E_B, we thus first search the paths to the curves $E^{5,\pm}$. We do this by growing two neighborhoods (with positive and negative orientation) from E_B. While the expected distance of the faulty curves is ≈ 7, it can be a lot larger. But such distances are rare: both $E^{5,\pm}$ having distance larger than 10 has probability $\approx 0.3\%$. Hence, we do expect to find a connection to at least one of the curves $E^{5,\pm}$ within distance 10, meaning that we

expect the first connection to cost no more than $2\sum_{i=0}^{10}\binom{74}{i} \approx 2^{40.6}$ isogeny step evaluations (and likely less). Once we identify the orientation of some primes, the support for `pubcrawl` becomes smaller and the search is more efficient.

Example 7.7.3 (CTIDH-512)**.** In CTIDH, in the worst case the distance from E_B to any $E^{r,\pm}$ is 14 (one prime per batch, all with the same orientation) and the average distance is 7 (Section 7.5.2). Thus, in a hashed variant, if we launch `pubcrawl` in both directions up to a distance 7, we are likely to already identify many hashes $H(E_t)$ and can recover E_t. We then crawl around these E_t to identify the other faulty curves. When we recover all E_t, we proceed as in Section 7.5.2.

Summary. In the hashed version, the main difference compared to the approach in Section 7.5 is that we can no longer mount meet-in-the-middle attacks between faulty curves, but we must always perform a one-way search from a given curve to a hash. Hence, we do not get the square-root speedup from meeting in the middle. Despite this increase in cost, this does not mean we cannot attack a hashed version. Although the brute-force search required to recover $E^{r,\pm}$ given only $H(E^{r,\pm})$ can get very expensive, especially for CSIDH over large fields $\mathbb{F}_p$, such a search always remains cheaper than the security level, as we only need to cover the gaps between all $E^{r,\pm}$ and E_B.

7.8 Twisting and precomputing

In this section, we use quadratic twists and precomputation to significantly speed up obtaining the private key $\mathfrak{a}$ given enough samples E_t, which is especially helpful for the "hashed" version described in Section 7.7. The attack target is a public key $E_B = \mathfrak{a} \star E_0$. Previously (Section 7.3), we attacked the computation of $\mathfrak{a} \star E_0$. In this section, we will use the quadratic twist E_{-B} as the input curve (Example 4.2.7). Since $E_{-B} = \mathfrak{a}^{-1} \star E_0$ by Remark 4.3.5, applying $\mathfrak{a}$ to it gives back the curve E_0. Faulting this computation therefore produces a faulty curve close to the fixed curve E_0. We refer to this attack variant as *using the twist*.

Moreover, twisting induces a symmetry around the curve E_0. This can be used to speed up `pubcrawl`: starting from E_0, if we reach E_t by a sequence of steps, then the opposite steps reach 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 $d' \leq d$ from E_0 can immediately be identified via a table lookup. And $\mathcal{C}$ can be precomputed once and for all, independent of the target instance. The symmetry of E_{-t} and E_t reduces storage by half.

Finally, this twisting attack cannot be prevented by simply 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 $\mathfrak{z} \star E_{-B}$ to the target device. The faulty curves E_t can then be moved back to the neighborhood of E_0 by computing $\mathfrak{z}^{-1} \star E_t$ at some cost

per E_t, or the attacker can precompute curves around $\mathfrak{z} \star E_0$. 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 $S^{r,\pm}$ 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 7.4.4). 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.

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 7.6), 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 from [UV21, § 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 $|S^{r,\pm}|$. Otherwise, no faulty curves end up within $\mathcal{C}$. For CSIDH, the smallest such sets are $S^{r_{\max},\pm}$ for the largest exponent $r_{\max}$ allowed by the keyspace (Section 4.3.4); e.g. for CSIDH-512, as always $S^{r_{\max},\pm} = S^{5,\pm}$ with 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 identify $S^{5,\pm}$ quickly by considering a small neighborhood of the curves $E^{5,\pm}$.

Note that for all the earlier rounds $r < r_{\max}$, the sets $S^{r,s}$ include $S^{r_{\max},s}$. Therefore, if we have $S^{r_{\max},s}$ for some s, we can shift all the faulty curves by two steps for every degree in $S^{r_{\max},s}$. 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 7.7: we precompute $\mathcal{C}'$ which instead of E_t includes $H(E_t)$ for all E_t in the neighborhood of E_0. Again, this works directly for 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} by the shift by the part of the secret key that is known (per orientation). 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 hash $H(E_t)$ are already in our database.

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.

7.9 Countermeasures

In this section, we present countermeasures against disorientation fault attacks from Section 7.3. We first review previous fault attacks on CSIDH and their countermeasures, as well as their influence on our attack in Section 7.9.1. We then discuss new countermeasures for one-point sampling from CSIDH and Elligator in Section 7.9.2, and estimate the costs of the countermeasures in Section 7.9.3.

7.9.1 Previous fault attacks and countermeasures

One way to recover secret keys is to target dummy isogenies [CKM+20, LH21].

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 [CKM+20, LH21] with essentially no computational overhead. When applied to dummy-based implementations, e.g., from [MCR19, OAYT19], 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 $batch_i$ 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 information with round-1 samples only.

Many fault attacks produce invalid intermediate values. In [CKM+20] some

low-level protections for dummy isogenies to detect fault injections are proposed. This approach does not prevent our disorientation attack, and is orthogonal to our proposed countermeasures. Its performance overhead for the CSIDH-512 implementation from [OAYT19] is reported to be 7%.

Faulting memory locations can identify dummy isogenies [CKM21]. In addition to the countermeasures above, the authors recommend using dummy-free implementations when concerned about fault attacks, with a roughly twofold slowdown [CCC+19]. However, as described in Section 7.5.3, dummy-free implementations are vulnerable to disorientation faults too.

Lastly, [CKM+20] reports that its fault attack theoretically could lead to disorientation of a point. Although the probability for this to happen is shown to be negligible, the authors propose to counter this attack vector by checking the field of definition of each isogeny kernel generator. This is rather expensive, with an overhead of roughly 30% for the implementation from [OAYT19], but also complicates the disorientation faults proposed in this work. We further discuss this in Section 7.9.2. We note that our countermeasures are significantly cheaper, but do not prevent the theoretical fault effect from [CKM+20].

7.9.2 Protecting square checks against fault attacks

The attack described in Section 7.3 can be applied to all implementations of CSIDH that use a call to `IsSquare` to determine the orientations of the involved point(s). The 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. For instance, faulting the computation of the Legendre-input $z = x^3 + Ax^2 + x$ results in a square check for a point unrelated to the actual x-coordinate in use, and yields a fault success probability of 50%.

Repeating square checks. One way to reduce the attacker's chances for a successful fault is to add redundant computations and repeat `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 [CKM+20]. 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 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. 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 is positive, if $\tilde{y}^2 = -z$, it is negative. If neither of these cases applies, i.e., $\tilde{y}^2 \neq \pm z$, 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 much 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 6.

Algorithm 6 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 $\prod [\mathfrak{l}_i]^{e_i} \star E_A = E_B$

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 $X_Q \leftarrow s \cdot z' \cdot X_{Q'}$, $Z_Q \leftarrow \tilde{y}^2 \cdot Z_{Q'}$.
9: Set $Q = (X_Q : Z_Q)$.
10: **for each** $i \in S$ **do**
11: Set $k \leftarrow k/\ell_i$ and compute $R \leftarrow [k]Q$. If $R = \infty$, **skip** this i.
12: Compute $\phi : E_A \to E_B$ with kernel $\langle R \rangle$.
13: Set $A \leftarrow B$, $Q \leftarrow \phi(Q)$, and $e_i \leftarrow e_i - s$.
14: **return** A.

Steps 3 and 4 of Algorithm 6 describe our method to determine 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 (see [CLM+18, BGS22], or a cheaper procedure [CKM+20]) at the end of the algorithm and abort. The attacker gains no information in both cases.

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 $z \neq x^3 + Ax^2 + x$. 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 that result in $z = z'$.

The attacker may still fault the computation of s in Step 4 of Algorithm 6. 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 in Algorithm 6 and the resulting curve is the correct output curve after all.

Protecting Elligator. Recall from Section 7.3 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 adapt the pseudo y-coordinate technique from Section 7.9.2: we determine orientations and verify their correctness by applying this countermeasure for both P_+ and P_- separately. We dub this version of the Elligator sampling `Elligreator`. Faulting the computations of the x-coordinates of the two points within Elligator (as in [CCC+19, Algorithm 3]) is also 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, as argued in [BBC+21]. It is unclear which approach should be favored.

7.9.3 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. (Note that neither has particularly low Hamming weight.) Furthermore, we require a handful of extra operations for computing z', X_Q, and Z_Q in Steps 7 and 8 of Algorithm 6. 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 of CSIDH will essentially not be noticeable in terms of performance, since the extra operations are negligible in comparison to the overall cost of the CSIDH action.

