BUFF Transform for Isogeny-Based SQISign Digital Signature

by

Toma Diaconescu-Grabari

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Science

in

Mathematics

Abstract

This thesis covers the mathematics of isogeny-based cryptography and presents the SQISign digital signature. SQISign is a digital signature, submitted to NIST's post-quantum cryptography standardization program. The thesis also covers the security notions for digital signatures called Beyond UnForgeability Features (BUFF) and discusses these security notions within the context of SQISign. A general transformation for digital signatures, called the BUFF transform, can be used to achieve all BUFF notions. This thesis ends with an application of the BUFF transform to SQISign and analyzes the performance impact.

Acknowledgments

I would like to thank my family for their continued support in all aspects of my life.

I would also like to thank my supervisor Daniel Panario for his guidance and insights during the thesis process.

Special thanks to Gustavo Zambonin for his support in setting up the implementation of SQISign, and Professors Monica Nevins and Colin Ingalls for their valuable corrections/comments to my thesis.

Contents

List of Tables

List of Figures

List of Appendices

List of Algorithms

Chapter 1

Introduction

Cryptography, in a broad sense, is the field of methods to communicate information in a specific manner that only intended recipients can understand. It has a long history, dating as far back to Egypt in 1900 BC. For "A Brief History of Cryptography" see this website[1].

Modern-day cryptography is based on what are known as cryptographic primitives such as digital signatures, one-way hash functions, and public key cryptography; these are based on mathematical problems that are computationally hard. Hard here means an adversary attempting to steal information from a secret communication should fail given only a reasonable amount of time and memory.

Two mathematical problems that are (believed to be) difficult for classical computers are the discrete logarithm problem and the integer factorization problem.

Problem 1. *(Discrete Logarithm Problem)* Let G be a group, and $g \in G$. Given $y = g^x$, for some (unknown) positive integer x, find x.

[1] `http://www.cypher.com.au/crypto_history.htm`

Example 1.0.1. Let G be the group $\mathbb{Z}/pZ$, where $p = 2053$. Given $g = 2$, and $y = g^x = 1608 \pmod{2053}$, find x.

Solution.

$x = 1234$.

Problem 2. *(Integer Factorization Problem)* Given a positive integer, find its prime factorization.

Example 1.0.2. Given $n = 15775087$, find its prime factorization.

Solution.

$n = 3793 \times 4159$.

Note that these two problems are easy for small groups G (in the DLP) and small integers (in the IFP), but hard for parameters[2] used today. These two problems are of major importance to modern-day cryptography. They underly the security of protocols such as the Diffie-Hellman key exchange [24], the RSA cryptosystem [51], and many more.

With the progress of quantum mechanics, physicists have speculated on the possibility of constructing quantum computers with different capabilities to those of classical computers. One of the consequences of these different capabilities was shown in 1994 by Shor [55]. Shor developed what is known now as Shor's Algorithm solving the discrete logarithm and integer factorization problems on a (large enough) quantum computer. Indeed, quantum computers are no longer just speculation, but also a reality. In 1998, the first quantum computer consisting of 2 qubits was created by Chuang, Gershenfeld, and Kubinec. One of the leading quantum computer developers at the time of this thesis writing is IBM, with

[2]Difficulty can be quantified, and is done in the field of theoretical computer science.

notable achievements such as “IBM Osprey” with 433 qubits, “IBM Condor”, and “IBM Heron”.

It is the widespread belief that quantum computers will eventually reach sizes capable of implementing Shor’s algorithm and breaking much of modern-day protocols. Cryptographers have been developing and studying different mathematical problems in hopes of finding problems that are both difficult on classical and quantum computers. The National Institute of Standards and Technology (NIST) has facilitated this process by creating the Post-Quantum Cryptography Standardization program. The program was announced in 2016 and finished accepting proposals in 2017. Since then, the program has been continuously updated.

The NIST program consists of several categories of submitted algorithms. These are:

- lattice-based cryptography,
- code-based cryptography,
- hash-based cryptography,
- multivariate cryptography,
- braid-group cryptography, and
- isogeny-based cryptography.

The focus of this thesis is on *isogeny-based cryptography*, and more specifically the SQISign digital signature [29].

This thesis is organized as follows. Chapter 2 introduces the mathematical background necessary for understanding the subsequent chapters. Chapter 3 discusses the history of isogeny-based cryptography and several protocols. Chapter 4 discusses the cryptanalysis of problems in isogeny-based cryptography. Chapter 5 introduces the SQISign digital algorithm.

Our contribution is in Chapter 6, where we introduce the notion of Beyond UnForgiability Features (BUFF) security, and discuss the application of the BUFF transform to SQISign to achieve full BUFF security.

Chapter 2

Preliminaries

2.1 Cryptography

We define some cryptography terms that are related to understanding the rest of this thesis. We provide more general, as opposed to rigorous, definitions. For the interested reader, more rigorous definitions can be found in [63].

Public key cryptography (also called asymmetric cryptography) is a subfield of cryptography where users participating in a public-key cryptosystem use a public key and a secret key. Public key cryptography splits into two main uses: public key encryption and digital signatures.

Public key encryption: Consider two parties, Alice and Bob. Alice has a public key-secret key pair $(\mathtt{pk}_A, \mathtt{sk}_A)$, and Bob has the pair $(\mathtt{pk}_B, \mathtt{sk}_B)$. Bob wishes to send a private message $\mathtt{msg}$ to Alice. To do so, Bob encrypts the message using Alice's public key $\mathtt{pk}_A$, and the function $\mathtt{Encrypt}(\mathtt{msg}, \mathtt{pk}_A)$, then sends it to Alice. Only Alice should be able to

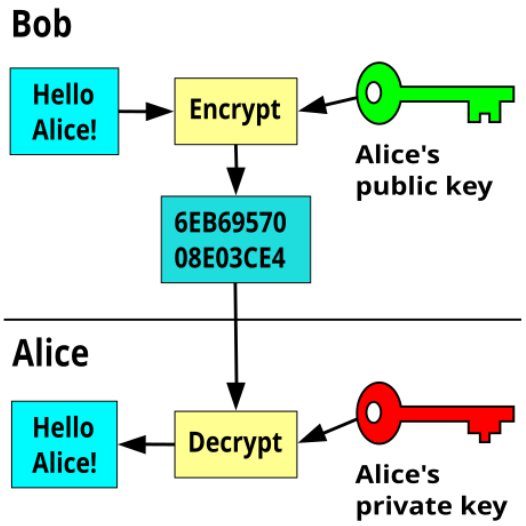

Figure 2.1: Diagram depicting public key encryption of the message "Hello Alice!", being send from Bob to Alice. Original illustration by David Göthberg.

decrypt the transmitted message because only she knows the secret key $\mathtt{sk}_A$.

Digital Signatures: Consider again two parties Alice and Bob. Alice wants to send a message to Bob, and Bob wants to verify that Alice, not a different party, indeed sent the message. Alice has a public key-secret key pair $(\mathtt{pk}_A, \mathtt{sk}_A)$ and a message $\mathtt{msg}$. She signs the message using her secret key, $\sigma = \mathtt{sign}(\mathtt{msg}, \mathtt{sk}_A)$. Anyone (in particular Bob) who knows Alice's public key can verify the signature, $\mathtt{Verify}(\mathtt{msg}, \sigma, \mathtt{pk}_A)$.

Definition 2.1.1. A *hash function* is a function taking in an arbitrary input and producing a fixed-length output. A hash function must satisfy certain properties:

- *Deterministic:* The same input always produces the same hash (output).
- *Pre-Image Resistance:* It should be easy to compute the hash of an input, but extremely difficult to compute an input given a hash.

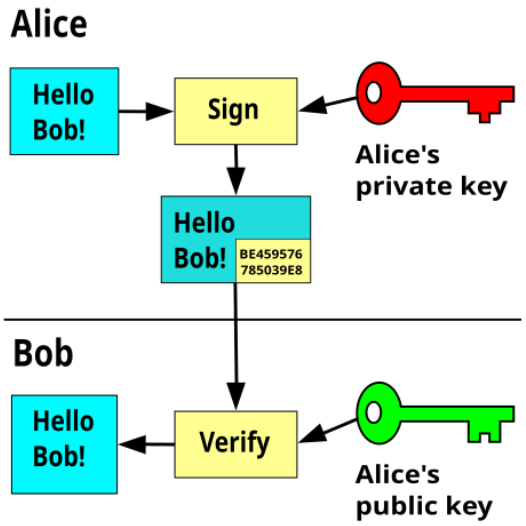

Figure 2.2: A diagram depicting a digital signature of the message "Hello Bob!" by Alice, which is then verified by Bob. From author FlippyFlink on Wikipedia.

- *Collision Resistance:* It should be extremely difficult to find two different inputs that produce the same hash.

Definition 2.1.2. A *proof of knowledge* is an interactive proof between two parties, a prover and a verifier. The goal of a proof of knowledge protocol is for the prover to prove knowledge of something to the verifier.

Definition 2.1.3. A Σ*-protocol* is a proof of knowledge protocol between two parties, a prover and a verifier. The goal is for the prover to convince the verifier that they know some secret w corresponding to an element x. The protocol consists of three steps: commitment, challenge and response.

1. (*Commitment*, first move by prover) The prover sends a commitment string com which is a function of x and w.

2. (*Challenge*, second move by verifier) The verifier sends a random λ-bit string chall to the prover. The constant λ corresponds to the security level of the Σ-protocol.

3. (*Response*, third move by prover) The prover responds with resp, which is a function of x, w and chall

The verifier decides to accept or reject based on the data they have seen, $x, \text{com}, \text{chall}$ and resp.

Example 2.1.4. Suppose Peggy wants to prove knowledge of some secret integer w to Victor. They can proceed as follows[1]:

1. *(Setup)* Peggy and Victor agree on: (1) A large prime p, and (2) a generator g of a cyclic group G of order p.

2. Peggy computes $x = g^w \pmod{p}$.

3. *(Commitment)* Peggy chooses a random number r_{com}, and computes $C = g^{r_{\text{com}}} \pmod{p}$. Peggy sends C to Victor.

4. *(Challenge)* Victor sends a random number r_{chall} to Peggy.

5. *(Response)* Peggy computes $s = r_{\text{com}} + r_{\text{chall}} \cdot w \pmod{p}$. Peggy sends s to Victor.

6. *(Verification)* Victor checks that $g^s = C \cdot x^{r_{\text{chall}}} \pmod{p}$ is true.

Victor should be confident that Peggy knows the secret integer w such that $x = g^w$, as Peggy used it to compute the response, and computing such a response is believe to be as

[1]This protocol shouldn't be implemented in practice, as the security is based on the DLP, which as mentioned earlier, will be broken by quantum computers.

difficult as solving the discrete logarithm problem (Problem 1). Furthurmore, Victor does not gain much information about the secret integer w.

Notation 2.1.5.

- $a||b$ means "a concatenate with b".
- A function with domain $\{0,1\}^*$ means the function takes an arbitrary length string of 0s and 1s.

Definition 2.1.6. [30] The *Fiat-Shamir transform* turns a Σ-protocol into a digital signature scheme. The transform makes use of a hash function (Definition 2.1.1) $\mathtt{H}: \{0,1\}^* \to \{0,1\}^\lambda$. It consists of three non-interactive steps:

1. *Key Generation:* A function $\text{Gen}(1^\lambda) \to (\mathtt{sk}, \mathtt{pk})$ which randomly outputs a signing key $\mathtt{sk}$ and a corresponding verification key $\mathtt{pk}$.
2. *Signing:* A function $\text{Sign}(\mathtt{sk}, \mathtt{msg})$ where the commitment, challenge, and response steps from the Σ-protocol are all done by the prover. The commitment stage is left unchanged, com is the output of some function of $\mathtt{sk}$ and $\mathtt{pk}$. The challenge stage becomes the hash $\text{chall} = \mathtt{H}(\text{com}||\mathtt{msg})$. The response is also left unchanged, resp is the output of some function of $\mathtt{sk}, \mathtt{pk}$, and chall. The final signature output is $\sigma = (\text{com}, \text{resp})$
3. *Verification:* A function $\text{Verify}(\mathtt{pk}, \mathtt{msg}, \sigma)$ which outputs 1 if valid, and 0 otherwise. The function first checks that $\text{chall} = \mathtt{H}(\text{com}||\mathtt{msg})$, then runs the same verification function from the Σ-protocol, dependent on $x, \text{com}, \text{chall}$ and resp.