In 2-point variants, we use `Elligreator`, which requires two exponentiations instead of one. Thus, the countermeasure is expected to add a more significant but still relatively small overhead. CTIDH uses two calls to `Elligreator` per round, and both executions contain two pseudo-y checks respectively. For CTIDH-512, we estimate the cost as follows: the exponentiation by $(p-1)/2$ costs 602 multiplications, including squarings [BBC+21]. As CTIDH-512 requires roughly 20 rounds per run, we add two additional exponentiations by $(p+1)/4$ per round, and these have almost the same cost of 602 multiplications, the overhead is approximately $2 \cdot 20 \cdot 602 = 24080$ multiplications. Ignoring the negligible amount of further multiplications we introduce, this comes on top of a CTIDH-512 group action, which takes 438006 multiplications on average. Thus, we expect the total overhead of our countermeasure to be roughly 5.5% in CTIDH-512.

Bibliography

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. *Status report on the third round of the NIST Post-Quantum Cryptography Standardization process*. 2022. https://doi.org/10.6028/NIST.IR.8413-upd1. (Cited on page 5.)

[AAM19] Gora Adj, Omran Ahmadi, and Alfred Menezes. On isogeny graphs of supersingular elliptic curves over finite fields. In *Finite Fields and Their Applications*, volume 55, pages 268–283, 2019. https://doi.org/10.1016/j.ffa.2018.10.002. (Cited on pages 37 and 75.)

[ACL+19] Sarah Arpin, Catalina Camacho-Navarro, Kristin Lauter, Joelle Lim, Kristina Nelson, Travis Scholl, and Jana Sotáková. Adventures in Supersingularland. arXiv:1909.07779, 2019. https://doi.org/10.48550/arXiv.1909.07779. (Cited on page 75.)

[ACL+23] Sarah Arpin, Catalina Camacho-Navarro, Kristin Lauter, Joelle Lim, Kristina Nelson, Travis Scholl, and Jana Sotáková. Adventures in Supersingularland. In *Experimental Mathematics*, volume 32, pages 241–268. Taylor & Francis, 2023. https://doi.org/10.1080/10586458.2021.1926009. (Cited on pages 1, 7, 11, 31, 39, and 40.)

[ACMR22] Gora Adj, Jesús-Javier Chi-Domínguez, Víctor Mateu, and Francisco Rodríguez-Henríquez. Faulty isogenies: a new kind of leakage. Cryptology ePrint Archive 2022/153, 2022. https://ia.cr/2022/153. (Cited on page 141.)

[ACR23] Gora Adj, Jesús-Javier Chi-Domínguez, and Francisco Rodríguez-Henríquez. Karatsuba-based square-root Vélu's formulas applied to

two isogeny-based protocols. In *Journal of Cryptographic Engineering*, volume 13, pages 89–106, 2023. https://doi.org/10.1007/s13389-022-00293-y. (Cited on pages 95, 96, 130, 135, 136, 137, and 157.)

[ADMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic Group Actions and Applications. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020*, volume 12492 of *Lecture Notes in Computer Science*, pages 411–439. Springer, 2020. https://doi.org/10.1007/978-3-030-64834-3_14. (Cited on pages 5, 82, and 85.)

[Ame04] American Mathematical Society. The Culture of Research and Scholarship in Mathematics: Joint Research and Its Publication. https://www.ams.org//profession/leaders/CultureStatement04.pdf, 2004. Accessed: 2024-03-11. (Cited on page 1.)

[BBC+21] Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, T. Chou, Tanja Lange, Michael Meyer, Benjamin Smith, and Jana Sotáková. CTIDH: faster constant-time CSIDH. In *CHES 2021*, volume 2021(4), pages 351–387, 2021. https://doi.org/10.46586/tches.v2021.i4.351-387. (Cited on pages 2, 7, 9, 77, 84, 94, 96, 115, 128, 132, 134, 140, 155, and 167.)

[BCC+23] Andrea Basso, Giulio Codogni, Deirdre Connolly, Luca De Feo, Tako Boris Fouotsa, Guido Maria Lido, Travis Morrison, Lorenz Panny, Sikhar Patranabis, and Benjamin Wesolowski. Supersingular Curves You Can Trust. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, pages 405–437. Springer, 2023. https://doi.org/10.1007/978-3-031-30617-4_14. (Cited on pages 34 and 36.)

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. In *Journal of Symbolic Computation*, volume 24, pages 235–265, 1997. Computational algebra and number theory (London, 1993). (Cited on page 110.)

[BDLS20] Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. In Steven Galbraith, editor, *ANTS XIV*, pages 39–55. Mathematical Sciences Publishers, 2020. https://doi.org/10.2140/obs.2020.4.39. (Cited on pages 9, 15, 92, 116, 130, and 134.)

[Ber18] Daniel J. Bernstein. djbsort. https://sorting.cr.yp.to, 2018. Accessed: 2024-03-12. (Cited on page 131.)

[BGDS23] Gustavo Banegas, Valerie Gilchrist, Anaëlle Le Dévéhat, and Benjamin Smith. Fast and Frobenius: Rational Isogeny Evaluation over Finite Fields. In Abdelrahaman Aly and Mehdi Tibouchi, editors, *LATINCRYPT 2023*, pages 129–148. Springer, 2023. https://doi.org/10.1007/978-3-031-44469-2_7. (Cited on page 92.)

[BGS22] Gustavo Banegas, Valerie Gilchrist, and Benjamin Smith. Efficient supersingularity testing over $GF(p)$ and CSIDH key validation. In *Mathematical Cryptology*, volume 2, page 21–35, 2022. https://journals.flvc.org/mathcryptology/article/view/132125. (Cited on page 166.)

[BHKL13] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC*, pages 967–980. ACM, 2013. https://doi.org/10.1145/2508859.2516734. (Cited on pages 90, 132, and 142.)

[BJS14] Jean-François Biasse, David Jao, and Anirudh Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves. In *INDOCRYPT 2014*, volume 8885 of *Lecture Notes in Comp[uter Science*, pages 428–442. Springer, 2014. https://doi.org/10.1007/978-3-319-13039-2_25. (Cited on pages 8 and 63.)

[BKL+23] Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska. Disorientation Faults in CSIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, pages 310–342. Springer, 2023. https://doi.org/10.1007/978-3-031-30589-4_11. (Cited on pages 2, 7, 10, 77, and 139.)

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient Isogeny Based Signatures Through Class Group Computations. In Steven D. Galbraith and Shiho Moriai, editors, *Asiacrypt 2019*, pages 227–247. Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-34578-5_9. (Cited on pages 5, 8, and 85.)

[BLMP19] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum Circuits for the CSIDH: Optimizing Quantum Evaluation of Isogenies. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019*, volume 11477 of *Lecture Notes in Computer Science*, pages 409–441. Springer, 2019. https://doi.org/10.1007/

978-3-030-17656-3_15. (Cited on pages 9, 84, 93, 95, 116, 117, 124, 125, 130, 132, and 140.)

[BLS11] Reinier Bröker, Kristin Lauter, and Andrew V. Sutherland. Modular polynomials via isogeny volcanoes. In *Mathematics of Computation*, volume 81, page 1201–1231. American Mathematical Society (AMS), 2011. http://dx.doi.org/10.1090/S0025-5718-2011-02508-1. (Cited on page 31.)

[BOS16] Jan Hendrik Bruinier, Ken Ono, and Andrew V. Sutherland. Class polynomials for nonholomorphic modular functions. In *Journal of Number Theory*, volume 161, page 204–229. Elsevier BV, 2016. http://dx.doi.org/10.1016/j.jnt.2015.07.002. (Cited on page 31.)

[Bru11] Peter Bruin. The Tate pairing for Abelian varieties over finite fields. In *Journal de Théorie des Nombres de Bordeaux*, volume 23, pages 323–328. Société Arithmétique de Bordeaux, 2011. http://www.numdam.org/articles/10.5802/jtnb.764/. (Cited on page 103.)

[BS] Reinier Bröker and Andrew V. Sutherland. An explicit height bound for the classical modular polynomial. volume 22, pages 293–313. https://doi.org/10.1007/s11139-010-9231-8. (Cited on page 32.)

[BS96] Wieb Bosma and Peter Stevenhagen. On the computation of quadratic 2-class groups. In *Journal de Théorie des Nombres de Bordeaux*, volume 8, pages 283–313. Société Arithmétique de Bordeaux, 1996. https://www.jstor.org/stable/43974214. (Cited on page 111.)

[BS20] Xavier Bonnetain and André Schrottenloher. Quantum Security Analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020*, pages 493–522, 2020. https://doi.org/10.1007/978-3-030-45724-2_17. (Cited on page 84.)

[BSS05] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart, editors. *Advances in elliptic curve cryptography*, volume 317 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 2005. (Cited on pages 101 and 113.)

[CCC+19] 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 Peter Schwabe and Nicolas Thériault, editors, *LATINCRYPT 2019*, volume 11774 of *Lecture Notes in Computer Science*, pages 173–193. Springer, 2019. https://doi.org/10.1007/

978-3-030-30530-7_9. (Cited on pages 95, 96, 130, 132, 135, 136, 137, 157, 164, and 166.)