The main difference between the Σ-protocol and its Fiat-Shamir transform is in the challenge (second move by verifier) in the Σ protocol. Instead of the verifier randomly choosing a string chall, a hash function takes the role of randomness, removing the need for interaction between the two parties.

Example 2.1.7. We apply the Fiat-Shamir transform to Example 2.1.4. Suppose Peggy wishes to sign the message `Hello Bob`. The main modification is in the challenge step: *(Challenge)* Peggy computes the challenge $r_{\text{chall}} = \mathtt{H}(C\,||\,\mathtt{msg})$. For example, if we were to use the `SHA-256` hash function [48], $p = 2053$, $g = 2$, $r_{\text{com}} = 2024$, then $C = g^{r_{\text{com}}} = 1133 \pmod{2053}$ and

$$
\begin{aligned}
r_{\text{chall}} &= \mathtt{SHA\text{-}256}(\text{`}\mathtt{1133Hello\ Bob}\text{'}) \\
&= \mathtt{3934c1b1248f23470881a873f66ee8666c0591e545eaf9cdd7440} \\
&\quad \mathtt{ad93d07489b} \quad \text{(In hexadecimal)} \\
&= 258750452294202767635986694793395126398816186359674074071897441022149860578 83 \quad \text{(In decimal)} \\
&= 1533 \pmod{2053}.
\end{aligned}
$$

Definition 2.1.8. A digital signature satisfies existential unforgeability under chosen message attack (EUF-CMA) if the following scenario is computationally hard for an adversary. A challenger generates a valid key pair $(\mathtt{sk}, \mathtt{pk})$ using the key generation algorithm of the digital signature. The adversary then asks the challenger to produce signatures $(\sigma_1, \ldots, \sigma_n)$ to messages $(\mathtt{msg}_1, \ldots, \mathtt{msg}_n)$ of the adversary's choosing. The adversary is then asked to produce a signature for a new message `msg` valid for the challenger's public key `pk`, hence

forging a signature.

Definition 2.1.9. [14, Section 9.1.1.2] A Σ-protocol satisfies *special soundness* if a cheating prover who only knows some element x and not its corresponding secret w, fails with overwhelming probability.

Definition 2.1.10. [14, Section 9.1.1.4] A Σ-protocol satisfies *weak honest-verifier zero-knowledge* (wHVZK) if there exists a polynomial-time algorithm that takes input an arbitrary element x, without its corresponding secret w, and outputs a valid $(\text{com}, \text{chall}, \text{resp})$ transcript.

Remark 2.1.11. It may seem at first that if a Σ-protocol satisfies wHVZK, then it cannot satisfy special soundness. We note that the polynomial-time algorithm to produce a valid transcript does not partake in the interaction required by a Σ-protocol. Indeed the algorithm chooses the random challenge as desired. The point of wHVZK is that even without knowledge of the secret w corresponding to an element x, it is possible to create a valid transcript. This means that no information about the secret w is leaked during the Σ-protocol, hence zero-knowledge. Although, a real interaction, where the challenge is truly random should require the prover to have knowledge of the secret to produce the response.

Example 2.1.12. We continue with Example 2.1.4. Given for example, $x = g^w = 1000 \pmod{p}$, we can create a valid looking transcript without knowing the corresponding secret w.

1. Select a random challenge r_{chall}.

2. Randomly choose a response s.

3. Compute the commitment $C = g^s \cdot x^{-r_{\text{chall}}} \pmod{2053}$.

Then the transcript (C, r_{chall}, s) is valid as

$$C \cdot x^{r_{\text{chall}}} = g^s \cdot x^{-r_{\text{chall}}} \cdot x^{r_{\text{chall}}} = g^s \pmod{p}.$$

Definition 2.1.13. A Σ-protocol satisfies security under passive impersonation attacks (IMP-PA) if an adversary is given an element x, without the corresponding secret w, and several valid transcripts $(\text{com}, \text{chall}, \text{resp})$ for x, still fails to convince a verifier that they know the secret w.

Definition 2.1.14. Let $f : \mathbb{C} \to \mathbb{C}$ and $g : \mathbb{R} \to \mathbb{R}$ be functions. We write $f = O(g)$ if there exists $M \in \mathbb{R}_{>0}$ and there exists $x_0 \in \mathbb{R}$ such that

$$|f(x)| \leq Mg(x) \quad \text{for all } x \geq x_0.$$

If the above holds, then f is big O of g.

Example 2.1.15. $f(x) := T(x) = 2 \cdot x \cdot \log(\pi x) + \log(x) - 2 \cdot x \in O(x \cdot \log(x))$. This follows from $M = 3$, $g(x) = x \cdot \log(x)$, and $x_0 = 1.87$.

Definition 2.1.16. Let $f : \mathbb{C} \to \mathbb{C}$ and $g : \mathbb{R} \to \mathbb{R}$ be functions. We write $f = o(g)$ if for every $\varepsilon \in \mathbb{R}_{>0}$, there exists $x_0 \in \mathbb{R}$ such that

$$|f(x)| \leq \varepsilon g(x) \quad \text{for all } x \geq x_0.$$

If the above holds, then f is little o of g.

Definition 2.1.17. We define the L-notation,

$$L_n(\alpha, c) = e^{(c+o(1))(\log n)^{\alpha}(\log\log n)^{1-\alpha}}$$

where $c \in \mathbb{R}_{>0}$ and $\alpha \in [0,1]$. This notation is useful for describing algorithms with sub-exponential[2] running time.

Definition 2.1.18. We define

$$\widetilde{O}(f) = O(f \cdot \log(f)^k)$$

where k is some constant.

Example 2.1.19. From Example 2.1.15, $T(x) \in \widetilde{O}(x)$, where $f(x) = x$ and $k = 1$.

Definition 2.1.20. An *oracle* is a black box that solves specific problems in polynomial time.

2.2 Quaternions, Lattices, and Orders

Notation 2.2.1.

- K denotes a field.
- Sometimes, "is equal to", $=$, is written instead of "is isomorphic to", $\cong$. It should be understood from the context.

[2]We do not define the terms constant, logarithmic, polynomial, exponential, etc. in this thesis, see [1] for details on computational complexity theory.

- $\langle \alpha \rangle$ denotes the ideal (in a ring understood by context) generated by α.

Definition 2.2.2. [18, Definition 28] A *quaternion algebra* is of the form

$$K = \mathbb{Q} + i\mathbb{Q} + j\mathbb{Q} + k\mathbb{Q},$$

where the generators $\{1, i, j, k\}$ satisfy

$$i^2, j^2 \in \mathbb{Q}, \qquad i^2, j^2 < 0 \qquad ij = -ji = k.$$

Example 2.2.3. For SQISign, we are interested in the quaternion algebra over $\mathbb{Q}$ with $i^2 = -1$ and $j^2 = -p$ (where $p \equiv 3 \pmod 4$). We denote this quaternion algebra $B_{p,\infty}$. Elements $\alpha \in B_{p,\infty}$ are represented as $(a, b, c, d, r) \in \mathbb{Z}^5$ representing

$$\alpha = \frac{a + bi + cj + dk}{r}.$$

Definition 2.2.4. [14, Section 2.4.4.1] Let $\alpha \in B_{p,\infty}$ be a quaternion written as $\alpha = \frac{a+bi+cj+dk}{r}$. Its conjugate is

$$\overline{\alpha} = \frac{a - bi - cj - dk}{r}.$$

Its reduced trace is

$$\mathrm{tr}(\alpha) = \alpha + \overline{\alpha} = \frac{2a}{r}.$$

Its reduced norm is

$$\mathrm{n}(\alpha) = \mathrm{n}(\overline{\alpha}) = \alpha\overline{\alpha} = \frac{a^2 + b^2 + p(c^2 + d^2)}{r^2}.$$

Example 2.2.5. Let $p = 7$, and $\alpha = (1,2,3,4,5) \in \mathbb{Z}^5$, or $\alpha = \frac{1+2i+3j+4k}{5} \in B_{7,\infty}$. Then α has, conjugate $\overline{\alpha} = (1,-2,-3,-4,5)$, trace $\mathrm{tr}(\alpha) = \frac{2}{5}$, and norm $\mathrm{n}(\alpha) = \frac{1+2^2+7(3^2+4^2)}{5^2} = \frac{36}{5}$.

Definition 2.2.6. [62, Definition 9.3.1] Let R be an integral domain, and K be its field of fractions. Let V be a finite-dimensional K-vector space. An R-lattice is a finitely generated R-submodule $M \subseteq V$ with $MK = V$. A $\mathbb{Z}$-lattice is a *lattice*.

Example 2.2.7. Let $R = \mathbb{Z}$, hence $K = \mathbb{Q}$. Let $V = \mathbb{Q}^2$, and $M \subseteq V$ be the submodule generated by $\{(1,0),(0,1)\}$. It is clear that $M\mathbb{Q} = \mathbb{Q}^2$, hence M is a lattice ($\mathbb{Z}$-lattice). M is the 2-dimentional integer lattice.

Definition 2.2.8. [14, Section 2.4.2.2] A matrix H is in *Hermite normal form* (HNF) if

- It is upper triangular, and any columns of zeros are located to the left.
- The leading coefficient, or pivot, of a nonzero column is always strictly below the leading coefficient of the column before it; moreover, it is positive.
- The elements to the left of pivots are zero and elements to the right of pivots are nonnegative and strictly smaller than the pivot.

A matrix A has H for HNF if H is in HNF and there exists an unimodular (integer entries and invertible) matrix U such that $UA = H$.

Example 2.2.9. The matrix

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

has

$$H = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

for Hermite normal form. It can easily be checked that $UA = H$ where,

$$U = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 1 & 1 & 1 & -2 \end{bmatrix}.$$

Definition 2.2.10. [14, Section 2.4.4.2] Let M be a lattice defined by a basis $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ of $\mathbb{Q}$-linearly independent quaternions. We define the matrix L as the matrix such that

$$(\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4) = (1\ i\ j\ k) \cdot L.$$

By abuse of notation, we call the matrix L, the lattice L, when we mean the lattice M. The dual to the lattice L is

$$L^* = \{f \in (\mathrm{span}(L))^* : \text{for any } x \in L,\ f(x) \in \mathbb{Z}\}.$$

Let L_1, L_2 be lattices. We define several operations on lattices:

- *(Equality)* L_1 and L_2 are equal if they have the same HNF.

- *(Union)* Concatenate the matrices L_1 and L_2 to obtain $L_1 || L_2$, and compute the HNF to obtain $L_1 + L_2$.
- *(Intersection)* The intersection $L_1 \cap L_2$ is the dual of $L_1^* + L_2^*$.
- *(Multiplication)* The product of two lattices is the lattice with matrix $L_1 L_2$.
- *(Right transporter, colon lattice)* The right transporter, also known as colon lattice, of L_2 with respect to L_1 is the lattice T of elements $\alpha \in B_{p,\infty}$, such that $L_1 \cdot \alpha \subset L_2$.

Definition 2.2.11. [62, Section 10.1] Let B be a finite dimensional $\mathbb{Q}$-algebra. An *order* $\mathcal{O} \subset B$ is a ($\mathbb{Z}$-)lattice that is also a subring of B.

Example 2.2.12. Consider the imaginary quadratic number field $A = \mathbb{Q}\left[\sqrt{-1}\right]$. The ring of integers $\mathcal{O} = \mathbb{Z}\left[\sqrt{-1}\right]$ is an order generated by $\{1, i\}$.
[3]If we consider the quaternion algebra $A = B_{11,\infty}$, an order is $\mathcal{O} = \mathbb{Z}\left[\frac{1+j}{2}, \frac{i+k}{2}, j, k\right]$. In fact, this order is *maximal*, that is, there is no other order that contains $\mathcal{O}$.

Definition 2.2.13. [14, Section 2.4.5] An *integral ideal* I is a sublattice of an order $\mathcal{O}$ such that $\mathcal{O}I \subseteq I$ or $I\mathcal{O} \subseteq I$. The *left order* of I is $\mathcal{O}_L(I) = \{\alpha \in B : \alpha I \subset I\}$ and the *right order* is $\mathcal{O}_R(I) = \{\alpha \in B : I\alpha \subset I\}$. We say I is a left ideal of $\mathcal{O}_L(I)$, and I is a right ideal of $\mathcal{O}_R(I)$.

Definition 2.2.14. [61, Definition 16.2.9.] Let $\mathcal{O} \subseteq B$ be an R-order. A *left fractional $\mathcal{O}$-ideal* is a lattice $I \subseteq B$ such that $\mathcal{O} \subseteq \mathcal{O}_L(I)$; similarly on the right.

Definition 2.2.15. [14, Section 2.4.5] The norm of an integral ideal I, denoted $n(I)$, is the greatest common divisor of the norm of its elements.

[3]This example, and related examples, were obtained with the help of the SageMath documentation on quaternion algebras.

Proposition 2.2.16. [14, Section 2.4.5] The norm of an integral ideal I is always an integer, and is equal to

$$n(I) = \sqrt{[\mathcal{O}_L(I) : I]} = \sqrt{[\mathcal{O}_R(I) : I]}.$$

Example 2.2.17. Consider the order $\mathcal{O} = \mathbb{Z}\left[\frac{1+j}{2}, \frac{i+k}{2}, j, k\right]$ from Example 2.2.12 and consider the sublattice

$$I = \mathbb{Z}[2+2j+140,\ 2i+4j+150k,\ 8j+104k,\ 152k] \subseteq \mathcal{O}.$$

The sublattice I can be shown to be an integral ideal. We have that the left and right orders of I are,

$$\mathcal{O}_L(I) = \mathbb{Z}\left[\frac{1+j+70k}{2},\ \frac{i+2j+75k}{4},\ j+32k,\ 38k\right]$$

and

$$\mathcal{O}_R(I) = \mathbb{Z}\left[\frac{1+j+32k}{2},\ \frac{i+11k}{2},\ j+13k,\ 19k\right].$$

Furthermore, the norm of the ideal is,

$$n(I) = 32.$$

Definition 2.2.18. [14, Section 2.4.5] Two orders $\mathcal{O}_1$ and $\mathcal{O}_2$ are equivalent if there exists $\beta \in B_{p,\infty}$ such that $\beta\mathcal{O}_1 = \mathcal{O}_2\beta$. Two left integral $\mathcal{O}$-ideals I and J are equivalent if there exists $\beta \in B_{p,\infty}$ such that $I = J\beta$.

Proposition 2.2.19. [14, Section 2.4.5.2] The map χ_I from $I\backslash\{0\}$ to the set of ideals J

equivalent to I defined by

$$\chi_I(\alpha) = I\frac{\overline{\alpha}}{n(I)}$$

is a surjection.

2.3 Elliptic Curves

Notation 2.3.1.

- $\mathbb{F}_q$ denotes a finite field of order q.
- $\overline{K}$ denotes an algebraic closure of the field K.
- $K[X,Y,Z]$ denotes the polynomial ring in 3 variables, X,Y, and Z.
- K^* denotes the set of units in the field (or ring) K.

Definition 2.3.2. [18, Definition 1] A *projective space* of dimension n over a field K denoted $\mathbb{P}^n$ or $\mathbb{P}^n(\overline{K})$ is the set of equivalence classes of $(n+1)$-tuples

$$(x_0, x_1, \ldots, x_n) \in \overline{K}^{n+1}$$

such that $(x_0, x_1, \ldots, x_n) \neq (0, \ldots, 0)$ and we have the equivalence relation

$$(x_0, x_1, \ldots, x_n) \sim (y_0, y_1, \ldots, y_n)$$

if and only if there exists $\lambda \in \overline{K}^*$ such that $(x_0, x_1, \ldots, x_n) = (\lambda y_0, \lambda y_1, \ldots, \lambda y_n)$.

Definition 2.3.3. [18, Definition 2] Let K be a field with characteristic not equal to 2 or 3. An *elliptic curve* E defined over K is the set of solutions (X,Y,Z) in $\mathbb{P}^2(\overline{K})$ of the equation

$$Y^2Z = X^3 + aXZ^2 + bZ^3, \tag{2.1}$$

for some choice of $a,b \in K$ such that $4a^3 + 27b^2 \neq 0$.

When $Z = 0$, the only solution to the equation is the point (equivalence class) $O = (0,1,0)$. The point O is the *point at infinity*.

If we let $x = X/Z$ and $y = Y/Z$, we can re-write Equation 2.1 as

$$y^2 = x^3 + ax + b. \tag{2.2}$$

Equation 2.1 is the Weierstrass form and Equation 2.2 is its affine form. When defining an elliptic curve in affine form, we must include the point at infinity O as a solution such that we can define the following group structure on the elliptic curve.

Notation 2.3.4. E/K denotes an elliptic curve E defined over a field K. That is, the coefficients of Equation 2.1 or Equation 2.2 are in K.

Notation 2.3.5. $E(K)$ denotes the subset of solutions to Equation 2.1 where $(X,Y,Z) \in \mathbb{P}^2$; it is the set of *K-rational points* of E.

Definition 2.3.6. [18, Definition 3] Let E be an elliptic curve defined by the equation $y^2 = x^3 + ax + b$. Let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be two points on the elliptic curve which are not the point at infinity. We define the following composition law:

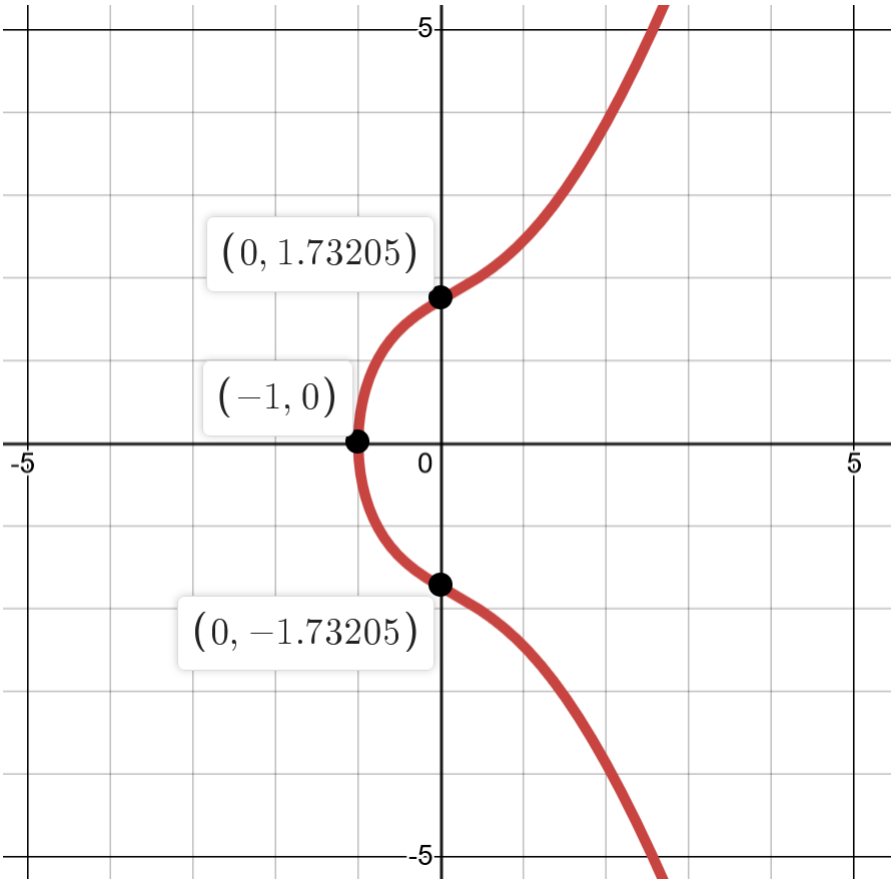

Figure 2.3: The graph of the elliptic curve $y^2 = x^3 + 2x + 3$ over $K = \mathbb{R}$. The graph was made using Desmos.

- $P + O = O + P = P$;
- if $x_P = x_Q$ and $y_P = -y_Q$, then $P + Q = O$;
- otherwise, $P + Q = R = (x_R, y_R)$ with

$$x_R = \lambda^2 - x_P - x_Q,$$

$$y_R = -\lambda x_R - y_P + \lambda x_P,$$

where

$$\lambda = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} & \text{if } P \neq Q, \\ \frac{3x_P^2 + a}{2y_P} & \text{if } P = Q. \end{cases}$$

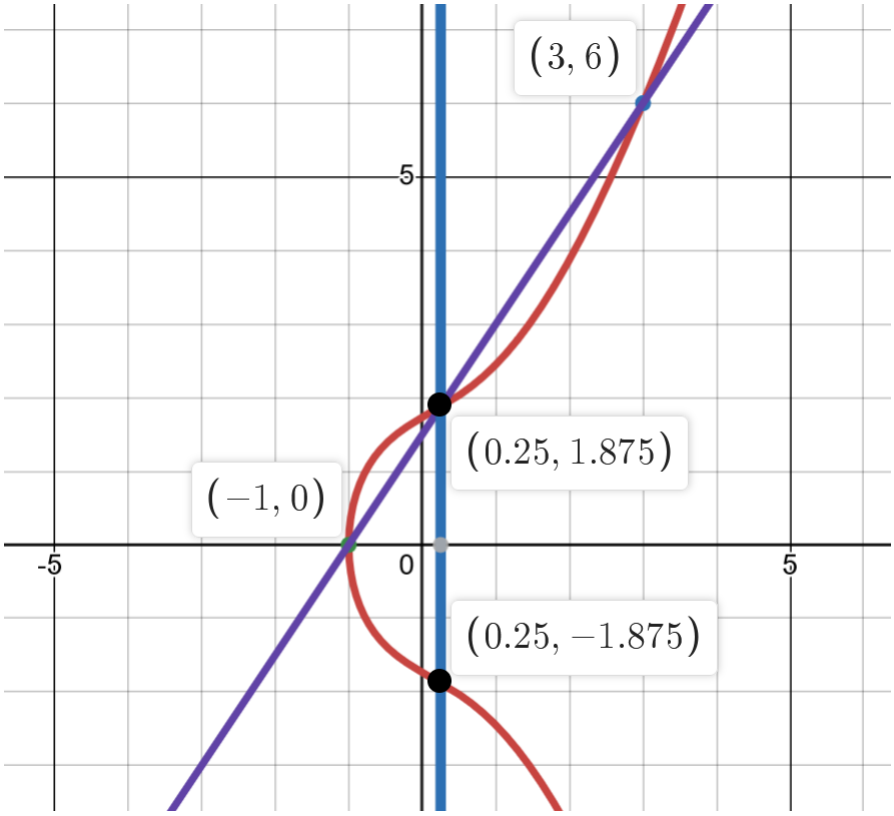

Figure 2.4: Addition of the points $(1,0)$ and $(3,6)$ on the elliptic curve $y^2 = x^3 + 2x + 3$ over $\mathbb{R}$. The result of the addition is $(\frac{1}{4}, \frac{15}{8})$

.

Proposition 2.3.7. [56, Proposition III.2.2] The above composition law defines a group structure on elliptic curves. Furthermore, the group is abelian.

Definition 2.3.8. A Montgomery curve curve over a field K is the set of solutions $(x,y) \in K^2$

to

$$M_{A,B} : By^2 = x^3 + Ax^2 + x \quad A,B \in K, \quad B(A^2-4) \neq 0,$$

where the point at infinity is included. In projective coordinates, the above equation becomes,

$$BY^2Z = X^3 + AX^2Z + XZ^2.$$

Proposition 2.3.9. A Montgomery curve is an elliptic curve.

Proof. Make the substitution $u = \frac{x}{B}$ and $v = \frac{y}{B}$ to obtain

$$v^2 = u^3 + \frac{A}{B}u^2 + \frac{1}{B^2}u.$$

Then make the substitution $u = t - \frac{A}{3B}$ to obtain,

$$v^2 = t^3 + \left(\frac{3-A^2}{3B^2}\right)t + \left(\frac{2A^3-9A}{27B^3}\right),$$

which is the equation of an elliptic curve in affine form. □

Remark 2.3.10. All Montgomery curves are elliptic curves, but not every elliptic curve can be converted into Montgomery form.