[CCJR22] 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*, volume 12, pages 349–368. Springer, 2022. https://doi.org/10.1007/s13389-021-00271-w. (Cited on pages 84, 94, 95, 119, 130, 135, 136, 137, 157, and 158.)

[CD20] Wouter Castryck and Thomas Decru. CSIDH on the surface. In Jintai Ding and Jean-Pierre Tillich, editors, *PQCrypto 2020*, volume 12100 of *Lecture Notes in Computer Science*, pages 111–129. Springer, 2020. https://doi.org/10.1007/978-3-030-44223-1_7. (Cited on pages 88 and 112.)

[CD23] Wouter Castryck and Thomas Decru. An Efficient Key Recovery Attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, pages 423–447. Springer, 2023. https://doi.org/10.1007/978-3-031-30589-4_15. (Cited on pages 5, 36, and 84.)

[CDEL21] Wouter Castryck, Ann Dooms, Carlo Emerencia, and Alexander Lemmens. A fusion algorithm for solving the hidden shift problem in finite abelian groups. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *PQCrypto 2021*, volume 12841 of *Lecture Notes in Computer Science*, pages 133–153. Springer, 2021. https://doi.org/10.1007/978-3-030-81293-5_8. (Cited on page 110.)

[CDV20] Wouter Castryck, Thomas Decru, and Frederik Vercauteren. Radical isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020*, pages 493–519. Springer, 2020. https://doi.org/10.1007/978-3-030-64834-3_17. (Cited on page 92.)

[CFL+] Anamaria Costache, Brooke Feigon, Kristin Lauter, Maike Massierer, and Anna Puskás. Ramanujan Graphs in Cryptography. In Jennifer S. Balakrishnan, Amanda Folsom, Matilde Lalín, and Michelle Manes, editors, *Research Directions in Number Theory*, Association for Women in Mathematics Series, pages 1–40. Springer. https://doi.org/10.1007/978-3-030-19478-9_1. (Cited on page 75.)

[CHM+23] 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 Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO*

2023, pages 762–792. Springer, 2023. https://doi.org/10.1007/978-3-031-38548-3_25. (Cited on page 112.)

[CHVW22] Wouter Castryck, Marc Houben, Frederik Vercauteren, and Benjamin Wesolowski. On the decisional Diffie–Hellman problem for class group actions on oriented elliptic curves. In *ANTS-XV*, volume 8 of *Research in Number Theory*, page 99, 2022. https://doi.org/10.1007/s40993-022-00399-6. (Cited on pages 112 and 113.)

[CJS14] Andrew M. Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. In *Journal of Mathematical Cryptology*, volume 8, pages 1–29, 2014. https://doi.org/10.1515/jmc-2012-0016. (Cited on page 84.)

[CK20] Leonardo Colò and David Kohel. Orienting supersingular isogeny graphs. In *Journal of Mathematical Cryptology*, volume 14, pages 414–437. De Gruyter, 2020. https://doi.org/10.1515/jmc-2019-0034. (Cited on pages 28 and 30.)

[CKM+20] 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 *FDTC 2020*, pages 57–65. IEEE, 2020. https://doi.org/10.1109/FDTC51366.2020.00015. (Cited on pages 141, 163, 164, and 166.)

[CKM21] Fabio Campos, Juliane Krämer, and Marcel Müller. Safe-Error Attacks on SIKE and CSIDH. In *SPACE 2021*, volume 13162 of *Lecture Notes in Computer Science*, page 104–125. Springer-Verlag, 2021. https://doi.org/10.1007/978-3-030-95085-9_6. (Cited on pages 141 and 164.)

[CLG09] Denis Xavier Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic Hash Functions from Expander Graphs. In *Journal of Cryptology*, volume 22, pages 93–113, 2009. https://doi.org/10.1007/s00145-007-9002-x. (Cited on pages 5, 36, 42, and 72.)

[CLM+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An Efficient Post-Quantum Commutative Group Action. In Thomas Peyrin and Steven D. Galbraith, editors, *ASIACRYPT 2018*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018. https://doi.org/10.1007/978-3-030-03332-3_15. (Cited on pages 5, 6, 7, 24, 36, 57, 77, 84, 87, 93, 94, 95, 110, 111, 117, 129, 136, 142, and 166.)

[CMRS22] Fabio Campos, Michael Meyer, Krijn Reijnders, and Marc Stöttinger. Patient Zero and Patient Six: Zero-Value and Correlation Attacks on CSIDH and SIKE. In *SAC 2022*, 2022. https://ia.cr/2022/904. (Cited on page 140.)

[Cos20] Craig Costello. B-SIDH: Supersingular Isogeny Diffie-Hellman Using Twisted Torsion. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020*, volume 12492(2) of *Lecture Notes in Computer Science*, pages 440–463. Springer, 2020. https://doi.org/10.1007/978-3-030-64834-3_15. (Cited on page 37.)

[Cou06] Jean-Marc Couveignes. Hard Homogeneous Spaces. IACR Cryptology ePrint Archive 2006/291, 2006. https://ia.cr/2006/291. (Cited on pages 5, 6, 77, 81, 82, 83, 84, and 85.)

[Cox89] David Cox. *Primes of the form* $x^2 + ny^2$. John Wiley and Sons, Inc., New York, 1989. https://doi.org/10.1002/9781118400722. (Cited on pages 11, 60, 98, 100, and 104.)

[CPV] Wouter Castryck, Lorenz Panny, and Frederik Vercauteren. Rational Isogenies from Irrational Endomorphisms. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020*, Lecture Notes in Computer Science, pages 523–548. Springer International Publishing. https://doi.org/10.1007/978-3-030-45724-2_18. (Cited on pages 52 and 75.)

[CR22] Jesús-Javier Chi-Domínguez and Francisco Rodríguez-Henríquez. Optimal strategies for CSIDH. In *Advances in Mathematics of Communications*, volume 16, pages 383–411, 2022. https://doi.org/10.3934/amc.2020116. (Cited on pages 94, 95, 96, 121, 126, 127, 135, 136, 137, 157, and 158.)

[CS97] 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*, pages 2369–2389, 1997. https://doi.org/10.1098/rspa.1997.0126. (Cited on page 159.)

[CS18] Craig Costello and Benjamin Smith. Montgomery curves and their arithmetic. In *Journal of Cryptographic Engineering*, volume 8, pages 227–240, 2018. https://doi.org/10.1007/s13389-017-0157-6. (Cited on page 87.)

[CS22] Mathilde Chenu and Benjamin Smith. Higher-degree supersingular group actions. In *Mathematical Cryptology*, volume 1, pages 85–101, 2022. https://journals.flvc.org/mathcryptology/article/view/130604. (Cited on pages 72 and 75.)

[CSA18] ECRYPT – CSA. Algorithms, Key Size and Protocols Report (2018), 2018. Available at https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf. Accessed on 2024-03-12. (Cited on pages 4 and 81.)

[CSV20] Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. Breaking the decisional Diffie-Hellman problem for class group actions using genus theory. In *CRYPTO 2020*, volume 12171(2) of *Lectures Notes in Computer Science*, pages 92–120. Springer, 2020. https://doi.org/10.1007/978-3-030-56880-1_4. (Cited on pages 1, 7, 8, 24, 27, 83, 97, and 98.)

[CSV22a] Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. Breaking the Decisional Diffie–Hellman Problem for Class Group Actions Using Genus Theory: Extended Version. In *Journal of Cryptology*, volume 35, page 24, 2022. https://doi.org/10.1007/s00145-022-09435-1. (Cited on pages 1, 7, 8, 97, and 112.)

[CSV22b] Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. Magma code breaking class group action DDH for elliptic and hyperelliptic curves, 2022. Available at https://github.com/KULeuven-COSIC/group_action_DDH. (Cited on page 110.)

[Deu41] Max Deuring. Die Typen der Multiplikatorenringe elliptischer Funktionenkörper. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 14, pages 197–272, 1941. https://doi.org/10.1007/BF02940746. (Cited on pages 23 and 26.)

[DF10] Luca De Feo. Fast algorithms for towers of finite fields and isogenies. 2010. PhD thesis. (Cited on page 108.)

[DFK+23] Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. SCALLOP: Scaling the CSI-FiSh. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023*, pages 345–375. Springer, 2023. https://doi.org/10.1007/978-3-031-31368-4_13. (Cited on pages 28 and 86.)

[DG16] Christina Delfs and Steven D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. In *Designs, Codes and Cryptography*, volume 78, pages 425–440, 2016. http://dx.doi.org/10.1007/s10623-014-0010-1. (Cited on pages 7, 21, 27, 40, 43, 44, 54, 62, 64, 67, and 84.)

[DG19] Luca De Feo and Steven D. Galbraith. SeaSign: Compact Isogeny Signatures from Class Group Actions. In Yuval Ishai

and Vincent Rijmen, editors, *EUROCRYPT 2019*, pages 759–789. Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-17659-4_26. (Cited on pages 5 and 85.)