Definition 2.3.11. [56, Page 3] Let E/K be an elliptic curve defined over a field K by Equation 2.1. The homogeneous coordinate ring of E, denoted $K[E]$ is defined by

$$K[E] = \frac{K[X,Y,Z]}{\langle Y^2Z - X^3 - aXZ^2 - bZ^3 \rangle}.$$

The function field of E, denoted $K(E)$, is the field of fractions of $K[E]$.

Definition 2.3.12. [56, Pages $11-12$] Let E and E' be elliptic curves defined by 2.1. A *morphism* from E to E' is a map of the form

$$\phi = [f_0, f_1, f_2] : E \to E',$$

where $f_0, f_1, f_2 \in \overline{K}(E)$ and for any $P \in E$

$$\phi(P) = [f_0(P), f_1(P), f_2(P)] \in E'.$$

If there exists some $\lambda \in \overline{K}^*$ such that $\lambda f_0, \lambda f_1, \lambda f_2 \in K(E)$, then ϕ is *defined over* K.

Definition 2.3.13. [56, Page 66] An *isogeny* is a morphism $\phi : E \to E'$ between elliptic curves E and E' that sends O to O. If $\phi(E) = \{O\}$, then ϕ is the *zero isogeny*. Two elliptic curves E and E' are *isogenous* if there exists a non-zero isogeny $E \to E'$.

Theorem 2.3.14. [56, Theorem III.4.8 and Page 66] A non-zero isogeny is a surjective group homomorphism.

Definition 2.3.15. [56, Page 20] Let E/K and E'/K be elliptic curves defined over a field K, and $\phi : E \to E'$ be a non-zero morphism defined over K. We define the map

$$\phi^* : K(E') \to K(E), \text{ where } \phi^*(f) = f \circ \phi.$$

Theorem 2.3.16. [56, Theorem II.2.4] $K(E)$ is a finite extension of $\phi^*(K(E'))$.

Definition 2.3.17. [56, Page 21] Let E/K and E'/K be elliptic curves defined over a field K, and $\phi : E \to E'$ be a morphism defined over K. If ϕ is constant we define the *degree* of

ϕ to be 0. Otherwise, ϕ has degree equal to the extension degree

$$\deg(\phi) = [K(E) : \phi^*(K(E'))].$$

ϕ is *separable, inseparable,* or *purely inseparable* if the field extension $K(E)/\phi^*(K(E'))$ has the corresponding property.

Proposition 2.3.18. [56, Theorem III.4.10.c and Proposition III.4.12] A separable isogeny is uniquely defined (up to isomorphism) by its kernel. Furthermore, the degree of a separable isogeny is equal to the cardinality of its kernel,

$$\deg(\phi) = \#\ker(\phi).$$

Notation 2.3.19. Let $\phi : E \to E'$ be a separable isogeny with kernel G. We denote the curve E' with E/G.

Notation 2.3.20. Given a point P of an elliptic curve E, we write $x(P)$ for the x-coordinate of P; and similarly for $y(P)$.

Theorem 2.3.21. [18, Proposition 38] Let $E : y^2 = x^3 + ax + b$ be an elliptic curve defined over a field K, and let $G \subset E(\overline{K})$ be a finite subgroup. The unique (up to isomorphism) separable isogeny $\phi : E \to E/G$, of kernel G can be written as

$$\phi(P) = \left(x(P) + \sum_{Q \in G \setminus \{O\}} \left(x(P+Q) - x(Q) \right), \quad y(P) + \sum_{Q \in G \setminus \{O\}} \left(y(P+Q) - y(Q) \right) \right);$$

and the curve E/G has equation $y^2 = x^3 + a'x + b'$, where

$$a' = a - 5 \sum_{Q \in G \setminus \{O\}} (3x(Q)^2 + a)$$
$$b' = b - 7 \sum_{Q \in G \setminus \{O\}} (5x(Q)^3 + 3ax(Q) + b).$$

Definition 2.3.22. An isogeny is *cyclic* if its kernel is cyclic.

Example 2.3.23. [56, Example III.4.1] For each $m \in \mathbb{Z}$ we have the *multiplication-by-m isogeny* $[m] : E \to E$ defined by

$$[m](P) = \underbrace{P + P + \cdots + P}_{m \text{ terms}}$$

when $m > 0$. When $m < 0$ we define $[m](P) = [-m](-P)$ and $[0](P) = O$.

Example 2.3.24. In most of isogeny-based cryptography, we are interested only in separable isogenies. This is because we can describe the isognies nicely by their kernels. Consider the elliptic curve $E : y^2 = x^3 + x + 1$ defined over the finite field $\mathbb{F}_7$, and the subgroup $G = \{(2,0), O\}$. One can verify that the curve E/G is defined by $y^2 = x^3 + 2x + 6$ and the isogeny $\phi : E \to E/G$ is defined by,

$$\phi(x,y) = \left(\frac{x^2 + 2x + 2}{x - 2}, \frac{x^2 y - 4xy + 2y}{x^2 - 4x + 4} \right)$$

and $\deg(\phi) = 2$.

Definition 2.3.25. [56, Page 69] Let E be an elliptic curve and $m \geq 1$. We define the

m-torsion subgroup of E, denoted $E[m]$ as the set of points of E with order m

$$E[m] = \{P \in E : [m]P = O\}.$$

Proposition 2.3.26. [56, Corollary III.6.4] Let E be an elliptic curve defined over a field of characteristic $p > 0$, and let $m \in \mathbb{Z}$ with $m \neq 0$. Then

- $\deg[m] = m^2$, and
- if $p \nmid m$, then,

$$E[m] = \frac{\mathbb{Z}}{m\mathbb{Z}} \times \frac{\mathbb{Z}}{m\mathbb{Z}}.$$

Definition 2.3.27. [18, Proposition 6] Let E be an elliptic curve defined by the equation $y^2 = x^3 + ax + b$. The quantity

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

is the *j-invariant* of E.

Proposition 2.3.28. [56, Proposition III.1.4] Two elliptic curves are *isomorphic* over $\overline{K}$ if and only if they both have the same j-invariant.

Theorem 2.3.29. [56, Theorem III.6.1] Given an non-constant isogeny $\phi : E_1 \to E_2$ of degree m, there exists a unique isogeny $\widehat{\phi} : E_2 \to E_1$ of degree m, the *dual isogeny*, satisfying $\phi \circ \widehat{\phi} = [m]$ and $\widehat{\phi} \circ \phi = [m]$.

Proposition 2.3.30. [37] If $\phi : E \to E'$ is an isogeny of elliptic curves over a field K with $\mathrm{char}(K) = p > 0$, then ϕ and $\widehat{\phi}$ are separable if and only if $p \nmid \deg(\phi)$.

Proposition 2.3.31. Let K be a field of characteristic $p > 3$ and let $\phi : E \to E'$ be a degree 2, separable, cyclic, isogeny defined over K with kernel $\ker(\phi) = \langle \alpha \rangle$. Let $P \in E$ be a point of order 2 with $P \notin \ker(\phi)$. Then $\ker(\widehat{\phi}) = \langle \phi(P) \rangle$.

Proof.

1. We show that $\phi(P)$ has order 2 in E'.
2. We show $\widehat{\phi}(\phi(P)) = O$.
3. We show that $\ker(\widehat{\phi}) = \langle \phi(P) \rangle$

Step 1 follows from the fact that isogenies are homomorphism (Theorem 2.3.14): $[2]\phi(P) = \phi([2]P) = \phi(O) = O$. Step 2 follows from the fact that $\widehat{\phi} \circ \phi = [2]$ (Theorem 2.3.29), and $[2]P = O$ (by assumption). Step 3 follows because $\widehat{\phi}$ is separable of degree 2 (Theorem 2.3.29 and Proposition 2.3.30), it has a kernel generated by a point of order 2 (Proposition 2.3.18). That point is $\phi(P)$ and hence $\ker(\phi) = \langle \phi(P) \rangle$. □

Proposition 2.3.32. Let K be a field of characteristic $p > 3$ and let $\phi : E \to E'$ be a separable isogeny defined over K, of degree 2. Then there are 3 points of order 2, $P_i \in E$, where $i = 1, 2, 3$, which can generate $\ker(\phi)$. Suppose $\ker(\phi) = \langle P_1 \rangle$, then $\phi(P_2) = \phi(P_3)$.

Proof. By Proposition 2.3.26, we have that $E[2] = \frac{\mathbb{Z}}{2\mathbb{Z}} \times \frac{\mathbb{Z}}{2\mathbb{Z}}$, thus there are 3 points of order 2. In $\frac{\mathbb{Z}}{2\mathbb{Z}} \times \frac{\mathbb{Z}}{2\mathbb{Z}}$, the three points are $P_1 = (0,1)$, $P_2 = (1,0)$, and $P_3(1,1)$. We have that $P_3 = P_1 + P_2$, $P_2 = P_3 - P_1$, $P_1 = P_3 - P_2$. From Theorem 2.3.14, we have that ϕ is a homomorphism. Thus, for example if $\ker(\phi) = \langle P_1 \rangle$,

$$\phi(P_3) - \phi(P_2) = \phi(P_3 - P_2) = \phi(P_1) = O.$$

Thus, $\phi(P_3) = \phi(P_2)$ as desired. The other cases follow similarly. □

Theorem 2.3.33. [56, Exercise 5.4] or [58] Let $E/\mathbb{F}_q$ and $E'/\mathbb{F}_q$ be elliptic curves defined over $\mathbb{F}_q$. Then E and E' are isogenous over $\mathbb{F}_q$ if and only if $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$.

Definition 2.3.34. Let E be an elliptic curve. An *endomorphism* is an isogeny with equal domain and codomain $\phi : E \to E$. The endomorphism ring of E is denoted $\mathrm{End}(E)$ and is the ring of all endomorphisms on E.

Remark 2.3.35. We denote the full ring of endomorphism $\mathrm{End}(E)$, that is, all endomorphisms defined over an algebraic closure of the underlying field of definition of E. We denote the subring of endomorphisms defined over $\mathbb{F}_p$ as $\mathrm{End}_{\mathbb{F}_p}(E)$.

Theorem 2.3.36. [18, Theorem 29] Let E be an elliptic curve defined over a field K of characteristic p. The ring $\mathrm{End}(E)$ is isomorphic to one of the following:

- $\mathbb{Z}$, can only occur if $p = 0$.
- An order $\mathcal{O}$ in a quadratic imaginary field (a number field of the form $\mathbb{Q}[\sqrt{-D}]$, $D > 0$); in this case E has *complex multiplication* by $\mathcal{O}$.
- Only if $p > 0$, a maximal order in the quaternion algebra $B_{p,\infty}$; in this case E is *supersingular*.

For $p > 0$, if E is not supersingular, it is ordinary, and it necessarily has complex multiplication.

Theorem 2.3.37. [29, Section 2.3][23] There is a correspondence of the endomorphism ring of a supersingular elliptic curve and maximal orders in $B_{p,\infty}$. In particular, isogenies of

supersingular elliptic curves are equivalent to left integral ideals in a maximal order. We summarize the result in Table 2.1.

Supersingular j-invariants over $\mathbb{F}_{p^2}$ $j(E)$ (up to Galois conjugacy)	Maximal orders $\mathcal{O} \cong \mathrm{End}(E)$ in $\mathcal{B}_{p,\infty}$ (up to isomorphism)
(E_1, ϕ) with $\phi : E \to E_1$	I_ϕ integral left $\mathcal{O}$-ideal and right $\mathcal{O}_1$-ideal
$\theta \in \mathrm{End}(E_0)$	Principal ideal $\mathcal{O}\theta$
$\deg(\phi)$	$n(I_\phi)$
$\widehat{\phi}$	$\overline{I_\phi}$
$\phi : E \to E_1$, $\psi : E \to E_1$	Equivalent Ideals $I_\phi \sim I_\psi$
$\tau \circ \rho : E \to E_1 \to E_2$	$I_{\tau\circ\rho} = I_\rho \cdot I_\tau$

Table 2.1: [29, Table 1] Summary of Deuring correspondence.