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, volume 22, pages 644–654, 1976. https://doi.org/10.1109/TIT.1976.1055638. (Cited on pages 3 and 78.)

[DJP14] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Journal of Mathematical Cryptology*, volume 8, pages 209–247, 2014. https://doi.org/10.1515/jmc-2012-0015. (Cited on pages 5, 37, 93, and 94.)

[DKL+20] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020*, volume 12491 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2020. https://doi.org/10.1007/978-3-030-64837-4_3. (Cited on pages 5, 37, and 38.)

[DKS18] Luca De Feo, Jean Kieffer, and Benjamin Smith. Towards Practical Key Exchange from Ordinary Isogeny Graphs. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018*, volume 11274(3) of *Lecture Notes in Computer Science*, pages 365–394. Springer, 2018. https://ia.cr/2018/485. (Cited on pages 6, 77, 85, 86, and 110.)

[DM20] Luca De Feo and Michael Meyer. Threshold Schemes from Isogeny Assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020*, pages 187–212. Springer International Publishing, 2020. https://doi.org/10.1007/978-3-030-45388-6_7. (Cited on page 5.)

[EHL+a] Eisenträger, Sean Hallgren, Kristin Lauter, Travis Morrison, and Christophe Petit. Supersingular Isogeny Graphs and Endomorphism Rings: Reductions and Solutions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018*, Lecture Notes in Computer Science, pages 329–368. Springer. https://doi.org/10.1007/978-3-319-78372-7_11. (Cited on page 36.)

[EHL+b] Kirsten Eisenträger, Sean Hallgren, Chris Leonardi, Travis Morrison, and Jennifer Park. Computing endomorphism rings of supersingular elliptic curves and connections to path-finding in isogeny

graphs. In *ANTS XIV*, volume 4, pages 215–232. Mathematical Sciences Publishers. https://doi.org/10.2140/obs.2020.4.215. (Cited on pages 72 and 75.)

[ElG84] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In George Robert Blakley and David Chaum, editors, *CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984. https://doi.org/10.1007/3-540-39568-7_2. (Cited on page 3.)

[EPSV23] Jonathan Komada Eriksen, Lorenz Panny, Jana Sotáková, and Mattia Veroni. Deuring for the People: Supersingular Elliptic Curves with Prescribed Endomorphism Ring in General Characteristic. Cryptology ePrint Archive 2023/106, 2023. https://ia.cr/2023/106. (Cited on page 92.)

[Fed22] Federal Office for Information Security (BSI). Quantum-safe cryptography – fundamentals, current developments and recommendations. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography, 2022. Accessed: 2024-03-11. (Cited on page 5.)

[FF20] Enric Florit and Gerard Finol. Isogeny graphs of supersingular elliptic curves (1.0.1) [data set]. Zenodo, 2020. https://doi.org/10.5281/zenodo.4304044. (Cited on page 75.)

[FIM+14] Katalin Friedl, Gábor Ivanyos, Frédéric Magniez, Miklos Santha, and Pranab Sen. Hidden translation and translating coset in quantum computing. In *SIAM J. Comput.*, volume 43, pages 1–24, 2014. https://doi.org/10.1137/130907203. (Cited on page 110.)

[Gal23] Steven Galbraith. Some comments on the CSIDH group action. https://ellipticnews.wordpress.com/2023/05/15/some-comments-on-the-csidh-group-action/, 2023. Accessed: 2024-03-12. (Cited on page 85.)

[Gol00] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, USA, 2000. https://doi.org/10.1017/CBO9780511546891. (Cited on pages 78 and 79.)

[GPS17] Steven D. Galbraith, Christophe Petit, and Javier Silva. Identification Protocols and Signature Schemes Based on Supersingular Isogeny Problems. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017*, pages 3–33. Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-70694-8_1. (Cited on page 5.)

[GW17] Alexandre Gélin and Benjamin Wesolowski. Loop-Abort Faults on Supersingular Isogeny Cryptosystems. In Tanja Lange and Tsuyoshi Takagi, editors, *PQCrypto 2017*, pages 93–106. Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-59879-6_6. (Cited on page 141.)

[HLKA20] 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 Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 2020*, volume 12146 of *Lecture Notes in Computer Science*, pages 481–501. Springer, 2020. https://doi.org/10.1007/978-3-030-57808-4_24. (Cited on pages 94, 95, 126, 135, 136, 137, and 140.)

[HM89] James L. Hafner and Kevin S. McCurley. A rigorous subexponential algorithm for computation of class groups. In *Journal of the American Mathematical Society*, volume 2, pages 837–850, 1989. https://doi.org/10.1090/S0894-0347-1989-1002631-0. (Cited on page 8.)

[Ibu82] Tomoyoshi Ibukiyama. On maximal orders of division quaternion algebras over the rational number field with certain optimal embeddings. In *Nagoya Math. J.*, volume 88, pages 181–195. Nagoya Mathematical Journal, 1982. https://doi.org/10.1017/S002776300002016X. (Cited on page 49.)

[IJ13] Sorina Ionica and Antoine Joux. Pairing the volcano. In *Mathematics of Computatio*, volume 82, pages 581–603, 2013. https://doi.org/10.48550/arXiv.1110.3602. (Cited on page 103.)

[Jan21] Ján Jančár. The state of tooling for verifying constant-timeness of cryptographic implementations. https://neuromancer.sk/article/26, 2021. Accessed: 2024-03-12. (Cited on page 133.)

[JD11] David Jao and Luca De Feo. Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In Bo-Yin Yang, editor, *PQCrypto 2011*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011. https://ia.cr/2011/506. (Cited on pages 5, 36, and 37.)

[JMV09] David Jao, Stephen D. Miller, and Ramarathnam Venkatesan. Expander graphs based on GRH with an application to elliptic curve cryptography. In *Journal of Number Theory*, volume 129, pages 1491–1504, 2009. https://doi.org/10.1016/j.jnt.2008.11.006. (Cited on page 24.)

[Kan89] Masanobu Kaneko. Supersingular j-invariants as singular moduli mod p. In *OSAKA JOURNAL OF MATHEMATICS*, volume 26(4), pages 849–855, 1989. https://doi.org/10.1142/S1793042122500555. (Cited on pages 44, 49, and 50.)

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007. (Cited on page 78.)

[KLPT14] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion ℓ-isogeny path problem. In *LMS J. Comput. Math.*, volume 17, pages 418–432, 2014. https://doi.org/10.1112/S1461157014000151. (Cited on page 16.)

[Koh96] David Kohel. *Endomorphism rings of elliptic curves over finite fields*. PhD thesis, University of California at Berkeley, 1996. (Cited on pages 11, 15, 22, 25, 26, 29, 30, 35, 49, and 92.)

[Lan10] Adam Langley. ctgrind—checking that functions are constant time with Valgrind. https://github.com/agl/ctgrind, 2010. Accessed: 2024-03-12. (Cited on page 133.)

[LB20] Jonathan Love and Dan Boneh. Supersingular curves with small noninteger endomorphisms. In *ANTS XIV (The Open Book Series)*, volume 4, pages 7–22. Mathematical Sciences Publishers, 2020. https://doi.org/10.2140/obs.2020.4.7. (Cited on page 75.)

[Len96] H.W. Lenstra, Jr. Complex Multiplication Structure of Elliptic Curves. In *Journal of Number Theory*, volume 56, pages 227–241. Elsevier, 1996. https://doi.org/10.1006/jnth.1996.0015. (Cited on pages 20, 30, and 102.)

[LGD21] Yi-Fu Lai, Steven D. Galbraith, and Cyprien Delpech de Saint Guilhem. Compact, Efficient and UC-Secure Isogeny-Based Oblivious Transfer. In *EUROCRYPT 2021*, volume 12696 of *Lecture Notes in Computer Science*, pages 213–241. Springer, 2021. https://doi.org/10.1007/978-3-030-77870-5_8. (Cited on pages 5 and 91.)

[LH21] J. T. LeGrow and A. Hutchinson. (Short Paper) Analysis of a Strong Fault Attack on Static/Ephemeral CSIDH. In Toru Nakanishi and Ryo Nojima, editors, *IWSEC 2021*, volume 12835 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 2021. https://doi.org/10.1007/978-3-030-85987-9_12. (Cited on pages 141 and 163.)

[LPS88] Alexander Lubotzky, Ralph S. Phillips, and Peter C. Sarnak. Ramanujan graphs. In *Combinatorica*, volume 8, pages 261–277, 1988. https://doi.org/10.1007/BF02126799. (Cited on page 72.)

[MCR19] Michael Meyer, Fabio Campos, and Steffen Reith. On Lions and Elligators: An efficient constant-time implementation of CSIDH. In Jintai Ding and Rainer Steinwandt, editors, *PQCrypto 2019*, volume 11505 of *Lecture Notes in Computer Science*, pages 307–325. Springer, 2019. https://doi.org/10.1007/978-3-030-25510-7_17. (Cited on pages 93, 95, 119, 122, 126, 133, 135, 136, 137, 140, 157, and 163.)

[Mes86] Jean-Francois Mestre. La méthode des graphes. exemples et applications. In *Proceedings of the international conference on class numbers and fundamental units of algebraic number fields*. Nagoya University, 1986. (Cited on page 35.)

[MM22] Luciano Maino and Chloe Martindale. An attack on SIDH with arbitrary starting curve. Cryptology ePrint Archive 2022/1026, 2022. https://ia.cr/2022/1026. (Cited on pages 5, 36, and 84.)

[MMS+06] Josep Miret, Ramiro Moreno, Daniel Sadornil, Juan Tena-Ayuso, and Magda Valls. An algorithm to compute volcanoes of 2-isogenies of elliptic curves over finite fields. In *Applied Mathematics and Computation*, volume 176, pages 739–750, 2006. https://doi.org/10.5565/PUBLMAT_PJTN05_08. (Cited on pages 30 and 102.)

[MR18] Michael Meyer and Steffen Reith. A Faster Way to the CSIDH. In Debrup Chakraborty and Tetsu Iwata, editors, *INDOCRYPT 2018*, volume 11356 of *Lecture Notes in Computer Science*, pages 137–152. Springer, 2018. https://doi.org/10.1007/978-3-030-05378-9_8. (Cited on pages 93 and 140.)

[MST+07] Josep Miret, Daniel Sadornil, Juan Tena-Ayuso, Rosana Tomàs, and Magda Valls. Volcanoes of ℓ-isogenies of elliptic curves over finite fields: The case $\ell = 3$. In *Publicacions Matemàtiques*, volume 51, pages 165–180, 2007. http://dx.doi.org/10.5565/PUBLMAT_PJTN05_08. (Cited on pages 30 and 102.)

[MW00] Ueli M. Maurer and Stefan Wolf. The Diffie–Hellman Protocol. In *Designs, Codes and Cryptography*, volume 19, pages 147–171. Springer, 2000. https://doi.org/10.1023/A:1008302122286. (Cited on page 79.)

[NOTT23] Kohei Nakagawa, Hiroshi Onuki, Atsushi Takayasu, and Tsuyoshi Takagi. l_1-norm ball for CSIDH: Optimal strategy for choosing

the secret key space. In *Discrete Applied Mathematics*, volume 328, pages 70–88, 2023. https://doi.org/10.1016/j.dam.2022.12.002. (Cited on pages 9, 116, and 117.)

[NS07] Nicholas Nethercote and Julian Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. In *ACM SIGPLAN 2007*, pages 89–100, 2007. https://doi.org/10.1145/1250734.1250746. (Cited on page 133.)

[OAYT19] Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi. A faster constant-time algorithm of CSIDH keeping two points. In Nuttapong Attrapadung and Takeshi Yagi, editors, *IWSEC 2019*, volume 11689 of *Lecture Notes in Computer Science*, pages 23–33. Springer, 2019. https://ia.cr/2019/353. (Cited on pages 94, 95, 96, 119, 120, 121, 122, 135, 136, 137, 163, and 164.)

[Ogg75] Andrew P Ogg. Automorphismes de courbes modulaires. In *Séminaire Delange-Pisot-Poitou. Théorie des nombres*, volume 16, pages 1–8, 1975. http://www.numdam.org/item/SDPP_1974-1975__16_1_A4_0/. (Cited on pages 69 and 72.)

[Onu21] Hiroshi Onuki. On oriented supersingular elliptic curves. In *Finite Fields and Their Applications*, volume 69, page 101777. Elsevier BV, 2021. http://dx.doi.org/10.1016/j.ffa.2020.101777. (Cited on page 31.)

[Pan23] Lorenz S. Panny. CSI-FiSh really isn't polynomial-time. https://yx7.cc/blah/2023-04-14.html, 2023. Accessed: 2024-03-11. (Cited on page 85.)

[Pei20] Chris Peikert. He Gives C-Sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020*, pages 463–492, 2020. https://doi.org/10.1007/978-3-030-45724-2_16. (Cited on page 84.)

[PR23] Aurel Page and Damien Robert. Introducing Clapoti(s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive 2023/1766, 2023. https://ia.cr/2023/1766. (Cited on page 85.)

[Rob22a] Damien Robert. Evaluating isogenies in polylogarithmic time. Cryptology ePrint Archive 2022/1068, 2022. https://ia.cr/2022/1068. (Cited on pages 36 and 92.)

[Rob22b] Damien Robert. Some applications of higher dimensional isogenies to elliptic curves (overview of results). Cryptology ePrint Archive 2022/1704, 2022. https://ia.cr/2022/1704. (Cited on page 36.)

[Rob23] Damien Robert. Breaking SIDH in Polynomial Time. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, pages 472–503. Springer, 2023. https://doi.org/10.1007/978-3-031-30589-4_17. (Cited on pages 5, 36, and 84.)

[RS06] Alexander Rostovtsev and Anton Stolbunov. Public-Key Cryptosystem Based on Isogenies. IACR Cryptology ePrint Archive 2006/145., 2006. https://ia.cr/2006/145. (Cited on pages 5, 6, 77, 81, 82, 83, and 84.)

[Sar18] Naser T. Sardari. Diameter of Ramanujan Graphs and Random Cayley Graphs. In *Combinatorica*, volume 39, pages 427–446, 2018. https://doi.org/10.1007/s00493-017-3605-0. (Cited on page 72.)

[Sch] René Schoof. Nonsingular plane cubic curves over finite fields. In *Journal of Combinatorial Theory, Series A*, volume 46, pages 183–211. https://doi.org/10.1016/0097-3165(87)90003-3. (Cited on page 23.)

[Sch95] René Schoof. Counting points on elliptic curves over finite fields. In *Journal de théorie des nombres de Bordeaux*, volume 7, pages 219–254. Université Bordeaux I, 1995. http://www.numdam.org/item/JTNB_1995__7_1_219_0/. (Cited on page 12.)

[Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In *SIAM Rev.*, volume 41, pages 303–332, 1999. http://dx.doi.org/10.1137/S0036144598347011. (Cited on pages 4 and 81.)

[SIK17] SIKE. Supersingular isogeny key encapsulation, 2017. https://www.sike.org. Accessed: 2024-03-12. (Cited on pages 5 and 36.)

[Sil94] Joseph H. Silverman. *Advanced Topics in the Arithmetic of Elliptic Curves*. Springer New York, New York, NY, 1994. https://doi.org/10.1007/978-1-4612-0851-8_3. (Cited on page 31.)

[Sil09] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, 2 edition, 2009. (Cited on pages 11, 12, 13, 14, 17, 20, 28, and 35.)

[Smi18] Benjamin Smith. Pre- and Post-quantum Diffie–Hellman from Groups, Actions, and Isogenies. In Lilya Budaghyan and Francisco Rodríguez-Henríquez, editors, *Arithmetic of Finite Fields*, pages 3–40. Springer, 2018. https://doi.org/10.1007/978-3-030-05153-2_1. (Cited on page 78.)

[Ste22] Bruno Sterner. Commitment Schemes from Supersingular Elliptic Curve Isogeny Graphs. In *Journal of Mathematical Cryptology*, volume 1, page 40–51, 2022. https://journals.flvc.org/mathcryptology/article/view/130656. (Cited on page 33.)

[Sut] Andrew Sutherland. Modular polynomials. https://math.mit.edu/~drew/ClassicalModPolys.html. Accessed: 2024-03-11. (Cited on pages 31 and 32.)

[Sut13a] Andrew Sutherland. On the evaluation of modular polynomials. In *ANTS X*, volume 1, pages 531–555. Mathematical Sciences Publishers, 2013. http://dx.doi.org/10.2140/obs.2013.1.531. (Cited on page 31.)

[Sut13b] Andrew V. Sutherland. Isogeny volcanoes. In *ANTS-X*, volume 1 of *Open Book Series*, pages 507–530. Mathematical Sciences Publishers, 2013. http://dx.doi.org/10.2140/obs.2013.1.507. (Cited on pages 11, 24, 103, 104, 109, and 110.)

[TDEP21] Élise Tasso, Luca De Feo, Nadia El Mrabet, and Simon Pontié. Resistance of isogeny-based cryptographic implementations to a fault attack. In Shivam Bhasin and Fabrizio De Santis, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 255–276. Springer International Publishing, 2021. https://doi.org/10.1007/978-3-030-89915-8_12. (Cited on page 141.)

[The24] The Sage Developers. *SageMath, the Sage Mathematics Software System (version 10.2)*, 2024. https://sagemath.org. (Cited on pages 64 and 72.)

[Ti17] Yan Bo Ti. Fault Attack on Supersingular Isogeny Cryptosystems. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography 2017*, pages 107–122. Springer, 2017. https://doi.org/10.1007/978-3-319-59879-6_7. (Cited on page 141.)