Proposition 2.3.38. [21, Theorem 2.1] Let $p > 3$ and let E be a supersingular elliptic curve defined over $\mathbb{F}_p$. Then $\mathrm{End}_{\mathbb{F}_p}(E)$ is an order in a quadratic imaginary field.

Theorem 2.3.39. [56, Theorem V.3.1] In characteristic $p > 0$, the j-invariant of a supersingular curve is in $\mathbb{F}_{p^2}$.

Theorem 2.3.40. [56, Theorem V.4.1] For $p \geq 5$ there are

$$\left\lfloor \frac{p}{12} \right\rfloor + \begin{cases} 0 & \text{if } p \equiv 1 \pmod{12}, \\ 1 & \text{if } p \equiv 5 \pmod{12}, \\ 1 & \text{if } p \equiv 7 \pmod{12}, \\ 2 & \text{if } p \equiv 11 \pmod{12}, \end{cases} \tag{2.3}$$

supersingular elliptic curves up to $\overline{\mathbb{F}_q}$-isomorphism. $\lfloor x \rfloor$ denotes the floor function.

Definition 2.3.41. [56, Example III.4.6] Let $E(\mathbb{F}_q)$ be an elliptic curve. The map $\pi : E(\overline{\mathbb{F}_q}) \to E(\overline{\mathbb{F}_q})$ defined by

$$\pi : (x,y) \to (x^q, y^q)$$

is the *Frobenius endomorphism* of $E(\overline{\mathbb{F}_q})$.

Proposition 2.3.42. [56, Theorem V.2.3.1] Let $E/\mathbb{F}_q$ be an elliptic curve. The Frobenius endomorphism satisfies the equation

$$\pi^2 - T\pi + q = 0 \tag{2.4}$$

in $\mathrm{End}(E)$, where $T = q + 1 - \#E(\mathbb{F}_q)$. T is the *trace of Frobenius*.

Definition 2.3.43. The *Frobenius discriminant* is the discriminant of Equation 2.4,

$$D_\pi = T^2 - 4q.$$

Proposition 2.3.44. Isogenous elliptic curves have the same Frobenius discriminant.

Proof. This follows from the definitions of the trace of Frobenius (2.3.42) and the Frobenius discriminant (2.3.43), as well as isogenous elliptic curves have an equal number of points (2.3.33). □

Definition 2.3.45. [29, Section 4.1] Suppose we have three elliptic curves E_0, E_1, E_2, and two separable isogenies $\phi_1 : E_0 \to E_1$ and $\phi_2 : E_0 \to E_2$ of coprime degrees N_1 and N_2. We define the *pushforward isogenies* $[\phi_1]_*\phi_2$ and $[\phi_2]_*\phi_1$ as the isogenies

$$[\phi_1]_*\phi_2 : E_1 \to E_3, \text{ with } \ker([\phi_1]_*\phi_2) = \phi_2(\ker(\phi_1)),$$

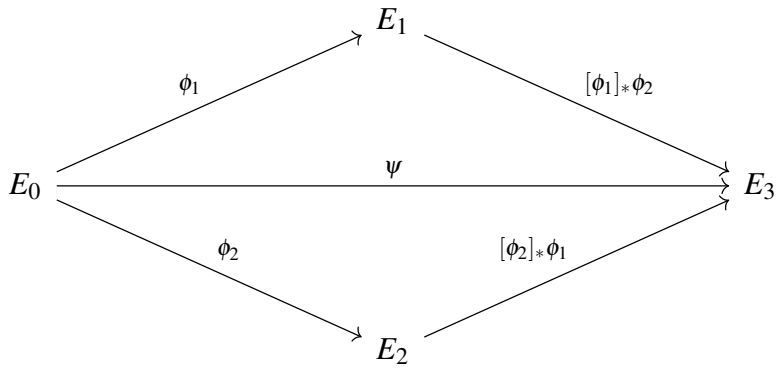

Figure 2.5: A commutative isogeny diagram showing the pushforward isogeny.

$$[\phi_2]_*\phi_1 : E_2 \to E_3', \text{ with } \ker([\phi_2]_*\phi_1) = \phi_1(\ker(\phi_2)).$$

Furthurmore, $\deg([\phi_1]_*\phi_2) = N_2$ and $\deg([\phi_2]_*\phi_1) = N_1$. When using the Deuring correspondence (Theorem 2.3.37), we write $[I]_*J$ for the ideal $I_{[\phi_J]_*\phi_I}$ corresponding to the pushforward isogeny $[\phi_J]_*\phi_I$.

Proposition 2.3.46. [29, Section 4.1] The curves E_3 and E_3', in Definition 2.3.45, are equal.

Proposition 2.3.47. [29, Lemma 3] In the setting of Definition 2.3.45, write I_1 (resp. I_2) for the corresponding ideal to the isogeny ϕ_1 (resp. ϕ_2). Similarly, write $J_1 = [I_2]_*I_1$, $J_2 = [I_1]_*I_2$, and $K = I_\psi$, where K is the ideal corresponding to the isogeny ψ.
We have that the ideals J_1, J_2 and K are well-defined, and

1. $K = I_1 \cap I_2$
2. $J_2 = I_1^{-1}(I_1 \cap I_2)$ and $J_1 = I_2^{-1}(I_1 \cap I_2)$.

2.4 Class Group Action

Definition 2.4.1. [18, Definition 50] Let $\mathcal{O}$ be an order in a number field K. A *fractional ideal* of $\mathcal{O}$ is a non-zero subgroup $I \subset K$ such that

- $xI \subset I$ for all $x \in \mathcal{O}$, and
- there exists a non-zero $x \in \mathcal{O}$ such that $xI \subset \mathcal{O}$.

A fractional ideal is *principal* if it is of the form $x\mathcal{O}$ for some $x \in K$.

Definition 2.4.2. [18, Page 29] A fractional ideal I is *invertible* if there is another ideal J such that $IJ = \mathcal{O}$.

Proposition 2.4.3. [18, Page 29] Invertible ideals form an abelian group, written $\mathcal{I}(\mathcal{O})$, under the operation

$$IJ = \left\{ \sum_{i=1}^{n} x_i y_i : n \in \mathbb{N}, x_i \in I, y_i \in J \right\},$$

where $\mathcal{O}$ is the identity element, and the set of principals ideals form a subgroup of it, written $\mathcal{P}(\mathcal{O})$.

Proposition 2.4.4. [18, Proposition 51] The ideal class group of $\mathcal{O}$ is the quotient

$$Cl(\mathcal{O}) = \mathcal{I}(\mathcal{O})/\mathcal{P}(\mathcal{O}).$$

It is a finite abelian group. Its order, denoted by $h(\mathcal{O})$, is the *class number* of $\mathcal{O}$.

Definition 2.4.5. [18, Definition 52] Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$. Let $\mathcal{O}$ be the endomorphism ring of E, and let $\mathfrak{a} \subset \mathcal{O}$ be an integral invertible ideal of norm

coprime to q. We define the $\mathfrak{a}$-torsion subgroup of E as

$$E[\mathfrak{a}] = \{P \in E | \alpha(P) = 0 \text{ for all } \alpha \in \mathfrak{a}\}.$$

We write $E_\mathfrak{a} = E/E[\mathfrak{a}]$, and define the isogeny $\phi_\mathfrak{a} : E \to E_\mathfrak{a}$ as the unique (up to isomorphism) isogeny with kernel $E[\mathfrak{a}]$.

Proposition 2.4.6. [18, Page 29] $\text{End}(E) \cong \text{End}(E_\mathfrak{a}) \cong \mathcal{O}$.

Theorem 2.4.7. [18, Theorem 53] Let $\mathbb{F}_q$ be a finite field, and let $\mathcal{O} \subset \mathbb{Q}[\sqrt{-D}]$ be an order in a quadratic imaginary field. Denote by $\text{Ell}_q(\mathcal{O})$ the set of elliptic curves defined over $\mathbb{F}_q$ with complex multiplication by $\mathcal{O}$.

Assume that $\text{Ell}_q(\mathcal{O})$ is non-empty, then the class group $Cl(\mathcal{O})$ acts freely and transitively on it; i.e. there is a map

$$Cl(\mathcal{O}) \times \text{Ell}_q(\mathcal{O}) \to \text{Ell}_q(\mathcal{O}),$$

$$(\mathfrak{a}, E) \to \mathfrak{a} \cdot E = E_\mathfrak{a},$$

such that $\mathfrak{a} \cdot (\mathfrak{b} \cdot E) = (\mathfrak{a}\mathfrak{b}) \cdot E$ for all $\mathfrak{a}, \mathfrak{b} \in Cl(\mathcal{O})$ and $E \in \text{Ell}_q(\mathcal{O})$, and such that for any $E, E' \in \text{Ell}_q(\mathcal{O})$ there is a unique $\mathfrak{a} \in Cl(\mathcal{O})$ such that $E' = \mathfrak{a} \cdot E$.

2.5 Graph Theory

Definition 2.5.1. [47, Page 1] A *graph* is a pair $G = (V, E)$ where V is a set with *vertices* and E is a set of pairs $\{v_1, v_2\}$ of vertices, the *edges*. Two vertices v_1 and v_2 are *neighbours*

(or *adjacent*) if there exists an edge $\{v_1, v_2\}$. An edge $\{v, v\}$ is a *loop*. A *simple graph* is a graph with no loops or multiple equal edges. If a graph is not simple it is a *multigraph*.

We consider only simple *undirected graphs* in this thesis, meaning the edges $\{v_1, v_2\}$ and $\{v_2, v_1\}$ are one and the same. This is because we look at *isogeny graphs*, which have vertices as elliptic curves and edges as isogenies. By (2.3.29), every isogeny $\phi : E_1 \to E_2$ has a dual $\widehat{\phi} : E_2 \to E_1$.

Definition 2.5.2. [47, Page 2] The *degree* of a vertex is the number of edges connected to the vertex. A *k-regular* graph is a graph in which each vertex has degree k.

Definition 2.5.3. [47, Page 2] A *path of length n* from vertex v_0 to vertex v_n is a sequence of distinct vertices $\{v_0, v_1, \ldots, v_{n-1}, v_n\}$ such that there exists an edge $\{v_{i-1}, v_i\}$ for each $i = 1, 2, \ldots, n$. A graph is *connected* if for any two vertices x, y there exists a path from x to y.

Remark 2.5.4. [47, Page 2] The words *walk* and *path* differ depending on the author. A path is as defined here, with no repeated vertices allowed. A walk is similarly defined, except that repeated vertices are allowed.

Definition 2.5.5. [47, Page 1] Given a graph G with n vertices, we associate an *adjacency matrix A* which is an $n \times n$ matrix. $[A]_{(i,j)}$ equals the number of edges connecting vertex i to vertex j.

It follows that an undirected graph has a symmetric adjacency matrix.

Theorem 2.5.6. [47, Theorems 1,3 and Corollary 2] If G is a k-regular graph, then all the eigenvalues λ of its adjacency matrix satisfy $|\lambda| \leq k$. Furthermore, $\lambda = k$ is an eigenvalue with multiplicity equal to the number of connected components of G.

Thus, if G is a connected k-regular graph, then there is one connected component and the eigenvalue $\lambda = k$ has multiplicity one. We can arrange the eigenvalues as:

$$k = \lambda_0(G) > \lambda_1(G) \geq \cdots \geq \lambda_{n-1}(G) \geq -k.$$

Definition 2.5.7. [47, Page 3] The eigenvalues $\lambda_i \neq \pm k$ are *non-trivial eigenvalues* and we denote $\lambda(G)$ the maximum of the absolute values of all the non-trivial eigenvalues.

Definition 2.5.8. [47, Page 3] A *Ramanujan multigraph* is a k-regular graph G satisfying

$$\lambda(G) \leq 2\sqrt{k-1}.$$

Definition 2.5.9. [47, Page 4] The *distance* $d(x,y)$ between two vertices $x, y \in V$ is the minimal length amongst all the paths from x to y.

Definition 2.5.10. [47, Page 15] For any subset X of a graph G, we define the *boundary* ∂X of X by,

$$\partial X = \{y \in G : d(y,X) = 1\}.$$

That is, the boundary of X consists of vertices which is a neighbour to some vertex in X.