[UV21] Aleksei Udovenko and Giuseppe Vitto. Revisiting Meet-in-the-Middle Cryptanalysis of SIDH/SIKE with Application to the $IKEp182 Challenge. Cryptology ePrint Archive 2021/1421, 2021. https://ia.cr/2021/1421. (Cited on page 162.)

[Vél71] Jacques Vélu. Isogénies entre courbes elliptiques. In *Comptes Rendus de l'Académie des Sciences de Paris*, volume 273(A)4, pages 238–241, 1971. https://gallica.bnf.fr/ark:/12148/bpt6k56191248/f52.item. (Cited on pages 15, 46, 91, and 108.)

[Wat69] William C. Waterhouse. Abelian varieties over finite fields. In *Annales scientifiques de l'École Normale Supérieure*, volume 2, pages 521–560. Elsevier, 1969. https://doi.org/10.24033/asens.1183. (Cited on pages 11, 20, 22, 23, and 37.)

[Wes22] Benjamin Wesolowski. The supersingular isogeny path and endomorphism ring problems are equivalent. In *FOCS 2021*, pages 1100–1111. IEEE, 2022. https://doi.org/10.1109/FOCS52979.2021.00109. (Cited on page 16.)

[Wit01] Christian Wittmann. Group Structure of Elliptic Curves over Finite Fields. In *Journal of Number Theory*, volume 88, pages 335–344, 2001. https://doi.org/10.1006/jnth.2000.2622. (Cited on page 29.)

Samenvatting

Ik ben van de generatie die opgegroeid is met de smartphone. Dat kleine dingetje is al bijna 20 jaar in mijn hand geweest en is voor mij de meest belangrijke bron van communicatie met de wereld. Niet alleen om te praten met mijn familie en vrienden, maar tegenwoordig doe ik bijna alles op mijn telefoontje.

Toen ik een bankrekening ging aanvragen, ging ik nog naar de bank in de buurt van mijn huis, maar die locatie is al jaren gesloten en alles regel ik via de app. De overheid in Nederland maakt gebruik van DigiD, dus alles wat de overheid met mij wil gebeurt over het Internet. En hoewel mijn eerste belastingervaring met het legendarische papieren M-formulier van 80 pagina's was, heb ik al mijn verdere belastingaangiften online kunnen indienen, en zou niet meer weten hoe ik het anders zou doen.

Het is bijna onmogelijk om met de online wereld los te breken, en we zijn gedwongen om al onze zaken online te regelen. We hebben daarvoor bescherming nodig, niet alleen voor privacy, maar ook tegen misbruik en inmenging door kwaadwillende actoren. Ik vertrouwde dat de werknemers in het gebouw van marmer echt bij de bank werkten, en dat ze mijn financiële zaken mochten behandelen. Wanneer ik met mijn familie in de tuin aan het praten ben, kan ik rondkijken en beslissen of ik de intieme details van mijn leven wil delen, of er iemand ongewenst aan het luisteren is.

Wanneer ik verbonden ben met het internet, wil ik dezelfde garanties. Ik wil dat mijn communicatie *versleuteld* is: extra beschermd zodat niemand mijn berichten kan lezen (of sterker nog, manipuleren). Dit is gedaan met algoritmes die zorgvuldig zijn gemaakt in het vakgebied van *cryptografie*.

Versleuteling is zoiets als jouw data opslaan in een veilige kluis. Alleen de correcte toegangscode kan het kluisje openen, en zonder de code krijgt niemand de kluis open, zelfs als men dynamiet gebruikt. Een cryptografische kluis is gemaakt van wiskundig materiaal, en dynamiet is alles wat computers kunnen doen. We hebben een goed begrip van wat soort wiskundige materialen we kunnen gebruiken zodat alle dynamiet ter wereld onze kluis niet kan beschadigen.

Misschien denk je nu: maar dit geldt niet alleen voor de smartphone, cryptografie is nodig voor alle electronische communicatie. Maar de smartphone is voor mij een voorbeeld van technologie die 50 jaar geleden alleen de droom van visionairen was, 30 jaar geleden een vrij onpraktische curiositeit was, maar sindsdien zo enorm technisch verbeterd is dat het onze hele wereld heeft veranderd.

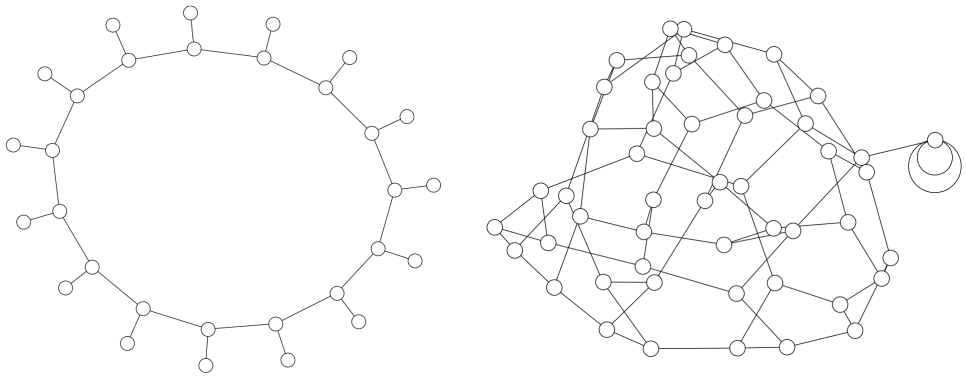
Een andere soort technologie die zelfs meer effect op onze samenleving zou kunnen hebben is *kwantumcomputing*. Deze term verwijst naar een nieuw type computers die de gebruikelijke 1's en 0's van klassieke computers op een andere manier *zouden kunnen* manipuleren. Maar we *kennen* al sommige effecten die kwantumcomputing zou hebben. Tegenwoordig wordt er massaal geïnvesteerd om dit soort technologie werkelijkheid te maken.

Het slechte nieuws is dat als de kwantum computers daadwerkelijk zouden bestaan, onze huidige cryptografie niet voldoende zou zijn. In de analogie van een kluis, zouden kwantum computers geen krachtiger dynamiet zijn, maar een totaal nieuwe stof. En daarmee kunnen wij de allersterkste kluizen openmaken.

Post-kwantum cryptografie houdt zich bezig met nieuwe wiskundige problemen die ons zouden helpen met het bouwen van *kwantum-veilige* kluizen. Tegenwoordig zijn de cryptografen bezig met het voorbereiden van nieuwe kluizen: we hebben mogelijke nieuwe materialen bepaald (bijvoorbeeld roosters en hashfuncties) en bereiden nu de beste ontwerpen voor (die dan door overheden en verschillende bureaus worden gestandaardiseerd). Wanneer dat klaar is, moeten we nog de nieuwe kluizen uitgebreid testen en zorgen dat ze wijdverspreid gebruikt worden. Al deze taken vereisen veel inspanning.

Maar we moeten meer materialen blijven onderzoeken. Dit is omdat verschillende mensen verschillende behoeften hebben voor de afmetingen van hun kluizen (cryptografische protocollen). En omdat er altijd de kans is dat een nieuwe cryptografische aanval (ander soort dynamiet) onze kluizen kan verzwakken.

In dit proefschrift richten wij ons op één soort materiaal, *isogenieën van elliptische krommen*. We kunnen denken aan elliptische krommen als simpele knopen (met rijke algebraïsche structuur), en een isogenie is een zijde (een lijntje) tussen twee knopen, gelabeld door een vast priemgetal ℓ (de *graad* van de isogenie). De ontstane *grafen van isogeniën* kunnen verschillende structuren hebben:



De graaf aan de linkerkant is een voorbeeld van een *vulkaan van isogenieën*: een reguliere, symmetrische graaf. De graaf aan de rechterkant heeft geen van de symmetrieën, en heeft de bijzondere *rapid-mixing* eigenschap dat als je met een knoop begint en 7 willekeurige *stappen* zet, je bij elke andere knoop in de graaf met ongeveer gelijke kans uitkomt. Dit is duidelijk niet waar aan de linkerkant.

Cryptografen hebben protocollen gemaakt van beiden. Aan de rechterkant, als je twee willekeurige knopen kiest, is het vinden van een verbindende route in de graaf vermoedelijk moeilijk. In de praktijk heeft de graaf niet 50 knopen maar ongeveer 2^{256}. Een route zoeken in dat soort graaf is zoals een route zoeken in een bos op een maanloze nacht: je ziet de knopen in jouw buurt, maar niet het hele bos. Een van de hoofdstukken van dit proefschrift gaat over dit soort grafen en hun wiskundige eigenschappen.