Definition 2.5.11. [47, Page 15] Let c be a positive real number. A k-regular graph G with n vertices is *c-expander* if

$$\frac{|\partial X|}{|X|} < c$$

for all $X \subset G$ with $|X| \leq |G|/2$.

Proposition 2.5.12. [27, Proposition 2.1] Let G be a k-regular graph. Suppose $\lambda(G) \leq c$

for some $c < k$. Let S be any subset of the vertices of G, and x be any vertex in G. Then a random walk of length at least $\frac{\log 2|G|/|S|^{1/2}}{\log k/c}$ starting from x lands in S with probability at least $\frac{|S|}{2|G|}$.

Chapter 3

Isogeny-Based Cryptography

We begin this chapter by giving a brief history of isogeny-based cryptography. We then move on to explore at a high level the protocols (1) of Couveignes [16], (2) of Rostvostsev and Stolbunov [53], (3) CSIDH [11], and (4) SIDH [27]. We also take a look at (5) the CGL hash function [13].

The protocols (1), (2), and (3) are all similar in nature and should be read in sequential order. The protocol (4) has ideas from (1), (2), (3), but the security problem underlying the protocol is fundamentally different. The construction in protocol (4) more closely resembles the construction from (5).

The goal of this chapter is to have a preview of the machinery that is used in the specification of SQISign, and we emphasize that this is by no means a complete coverage. There are many other isogeny-based cryptographic protocols/algorithms that we do not cover, including but not limited to: SeaSign digital signature [19], an elliptic curve trapdoor system [59], CSI-FiSh digital signature [8], two verifiable delay functions [28]. The ideas

that we do cover are selected due to their historical interest, importance to the NIST post-quantum standardization efforts, and applicability to the main topic of the thesis, the SQISign digital signature.

For an overview on the security/cryptanalysis of the systems we discuss, see Chapter 4.

3.1 History

The oldest known writings of a cryptographic protocol suggesting the use of isogenies date back to Jean-Marc Couveignes in 1997 with the paper titled Hard Homogeneous Spaces [16]. Couveignes submitted his paper to the Crypto97 conference but was rejected and he left the paper unpublished until August 2006.

Independently, Rostovtsev and Stolbunov rediscovered Couveignes' idea, publishing it around the start of 2006 [53], hinting at the applicability of the use of isogenies for post-quantum cryptography. In 2010, Stolbunov provided an implementation of his earlier paper [57]. In 2018, Childs, Jao, and Soukharev developed a quantum subexponential time attack on the Stolbunov system [15]. The foundations of this attack can be applied to any system built on the ideas of Couveignes. Although this attack was not devastating, the efficiency of the protocol was unacceptable, and the ideas were not pursued in much depth until around 2018 (see CSIDH below).

Around the same time as Rostovtsev and Stolbunov published their paper, Charles, Goren, and Lauter proposed a hash function [13]. The collision-resistance of the hash function is derived from the supersingular isogeny problem (Problem 6). The pseudo-

random output property comes from the rapid mixing properties of random walks on expander graphs (Theorem 3.5.1). The Charles, Goren and Lauter (CGL) hash function has not been successfully attacked as of the writing of this thesis.

In 2011, Jao, De Feo, and Plût proposed SIDH (supersingular isogeny Diffie-Hellman) as a key-exchange protocol conjecturally resistant to quantum computers [27]. In 2017, SIDH was submitted to NIST under the name SIKE, and passed three rounds of examination, until in 2022, Castryk and Decru published an attack breaking SIDH on a classical computer [10].

In 2018, Castryck, Lange, Martindale, Panny, and Renes published CSIDH (pronounced "sea side", and stands for Commutative-SIDH), the reincarnation of the ideas of Couveignes, Rostovtsev, and Stolbunov, which attempts to fix the performance issues [11].

3.2 Hard Homogeneous Spaces

The goal of the paper by Couveignes [16] is to create a framework in which cryptographic protocols can be created based on the difficulty of certain problems, such as the discrete logarithm problem (Problem 1).

Definition 3.2.1. Let G be a finite commutative group. A *homogeneous space* H for G is a finite set with equal cardinality, $\#H = \#G$, which is acted on by G simply transitively.

Note that because the action is simply transitive, for any $h_1, h_2 \in H$ there is a unique $g \in G$ such that $g.h_1 = h_2$. We denote this g as $\delta(h_2, h_1)$.

Couveignes then defines several algorithmic problems a cryptographic protocol should

consider. We assume elements in G and H are represented by strings (not necessarily uniquely).

Problem 1 (Group Operations) Given strings g_1 and g_2 decide if they represent elements in G and if the strings represent the same element. Given $g_1, g_2 \in G$ compute $g_1^{-1}, g_1 g_2$, and decide if $g_1 = g_2$.

Problem 2 (Random Element) Find a random element in G with uniform probability.

Problem 3 (Membership) Given a string h_0 decide if h_0 represents an element in H.

Problem 4 (Equality) Given $h_1, h_2 \in H$ decide if $h_1 = h_2$.

Problem 5 (Action) Given $g \in G$ and $h \in H$ compute $g.h$.

Problem 6 (Vectorization) Given $h_1, h_2 \in H$ find $g \in G$ such that $g.h_1 = h_2$ (i.e. find $\delta(h_2, h_1)$).

Problem 7 (Parallelization) Given $h_1, h_2, h_3 \in H$ compute the unique h_4 such that $\delta(h_2, h_1) = \delta(h_4, h_3)$. (Note: $h_4 = \delta(h_2, h_1).h_3$)

Problem 8 (Parallel Testing) Given $h_1, h_2, h_3, h_4 \in H$ decide if $\delta(h_2, h_1) = \delta(h_4, h_3)$.

Definition 3.2.2. A *hard homogeneous space* (HHS) is a homogeneous space where Problems 1 to 5 are computationally easy, while Problems 6 and 7 are hard. If Problem 8 is hard then we call the homogeneous space a *very hard homogeneous space* (VHHS).

Couveignes then describes a key exchange scheme based on an arbitrary HHS, which can be seen as a generalization of the Diffie-Hellman scheme. As usual, suppose we have two parties, Alice and Bob, who wish to exchange a secret key.

Key Exchange:

1. Alice chooses a random element $h_0 \in H$ and a random element $g_1 \in G$ (easy by Problem 2). She computes $h_1 = g_1.h_0$ (easy by Problem 5) and sends (h_0, h_1) to Bob.

2. Bob chooses a random element $g_2 \in G$ and computes $h_2 = g_2.h_0$. He sends h_2 to Alice.

3. Alice computes $g_1.h_2$ and Bob computes $g_2.h_1$. We see that $g_2.h_1 = g_2.(g_1.h_0) = (g_2g_1).h_0$ and $g_1.h_2 = g_1.(g_2.h_0) = (g_1g_2).h_0$. Because the group G is assumed to be commutative, $g_1g_2 = g_2g_1$, thus Alice and Bob now have a shared secret.

An adversary listening to the communications between Alice and Bob would have access to (h_0, h_1, h_2). To find the shared secret he would need to find g_1 or g_2, i.e. solving $h_1 = g_1.h_0$ for g_1, or $h_2 = g_2.h_0$ for g_2 (hard by Problems 6 and 7).

Example 3.2.3. [11, Page 5] The original Diffie-Helman key exchange [24] on a cyclic group C can be viewed as an instance of a key exchange in a HHS where H is the set of generators of C and $G = (\mathbb{Z}/\#C)^*$ and the action is exponentiation.

Couveignes finalizes his paper by giving an example of a HHS not dependent on the discrete logarithm problem, but the problem of computing isogenies between elliptic curves. We cover this in the following subsection in the scheme proposed by Rostovtsev and Stolbunov.

3.3 Isogeny Stars

The paper by Rostovtsev and Stolbunov [53] proposes an encryption protocol with security based on the problem of computing isogenies between elliptic curves. They do not create the framework of a HHS as is done in the Couveignes paper, but their construction matches Couveignes' example of a new HHS. We outline the construction here.

Theorem 3.3.1. [53, Theorem 2] Let $E(\mathbb{F}_p)$ be an elliptic curve (Definition 2.3.3) with Frobenius discriminant (Definition 2.3.43) D_π, and $\left(\frac{D_\pi}{l}\right)$ be the Kronecker symbol. If $\left(\frac{D_\pi}{l}\right) = -1$, then there are no l-degree isogenies; if $\left(\frac{D_\pi}{l}\right) = 1$, then two l-degree isogenies exist; if $\left(\frac{D_\pi}{l}\right) = 0$, then 1 or $l+1$ l-degree isogenies exist.

Definition 3.3.2. In the notation of the above theorem, if a prime l satisfies $\left(\frac{D_\pi}{l}\right) = 1$, the prime is called an *Elkies prime*.

For the encryption protocol, we consider a finite subset $U \subseteq \mathrm{Ell}_q(\mathcal{O})$, where $\mathrm{Ell}_q(\mathcal{O})$ is the set of ordinary elliptic curves $E/\mathbb{F}_q$ with $\mathrm{End}(E) \cong \mathcal{O} \subseteq \mathbb{Q}[\sqrt{-D}]$. We further impose that all elliptic curves in U have an equal number of points, hence are all isogenous by Theorem 2.3.33. We choose some elliptic curve $E_0 \in U$, and calculate the corresponding Frobenius discriminant D_π (Definition 2.3.43). Note that all elliptic curves in U have equal Frobenius discriminant (Proposition 2.3.44).

Now if we consider an Elkies prime l, with $\left(\frac{D\pi}{l}\right) = 1$, then by Theorem 3.3.1, we have that two l-degree isogenies from E_0 exist. As all elliptic curves in U are isogenous, by Proposition 2.3.44, all elliptic curves have the same Frobenius discriminant. Hence l is an Elkies prime for all $E \in U$. Since U is finite, this implies that U forms an l-isogeny graph of branchless cycles.

It can be shown (see for example [36]) that when the cardinality of U is prime, the l-isogeny graph of U is a single cycle. This applies for any Elkies prime.

By considering another Elkies prime l' for E_0 we build another cycle of the vertices in U. Overlapping the l-isogeny and l'-isogeny cycles forms a so-called *isogeny star*; see Figure 3.1.

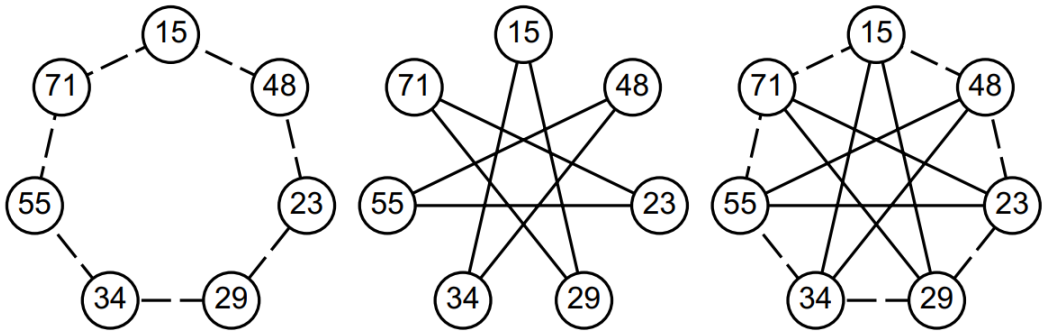

Figure 3.1: [53, Figure 1]. A set U with 7 elliptic curves defined over $\mathbb{F}_{83}$ with trace of Frobenius $T = 9$. The j-invariants are listed in the nodes. The first two graphs represent the 3-degree and 5-degree isogeny graphs respectively, while the last shows their overlap, i.e. an isogeny star.

The idea of the encryption protocol is to select a list of d Elkies primes l_i of E_0, where $1 \leq i \leq d$, and to take a walk through the isogeny star.

Remark 3.3.3. To implement a walk in the protocol, we must have a way to express which direction each step in the walk takes. For example, in Figure 3.1, if the walk starts at the vertex with j-invariant 15 and we wish to take a walk in the 3-degree cycle, we have two options. We can either go to 48 (clockwise/positive direction) or to 71 (counter-clockwise/negative direction). See [53, Section 8] for the details.