De grafen aan de linkerkant kunnen we ook gebruiken in de cryptografie. We kunnen meerdere zulke grafen maken met *dezelfde elliptische krommen* voor priemgetallen $\ell = 3, 5, 7, \ldots$ Meestal zien deze eruit als een cykel (zonder de uitstekende zijden). En voor elk priemgetal is de volgorde van de knopen anders. Dus, we kunnen een paar stappen nemen in de graaf voor $\ell = 3$, daarna switchen naar de graaf voor $\ell = 5$, nemen hier een paar stappen, enzovoort voor een aantal priemgetallen. Verrassend genoeg heeft de resulterende route dezelfde eigenschappen als de graaf aan de rechterkant: met een klein aantal stappen (een paar honderd) kunnen we alle knopen bereiken (alle 2^{256} knopen). Protocollen gebaseerd op dit soort constructie zijn de focus van dit proefschrift.

Een van de protocollen die we zo grofweg hebben beschreven is CSIDH ("seaside") van Castryck, Lange, Martindale, Panny, en Renes (2018). We nemen hun bouwtekening voor de kluis en bestuderen die van meerdere kanten.

Ten eerste bestuderen we sommige van de wiskundige problemen die ze gebruiken: namelijk het *decisional Diffie-Hellman problem* (DDH). We vinden dat het CSIDH protocol veilig is, maar identificeren een verrassende eigenschap in gerelateerde protocollen, en veel gevallen waarin het DDH probleem niet moeilijk is. Vervolgens bestuderen wij implementaties: we implementeren het protocol op een manier dat de *timing* van de berekening niets zegt over de geheimen gebruikt in het protocol. Dit is een essentiële eigenschap voor cryptografische protocollen in de praktijk; onze software, CTIDH (uitgesproken zoals "sea tide"), is het snelste onder *constant-time* implementaties van CSIDH.

Uiteindelijk bestuderen wij de fysieke veiligheid van de kluis: we stellen ons voor dat iemand toegang heeft tot een apparaat dat het protocool uitvoert, en goed getimede fouten kan forceren. We laten zien dat fouten die de richting van de stappen veranderen (met de klok mee of tegen de klok in aan de linkerkant) te veel informatie lekken, die we dan kunnen gebruiken om het geheim te reconstrueren.

Grafen van isogeniën zijn een mogelijk materiaal voor het bouwen van post-quantum kluizen. Dit proefschrift verdiept het inzicht in zowel de wiskundige veronderstellingen (het materiaal) als de implementatie (hoe de kluis gebouwd zal worden).

Summary

I come from a generation that grew up with the smartphone. The small gadget has been in my hand for almost 20 years, and has been my primary source of communication with the world. Not just talking to my family and friends, but my affairs are increasingly handled through my phone.

I opened my bank account by walking into the branch office near my home, but that office is long gone and all my finances are handled via the app. The government in the Netherlands also uses an electronic identity, so any official business the government wants with me happens over the internet. And while my first tax experience in the Netherlands was with the legendary 80-page paper M-form, all my other tax returns were handled electronically, and I would not know how to do my civic duty any other way.

In a world in which it is impossible to disconnect and we are forced to handle all our matters online, we need protections for our privacy, as well as protection against abuse and interference by malicious actors. I trusted that the workers in the marble building of my bank were truly authorized to handle my finances, and what we arranged was to remain my private matter. When I talk to my family sitting in the garden, I can look around and decide whether I can share intimate details of my life or whether somebody is listening.

When I am connected to the internet, I want the same guarantees. I want my communication to be *encrypted*: protected with an extra coating that guarantees that nobody else can read (or even stronger protections, so nobody can manipulate) my messages. This extra layer is achieved via carefully crafted algorithms. And figuring out what these algorithms should be is the art of *cryptography*.

One might think of cryptography as putting our information in a strong safe. The correct combination will open the safe, but without the correct code, neither a sledgehammer nor dynamite will help you see what is inside. The cryptographic safe is built from mathematical problems, and the dynamite is anything computers can do. We know what mathematical material to use to build our safes so that there is not enough dynamite in the world to damage it.

You might think: all this is not unique to the smartphone, we need cryptography for any kind of electronic communication. But the smartphone, to me, is an example of technology that was dreamt of by visionaries 50 years ago, only existed as a barely-practical curiosity 30 years ago, underwent a massive technological boost and by now has changed the world we live in.

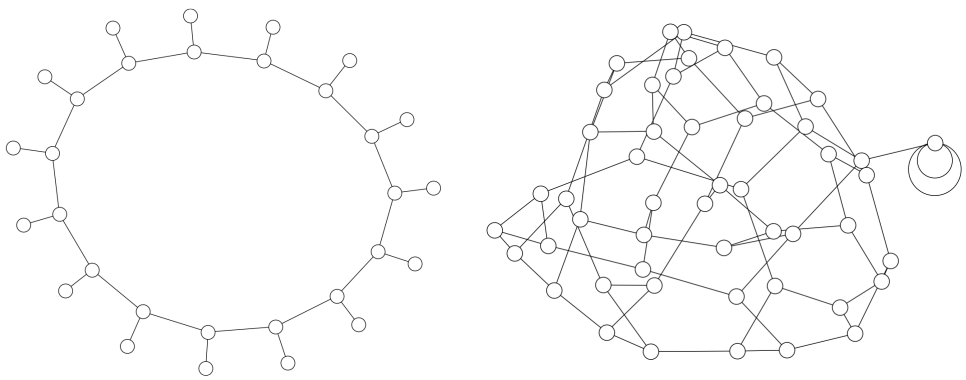
Another such novel technology that could have even more impact on our lives is *quantum computing*. This term refers to a new kind of computers that will use different ways to manipulate the ones and zeros that classical computers use. And we already *know* what some of the impact of quantum computing would be – bringing about many exciting possibilities. Massive amounts of both public and private money are being spent on making this technology real.

The bad news is that should quantum computers become real technology, our current cryptography will not be enough. In the safe analogy, quantum computers would not add extra potency to the dynamite already out there, but rather produce a new substance. Armed with this new substance, we would see the strongest safes opening before our eyes.

Post-quantum cryptography is the search for new mathematical problems that will help us build *quantum-safe* safes. Currently, the cryptographic community is working on preparing the new safes: we have identified potential material (for instance lattices and hash functions) and are in the process of choosing the best designs (standardized by various government agencies and accepted widely by the community). When that is done, we need to test the new safes, and make sure they are deployed widely. All of these tasks require significant effort.

But we continue to explore more materials. This is because different people have different requirements for the size and shape of their safes (cryptographic protocols). And because there is always the chance that new attack avenues (different dynamite, or better algorithms) might weaken our preferred safe.

In this thesis, we focus on a particular material, *isogenies of elliptic curves*. We can abstract elliptic curves as simple dots (with extremely rich algebraic structure), and an isogeny is an edge (line segment) connecting two dots, with labels some fixed prime ℓ (the *degree* of the connecting isogeny). The resulting *isogeny graphs* can have very different structure:



The graph on the left is an example of an *isogeny volcano*: a regular and symmetric graph. The graph on the right exhibits none of this symmetry, and even has the special *rapid mixing* property that if we start with one dot, then *walk* following 7 edges at random, will reach any other dot in the graph with roughly the same probability. This does not work for the graph on the left.

Cryptographers have built protocols from either of the situations. On the right hand side, if we select two random vertices, finding a path between them in the graph is believed to be a hard problem. In practice, the graph will not have 50 vertices as displayed, but the order of 2^{256}. Finding a path in the graph is like finding a path in a forest on a moonless night: we can see the neighboring vertices, but not the whole area. One of the chapters of this thesis is devoted to studying these graphs and their mathematical properties.

The graphs on the left can also be turned into cryptographic protocols. We can construct these graphs for the *same set of elliptic curves* for primes $\ell = 3, 5, 7, \ldots$. In most cases, these graphs look like cycles (without the edges sticking out). And for each different prime, the order of the vertices on the cycle will be very different. Hence, one can take a few steps in the graph for $\ell = 3$, jump to the graph for $\ell = 5$, take a few steps there, and repeat this for a number of primes. Miraculously, the resulting *walk* will have the same properties as in the graph on the right: we will be able to reach any elliptic curve with just a *few* steps (a few hundred steps, which is very small compared to having 2^{256} vertices). Protocols based on this construction are the main focus of this thesis.

One instantiation of the setup so roughly described above is the cryptographic protocol CSIDH (pronounced like "seaside") by Castryck, Lange, Martindale, Panny, and Renes (2018). We take their schematics for a safe, and study them from several sides.

First, we examine the hardness of the mathematical problems they base their protocol on: namely the *decisional Diffie–Hellman problem* (DDH). We find that the CSIDH protocol is secure, but identify a surprising structure in related protocols, and find many instances in which the DDH problem is not hard.

Next, we consider implementation security: we implement the protocol in a way such that the *timing* of the computation does not reveal any information about the secrets. This is an essential requirement for cryptographic protocols in practice; our software, which we call CTIDH (pronounced "sea tide"), produces record speeds among the constant-time implementations of CSIDH.

Finally, we consider the *physical* security of the safe: we suppose somebody has access to a device running the computation, and is able to insert a well-timed error in the computation. We show that errors that flip the direction of the step (clockwise or counterclockwise in the cycle on the left) leak too much information, which can be used to reconstruct the secret.