Remark 3.3.4. To build a secure cryptosystem, the size of the isogeny star, $\#U$, must be

very large. To compute the size of $\#U$, see [26, Page 164] or [20, Page 7]. Citing, "The size of the cycle is the order of $(\pi - \lambda, l)$ in $Cl(\mathcal{O})$, thus partitioning $\mathrm{Ell}_q(\mathcal{O})$ into cycles of equal size."

Setup. Two parties, Alice and Bob, agree on:

1. A finite field $\mathbb{F}_p$.
2. An isogeny star U and an initial elliptic curve $E_{init} \in U$.
3. $L = \{l_i\}, 1 \leq i \leq d$, the set of isogeny degrees being used, each being an Elkies prime of E_{init}.

Private Key. Alice chooses a random walk R_{priv} along the isogeny star. The walk R_{priv} is her private key, and corresponds to an isogeny $\phi_{priv} : E_{init} \to E_{pub}$. This isogeny, ϕ_{priv}, is the composition of the isogenies along her walk, R_{priv}.

Public Key. Alice's public key is the elliptic curve E_{pub}.

Encryption. Bob wishes to encrypt a cleartext message m and send it to Alice so she can decrypt it. He proceeds as follows:

1. He selects a random path in the isogeny star R_{enc}. This corresponds to an isogeny $\phi_{enc} : E_{init} \to E_{add}$.
2. He computes the elliptic curve $E_{enc} = R_{enc}(E_{pub})$ and computes the ciphertext $s = m \cdot j(E_{enc}) \pmod{p}$.
3. He communicates s and E_{add} to Alice.

Decryption. Alice wishes to decrypt the ciphertext s received from Bob into the cleartext m. She proceeds as follows:

1. She computes the elliptic curve $E_{enc} = R_{priv}(E_{add})$.

2. She computes the cleartext $m = \frac{s}{j(E_{enc})} \pmod{p}$

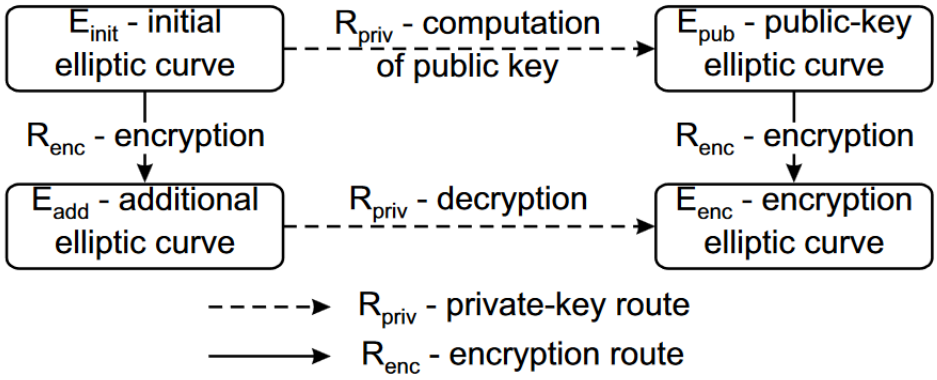

Figure 3.2: [53, Figure 3] The encryption protocol.

Remark 3.3.5. We note that the validity of this protocol depends on the routes being commutative, $R_{enc}R_{priv} = R_{priv}R_{enc}$. This commutativity is derived by the fact that isogenies in the isogeny star can be represented as ideals in the ideal class group $Cl(\mathcal{O})$ (Theorem 2.4.7), which is commutative.

As will be discussed in more detail in Chapter 4, the Couveignes-Rostovtsev-Stolbunov protocol was attacked by Childs et al. with subexponential complexity [15]. This was enough for cryptographers to consider the protocol insecure. But besides the security concerns, it is impractical for high-speed cryptography. For 128-bit security, one step in the isogeny walk exchange takes about 200 seconds [57, Table 1]. A full route in the isogeny

star is composed of many steps, making the full protocol take many minutes to hours for completion. For comparison, an implementation of ECDH on a C5509A DSP processor can perform a full key exchange in around 100 milliseconds [35, Section 5.2].

Remark 3.3.6. An attempted speed-up of this protocol was done in [20] by improving the calculations of walks in the isogeny star, resulting in one isogeny walk taking (in worst-case scenario) around 520 seconds. The authors of CSIDH claim that even with these speedups (and other potential optimizations), without major changes to the protocol, it will not be practical.

3.4 CSIDH

The CSIDH protocol is a reincarnation of the ideas of Couveignes, Rostovtsev, and Stolbunov, which attempts to fix the performance issues. The major difference between CSIDH and the previous ideas is the use of supersingular elliptic curves rather than ordinary ones.

The proposals of Couveignes, Rostovtsev, and Stolubunov used the action of ideals in the class group $Cl(\mathcal{O})$ (Theorem 2.4.7) where $\mathcal{O}$ is an order in a quadratic imaginary field. For supersingular elliptic curves, the endomorphism ring over the *algebraic closure* is an order in a quaternion algebra (Theorem 2.3.36). By choosing a suitable elliptic curve, one such that its endomorphism ring over some field of prime power is an order in a quadratic imaginary field (Proposition 3.4.1), we can apply the formulation of the previous 3 authors. More specifically, CSIDH restricts itself to supersingular curves defined over $\mathbb{F}_p$. Instead of considering the full ring of endomorphisms, CSIDH considers the subring of $\mathbb{F}_p$-rational endomorphisms, which is isomorphic to an order in an imaginary quadratic field.

Proposition 3.4.1. [11, Proposition 8] Let $p \geq 11$ be a prime such that $p \equiv 3 \pmod 8$ and $E/\mathbb{F}_p$ be a supersingular elliptic curve. Then $\mathrm{End}(E) = \mathbb{Z}[\pi]$ (π is the Frobenius endomorphism, Definition 2.3.41) if and only if there exists a Montgomery coefficient (Definition 2.3.8) $A \in \mathbb{F}_p$ such that E is $\mathbb{F}_p$-isomorphic to the Montgomery curve $M_{A,1}$: $y^2 = x^3 + Ax^2 + x$. Moreover, if A exists, then it is unique.

Why supersingular? [11, Section 4] The main bottleneck in the Couveignes-Rostovtsev-Stolbunov scheme is the construction of a curve with highly composite[1] order. This becomes trivial when using supersingular curves because such curves over $\mathbb{F}_p$ for $p \geq 5$ have exactly $p+1$ rational points. Thus, when choosing parameters for CSIDH, we choose primes of the form $p = 4 \cdot l_1 l_2 \dots l_n - 1$ where l_i are small distinct odd primes.

For completeness, we include the description of the key exchange [11, Section 6] although it is almost identical to that of the previous subsection.

Setup. Two parties, Alice and Bob agree on

1. A large prime $p = 4 \cdot l_1 \dots l_n - 1$ where l_i are small distinct odd primes.
2. The supersingular elliptic curve $E_0 : y^2 = x^3 + x$ defined over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O} = \mathbb{Z}[\pi]$.

Private Key. The private keys are n-tuples $(e_1, \dots, e_n)$ of integers, each sampled randomly from a range $\{-m, \dots, m\}$. These integers represent the ideal class $[\mathfrak{a}] = [\mathfrak{l}_1^{e_1} \dots \mathfrak{l}_n^{e_n}] \in Cl(\mathcal{O})$, where $\mathfrak{l}_i = (l_i, \pi - 1)$.

[1] A highly composite number is a positive integer that has more divisors that any smaller positive integer.

Public Key. The public keys are the Montgomery coefficients $A \in \mathbb{F}_p$ of the elliptic curve $[\mathfrak{a}]E_0 : y^2 = x^3 + Ax^2 + x$ obtained by applying the action of $[\mathfrak{a}]$ to the curve E_0.

Key exchange. Suppose Alice an Bob have key pairs $([\mathfrak{a}], A)$ and $([\mathfrak{b}], B)$. Upon receiving Bob's public key $B \in \mathbb{F}_p \backslash \{\pm 2\}$, Alice verifies that the elliptic curve $E_B : y^2 = x^3 + Bx^2 + x$ is indeed supersingular. She then applies the action of her secret key $[\mathfrak{a}]$ to E_b to comput the curve $[\mathfrak{a}]E_B = [\mathfrak{a}][\mathfrak{b}]E_0$. Bob proceeds analogously with his own secret $[\mathfrak{b}]$ and Alice's public key A to compute the curve $[\mathfrak{b}]E_a = [\mathfrak{b}][\mathfrak{a}]E_0$.

Secret Key. The shared secret is the Montgomery coefficient S of the common secret curve $[\mathfrak{a}][\mathfrak{b}]E_0 = [\mathfrak{b}][\mathfrak{a}]E_0$ written in the form $y^2 = x^3 + Sx^2 + x$, which is the same for Alice and Bob due to the commutativity of $Cl(\mathcal{O})$ (Theorem 2.4.7).

3.5 CGL Hash Function

We now move from the world of key-exchanges to the world of hash functions. In particular, we look at the hash function by Charles, Goren, and Lauter [13]. Hash functions will play an important role when discussing the application of the BUFF transform to SQISign in Chapter 6.

Charles et al. proposed constructing a hash function from expander graphs (Definition 2.5.11). The input of the hash is used as directions for walking along the graph, and the ending vertex is the output of the hash function. The authors choose to investigate this idea for two families of Ramanujan graphs (Definition 2.5.8) which are optimal expander graphs. The two families are (1) the isogeny graphs of supersingular elliptic curves and (2)

a Cayley graph by Lubotzky, Phillips, and Sarnak [45], where vertices are matrices in the projective special linear group $PSL(2,\mathbb{F}_p)$. In this section, we concern ourselves with the expander graph formed by supersingular elliptic curves.

We consider the first graph, with vertices as the j-invariants of supersingular elliptic curves, and edges being l-isogenies, l a different prime from p. In 1990, Pizer showed that this graph, denoted by $G(p,l)$, is Ramanujan [50]. We give an example of hashing when the prime $l = 2$, but it can be easily generalized to any prime. $G(p,l)$ is connected and $l+1$-regular [18, Theorem 47], thus for $l = 2$, there are 3 edges connected to any vertex.

A feature of the hashing construction is that backtracking is not allowed, hence we consider only 2 of the possible 3 edges. For a supersingular elliptic curve $E_1 : y^2 = f(x)$, there are three non-trivial 2-torsion points. This is because $\ker([2]) \cong \mathbb{Z}_2 \times \mathbb{Z}_2$ (Proposition 2.3.26), and the points are $P_i = (x_i, 0)$ where

$$f(x) = (x - x_1)(x - x_2)(x - x_3).$$

The three 2-isogenies correspond to taking the quotient of E_1 by $\langle P_i \rangle$ (Proposition 2.3.18). Suppose we choose the isogeny $\phi : E_1 \to E_2 = E_1/\langle P_1 \rangle$, then $\phi(P_2) = \phi(P_3)$ (Proposition 2.3.32). Furthermore, the dual isogeny $\hat{\phi} : E_2 \to E_1$ has kernel $\phi(P_2)$ (Proposition 2.3.31). Thus for the next step in the walk, we factor $g(x)$ in the defining equation for $E_2 : y^2 = g(x)$, and we have

$$y^2 = (x - x(\phi(P_2)))(x - x_4)(x - x_5).$$

Here $x(\phi(P_2))$ denotes the x-coordinate of $\phi(P_2)$. Because we do not allow backtracking,

the kernel of the next isogeny should be $\langle(x_4,0)\rangle$ or $\langle(x_5,0)\rangle$ and not $\langle(x(\phi(P_2)),0)\rangle$. We continue this process, as many times as our input requires.

Say we wish to hash some input $M = b_1 b_2 \dots b_n$ of n bits. Each bit represents a decision to select one of the two options for the kernel of the following isogeny. The decision is made by setting some ordering on choosing the kernel. For example, we could make the order by choosing the kernel $\langle(x_1,0)\rangle$ versus $\langle(x_2,0)\rangle$ if $x_1 > x_2$ and the bit is 0, and visa-versa if the bit is 1. The hash output is represented as the j-invariant of the final elliptic curve in the walk.

(Pseudo-)Randomness. The randomness of the output of the CGL hash function comes from the fact that random walks on expander graphs tend to uniform distribution.