Isogeny graphs are one possible material from which to build post-quantum safes. This thesis deepens the understanding of both the mathematical assumptions (the material) and the implementation (how the safe should be built).

Curriculum Vitae

Jana Sotáková was born on a Sunday, on April 25, 1993, in Brno, Czech Republic. After spending her childhood in Blansko, she attended the mathematical class at the Grammar School at třída Kapitána Jaroše 14, Brno. She competed in the final of the 2010 Youth Olympic Games in Singapore in 100m hurdles, and also competed at the World Junior Championships in 2012.

In 2012 she started attending Masaryk University in Brno. She spent one semester of her bachelor's program at Leiden University as part of the Erasmus+ mobility scheme. Her bachelor thesis was supervised by her long-term mentor, prof. Radan Kučera. She graduated *cum laude* in 2015, and received a prize of the Head of the Department of Mathematics for her academic achievements. In 2013, she competed in the European Under 23 Championships in 100m hurdles.

In 2015, she started the ALGANT masters program, fully supported by the scholarship of the ALGANT Consortium. She spent her first year at the University of Regensburg and the second year at Leiden University. She wrote her thesis under the supervision of dr. Marco Streng, and graduated *cum laude/Sehr Gut*. Unfortunately, Jana did not compete during those years.

Afterwards, Jana spent 2 years in the graduate program in mathematics at the University of California, Berkeley, supported by the prestigious Fulbright Scholarship in 2017-2018. In Berkeley, Jana improved her personal best in high jump. After discovering her passion for isogenies, Jana moved back to the Netherlands.

There she started a PhD program in 2019 at the University of Amsterdam, supervised by prof. Christian Schaffner, prof. Serge Fehr, and dr. Peter Bruin. Her PhD is funded by the Quantum Software Consortium. Throughout her academic career, Jana most enjoyed interacting with other researchers at conferences and summer schools in many beautiful locations. Despite the impact the COVID-19 pandemic had on Jana during her PhD, she improved her PB in javelin throw.

Jana lives in Oegstgeest with her partner Stefan and knuffelbeer Claude. She is a hiker, an avid video game player, and an even more enthusiastic bachata dancer. In her everyday life she speaks Czech, English, Dutch, and Spanish.

Titles in the ILLC Dissertation Series:

ILLC DS-2019-06: **Peter T.S. van der Gulik**
Considerations in Evolutionary Biochemistry

ILLC DS-2019-07: **Frederik Möllerström Lauridsen**
Cuts and Completions: Algebraic aspects of structural proof theory

ILLC DS-2020-01: **Mostafa Dehghani**
Learning with Imperfect Supervision for Language Understanding

ILLC DS-2020-02: **Koen Groenland**
Quantum protocols for few-qubit devices

ILLC DS-2020-03: **Jouke Witteveen**
Parameterized Analysis of Complexity

ILLC DS-2020-04: **Joran van Apeldoorn**
A Quantum View on Convex Optimization

ILLC DS-2020-05: **Tom Bannink**
Quantum and stochastic processes

ILLC DS-2020-06: **Dieuwke Hupkes**
Hierarchy and interpretability in neural models of language processing

ILLC DS-2020-07: **Ana Lucia Vargas Sandoval**
On the Path to the Truth: Logical & Computational Aspects of Learning

ILLC DS-2020-08: **Philip Schulz**
Latent Variable Models for Machine Translation and How to Learn Them

ILLC DS-2020-09: **Jasmijn Bastings**
A Tale of Two Sequences: Interpretable and Linguistically-Informed Deep Learning for Natural Language Processing

ILLC DS-2020-10: **Arnold Kochari**
Perceiving and communicating magnitudes: Behavioral and electrophysiological studies

ILLC DS-2020-11: **Marco Del Tredici**
Linguistic Variation in Online Communities: A Computational Perspective

ILLC DS-2020-12: **Bastiaan van der Weij**
Experienced listeners: Modeling the influence of long-term musical exposure on rhythm perception

ILLC DS-2020-13: **Thom van Gessel**
Questions in Context

ILLC DS-2020-14: **Gianluca Grilletti**
Questions & Quantification: A study of first order inquisitive logic

ILLC DS-2020-15: **Tom Schoonen**
Tales of Similarity and Imagination. A modest epistemology of possibility

ILLC DS-2020-16: **Ilaria Canavotto**
Where Responsibility Takes You: Logics of Agency, Counterfactuals and Norms

ILLC DS-2020-17: **Francesca Zaffora Blando**
Patterns and Probabilities: A Study in Algorithmic Randomness and Computable Learning

ILLC DS-2021-01: **Yfke Dulek**
Delegated and Distributed Quantum Computation

ILLC DS-2021-02: **Elbert J. Booij**
The Things Before Us: On What it Is to Be an Object

ILLC DS-2021-03: **Seyyed Hadi Hashemi**
Modeling Users Interacting with Smart Devices

ILLC DS-2021-04: **Sophie Arnoult**
Adjunction in Hierarchical Phrase-Based Translation

ILLC DS-2021-05: **Cian Guilfoyle Chartier**
A Pragmatic Defense of Logical Pluralism

ILLC DS-2021-06: **Zoi Terzopoulou**
Collective Decisions with Incomplete Individual Opinions

ILLC DS-2021-07: **Anthia Solaki**
Logical Models for Bounded Reasoners

ILLC DS-2021-08: **Michael Sejr Schlichtkrull**
Incorporating Structure into Neural Models for Language Processing

ILLC DS-2021-09: **Taichi Uemura**
Abstract and Concrete Type Theories

ILLC DS-2021-10: **Levin Hornischer**
Dynamical Systems via Domains: Toward a Unified Foundation of Symbolic and Non-symbolic Computation

ILLC DS-2021-11: **Sirin Botan**
Strategyproof Social Choice for Restricted Domains

ILLC DS-2021-12: **Michael Cohen**
Dynamic Introspection

ILLC DS-2021-13: **Dazhu Li**
Formal Threads in the Social Fabric: Studies in the Logical Dynamics of Multi-Agent Interaction

ILLC DS-2021-14: **Álvaro Piedrafita**
On Span Programs and Quantum Algorithms

ILLC DS-2022-01: **Anna Bellomo**
Sums, Numbers and Infinity: Collections in Bolzano's Mathematics and Philosophy

ILLC DS-2022-02: **Jan Czajkowski**
Post-Quantum Security of Hash Functions

ILLC DS-2022-03: **Sonia Ramotowska**
Quantifying quantifier representations: Experimental studies, computational modeling, and individual differences

ILLC DS-2022-04: **Ruben Brokkelkamp**
How Close Does It Get?: From Near-Optimal Network Algorithms to Suboptimal Equilibrium Outcomes

ILLC DS-2022-05: **Lwenn Bussière-Carae**
No means No! Speech Acts in Conflict

ILLC DS-2022-06: **Emma Mojet**
Observing Disciplines: Data Practices In and Between Disciplines in the 19th and Early 20th Centuries

ILLC DS-2022-07: **Freek Gerrit Witteveen**
Quantum information theory and many-body physics

ILLC DS-2023-01: **Subhasree Patro**
Quantum Fine-Grained Complexity

ILLC DS-2023-02: **Arjan Cornelissen**
Quantum multivariate estimation and span program algorithms

ILLC DS-2023-03: **Robert Paßmann**
Logical Structure of Constructive Set Theories

ILLC DS-2023-04: **Samira Abnar**
Inductive Biases for Learning Natural Language

ILLC DS-2023-05: **Dean McHugh**
Causation and Modality: Models and Meanings

ILLC DS-2023-06: **Jialiang Yan**
Monotonicity in Intensional Contexts: Weakening and: Pragmatic Effects under Modals and Attitudes

ILLC DS-2023-07: **Yiyan Wang**
Collective Agency: From Philosophical and Logical Perspectives

ILLC DS-2023-08: **Lei Li**
Games, Boards and Play: A Logical Perspective

ILLC DS-2023-09: **Simon Rey**
Variations on Participatory Budgeting

ILLC DS-2023-10: **Mario Giulianelli**
Neural Models of Language Use: Studies of Language Comprehension and Production in Context

ILLC DS-2023-11: **Guillermo Menéndez Turata**
Cyclic Proof Systems for Modal Fixpoint Logics

ILLC DS-2023-12: **Ned J.H. Wontner**
Views From a Peak: Generalisations and Descriptive Set Theory

ILLC DS-2024-01: **Jan Rooduijn**
Fragments and Frame Classes: Towards a Uniform Proof Theory for Modal Fixed Point Logics

ILLC DS-2024-02: **Bas Cornelissen**
Measuring musics: Notes on modes, motifs, and melodies

ILLC DS-2024-03: **Nicola De Cao**
Entity Centric Neural Models for Natural Language Processing

ILLC DS-2024-04: **Ece Takmaz**
Visual and Linguistic Processes in Deep Neural Networks: A Cognitive Perspective

ILLC DS-2024-05: **Fatemeh Seifan**
Coalgebraic fixpoint logic Expressivity and completeness result