Theorem 3.5.1. [34, Lecture 10, Corrolary 10.6] A random walk on an expander graph with N vertices tends to uniform distribution after $O(\log(N))$ steps.

3.6 SIDH

Before CSIDH was proposed, Jao and De Feo aimed to address the two problems of performance and security of the Couveignes-Rostvostsev-Stolbunov scheme.

Similar to CSIDH and the CGL hash function, SIDH uses supersingular elliptic curves, rather than ordinary elliptic curves. The endomorphism ring of ordinary elliptic curves is commutative while the endomorphism ring of supersingular elliptic curves is not (Theorem 2.3.36), hence we cannot apply the formulation of the class group action as before. The non-commutativity of the endomorphism ring raises an issue when attempting to perform a Diffie-Hellman type key-exchange. To fix this, Jao and De Feo introduced so-called

auxiliary points. These auxiliary points created extra avenues of attacks ultimately leading to the demise of SIDH; see Chapter 4.

Setup. Two parties Alice and Bob agree on

1. A prime $p = l_A^{e_A} l_B^{e_B} \cdot f \pm 1$, where l_A, l_B are small primes and f is chosen such that p is prime.
2. A supersingular curve E_0 defined over $\mathbb{F}_{p^2}$.
3. A basis $\{P_A, Q_A\}$ which generates $E_0[l_A^{e^A}]$.
4. A basis $\{P_B, Q_B\}$ which generates $E_0[l_B^{e^B}]$.

Private Keys.

1. Alice picks at random two integers $m_A, n_A \in \mathbb{Z}/l_A^{e_A}\mathbb{Z}$ and computes the isogeny $\phi_A : E_0 \to E_A$ with kernel $K_A = \langle [m_A]P_A + [n_A]Q_B \rangle$.
2. Bob picks at random two integers $m_B, n_B \in \mathbb{Z}/l_B^{e_B}\mathbb{Z}$ and computes the isogeny $\phi_B : E_0 \to E_B$ with kernel $K_B = \langle [m_B]P_B + [n_A]Q_B \rangle$.

The private key for Alice (resp. Bob) is the isogeny ϕ_A (resp. ϕ_B) or equivalently the kernel K_A (resp. K_B).

Public Keys. Alice's (resp Bob's) public key is the tuple $(E_A, \phi_A(P_B), \phi_A(P_B))$ (resp. $(E_B, \phi_B(P_A), \phi_B(P_A))$).

Key Exchange:

1. Alice computes the isogeny $\phi'_A : E_B \to E_{AB}$ with kernel $K_{AB} = \langle [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A) \rangle$.

2. Bob computes the isogeny $\phi'_B : E_A \to E_{BA}$ with kernel $K_{BA} = \langle [m_B]\phi_A(P_B) + [n_B]\phi_A(Q_B) \rangle$.

Alice and Bob compute the shared secret $j(E_{AB}) = j(E_{BA})$. The curves E_{AB} and E_{BA} are isomorphic as they are exactly the images of the pushforward isogenies (Definition 2.3.45) $[\phi_A]_*\phi_B$ and $[\phi_B]_*\phi_A$.

In fact, while all that is needed for a shared secret is a common j-invariant, Leonardi has shown that the ending curves are in fact equal [44]. Thus instead of using the j-invariant of the elliptic curve as the shared secret, the parties could use the coefficients defining the elliptic curve as the shared secret.

$$\begin{array}{ccc} E_0 & \xrightarrow{\phi_A} & E_A \\ \phi_B \downarrow & & \downarrow \phi'_B \\ E_B & \xrightarrow{\phi'_A} & E_{AB} = E_{BA} \end{array}$$

Figure 3.3: Diagram of the SIDH key exchange protocol.

ϕ_A and ϕ_B can be viewed as random routes in the the l_a-isogeny graph and l_b-isogeny graph of supersingular elliptic curves.

Chapter 4

Cryptanalysis

We devote this chapter to discussions about the security of the cryptographic protocols discussed in Chapter 3. We begin by presenting results on the most fundamental problems in isogeny-based cryptography, namely, the *isogeny problem*. That is, the problem of computing an (arbitrary) isogeny between two given elliptic curves. This problem is naturally divided into two sub-problems, when the elliptic curves are ordinary or supersingular. Since most isogeny-based protocols do not require arbitrary isogenies, but other specific properties, such as the degree of the isogeny, we discuss these problems as well. We note that there are a plethora of problems within isogeny-based cryptography, and for the sake of brevity, we direct the interested reader to the website[1] "is SIKE broken yet" following some isogeny-based schemes and their security.

There are many ways to attack cryptographic protocols, algebraic attacks, side-channel attacks, brute-force attacks, etc. See for example the textbook by Stallings [63]. The

[1] `https://issikebrokenyet.github.io/about.html`

types of attacks we consider here are algebraic attacks. That is, we look at the underlying mathematics of a cryptosystem and determine if there are exploits.

In general, when analyzing a cryptosystem algebraically, we consider: (1) what is the public information shared with all parties, and (2) what is the information shared between two communicating parties. From this public/transmitted information, we attempt to recover secret information by using algebraic techniques.

Remark 4.0.1. We define the notations $O, \widetilde{O}, L_n$ in Section 2.1.

4.1 Isogeny Problems

4.1.1 Ordinary Isogeny Problems

Problem 3. *(Ordinary Isogeny Problem)* Given two isogenous ordinary elliptic curves E and E' defined over $\mathbb{F}_p$, find a (non-trivial) isogeny $\phi : E \to E'$ defined over $\mathbb{F}_p$.

In [32], Galbraith shows that Problem 3 can be solved probabilistically in exponential time. More specifically, the algorithm has (worst-case) expected time $O(p^{3/2} \ln p)$ and expected space $O(p \ln p)$. The algorithm is essentially a meet-in-the-middle attack [63, Page 210] on the isogeny graphs of E and E'.

The protocols we have covered which use ordinary elliptic curves (the protocols by Couveignes-Rostovtsev-Stolbunov, Sections 3.2 and 3.3), also have the condition that the endomorphism rings are equal.

Problem 4. Given two isogenous elliptic curves E_0 and E_1 with equal endomorphism ring $\mathcal{O}$, find a (non-trivial) isogeny $\phi : E \to E'$ defined over $\mathbb{F}_p$.

Currently, the best solution to this problem is built on the work done by Kuperberg [42], giving a subexponential-time quantum algorithm to the *hidden shift problem* for finitely generated abelian groups.

Problem 5. [12, Definition 1.1] *(Abelian Hidden Shift Problem)* Given a finite abelian group $(G,+)$, a set X, and oracle (Definition 2.1.20) access to injective functions $f_1, f_2 : G \to X$ for which there exists an $s \in G$ such that $f_1(g) = f_2(g+s)$ for all $g \in G$, find s.

In [15], Childs, Jao, and Soukharev reduced the problem of constructing isogenies in a hard homogeneous space to the abelian hidden shift problem. Their algorithm performance is summarized in the following theorem.

Theorem 4.1.1. [15, Theorem 3.3] Assuming the generalized Riemann hypothesis, Problem 4 can be solved with time $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2})$.

Remark 4.1.2. Although CSIDH uses supersingular elliptic curves, the attack by Childs et. al. [15] was extended by Biasse, Jao, and Sankar [9] to the supersingular case. This is possible because CSIDH uses the same class group action 2.4.7. This attack runs with similar complexity and is (currently) one of the fastest attacks against CSIDH.

4.1.2 Supersingular Isogeny Problems

The ordinary isogeny problem is rephrased to the supersingular case as follows.

Problem 6. *(Supersingular Isogeny Problem)* Given two isogenous supersingular elliptic curves E and E', find a (non-trivial) isogeny $\phi : E \to E'$.

The algorithm by Galbraith [32] for the ordinary analog, using a meet-in-the-middle attack, can be again used in the supersingular case on the supersingular isogeny graph, see [33, Section 8.2]. The time and space complexity of such an attack would be $\widetilde{O}(p^{1/2})$.

For protocols using supersingular isogenies, the following problem is usually considered.

Definition 4.1.3. *(l-Isogeny Path Problem)* Given two supersingular elliptic curves, E and E', find an l-isogeny $\phi : E \to E'$.

For this problem, it is not known whether the constraint of the degree of the isogeny makes this problem easier or harder than the supersingular isogeny problem. On one hand, it may make the problem easier as there are fewer possible isogenies to search through. On the other hand, it may make it harder for the same reason, there are fewer isogeny paths between two given elliptic curves. The fastest algorithms are still those for Problem 6.

The CGL hash function derives its pre-image resistance and collision resistance properties from the difficulty of Problem 4.1.3. The pre-image resistance property (2.1.1) comes from the fact that given a hash, equivalently the j-invariant of a vertex in an l-isogeny graph, finding a path from the starting vertex to the hash vertex is Problem 4.1.3. This path corresponds to the message, thus if solving the above problem was easy, the CGL hash function would not have pre-image resistance. Similarly, the collision resistance property (2.1.1) is also derived from this problem. Constructing a collision is equivalent to, finding two different l-isogenies $\phi, \psi : E \to E'$.

Similarly, the SIDH protocol (Section 3.6) partially derived[2] its security from this

[2]Past tense because SIDH is broken.

problem. Recall that the secret keys in the protocol are the secret isogenies $\phi_A : E_0 \to E_A$ and $\phi_B : E_0 \to E_B$, with the public keys $(E_A, \phi_A(P_B), \phi_A(Q_B))$ and $(E_B, \phi_B(P_A), \phi_B(Q_A))$. Solving the above problem would allow an attacker to find some l-isogeny $\psi : E_0 \to E_A$. Given the parameters in SIDH, $\psi = \phi_A$ ([60, Lemma 3.2]), hence uncovering the secret key. Although this problem is still believed to be difficult, SIDH was attacked by a different method as we comment on later.

4.1.3 Equivalence to Endomorphism Problems

Suppose that we wish to construct an isogeny $\phi : E \to E'$. We comment on how knowledge of the endomorphism rings affects this problem.

In the case of ordinary elliptic curves with equal endomorphism rings, we can describe isogenies as ideal classes in the class group $Cl(\mathcal{O})$ (Theorem 2.4.7). Although, given the class group $Cl(\mathcal{O})$ and an isogeny $\phi : E \to E'$, it does not seem possible to efficiently determine an ideal $\mathfrak{a} \in Cl(\mathcal{O})$ such that $E' = \mathfrak{a} \cdot E$.

In contrast to the supersingular case, we have the following result.

Theorem 4.1.4. [14, Page 5] For isogenous supersingular elliptic curves E_1, E_2 with corresponding endomorphism rings $\mathrm{End}(E_1)$, $\mathrm{End}(E_2)$ we have:

- Given E_1, E_2, $\mathrm{End}(E_1)$, and $\mathrm{End}(E_2)$, one can find an isogeny $E_1 \to E_2$ in polynomial time.
- Given E_1, E_2, $\mathrm{End}(E_1)$, and an isogeny $E_1 \to E_2$, one can compute $\mathrm{End}(E_2)$ in polynomial time.

Thus, for isogeny-based protocols using supersingular curves, the following problem is expected to be hard.

Problem 7. *(Supersingular Endomorphism Problem)* Given a supersingular elliptic curve E, compute its endomorphism ring $\mathrm{End}(E)$.

Currently, the best algorithms for this problem are based on Theorem 4.1.4. That is, use some elliptic curve E_0 with known endomorphism ring $\mathrm{End}(E_0)$, and construct an isogeny $\phi : E_0 \to E$ (which is difficult by Problem 6), and apply the theorem.

4.2 SIDH

We use SIDH as an example of when the difficulty of the aforementioned problems does not secure a protocol. Indeed, isogeny-based protocols have information publicly available or transmitted between communicating parties (which might be recovered by eavesdropping), that is not purely elliptic curves and secret isogenies. The protocols by Couveignes, Rostovtsev, and Stolbunov, as well as CSIDH, have a requirement on the degrees of the isogenies being used, a requirement of the shape of the isogeny star, etc. These requirements are usually put in place for efficiency purposes, or as we saw in SIDH, auxiliary points are transmitted to make the protocol function correctly.

The security problem of SIDH can be stated as follows.

Problem 8. *(SIDH Problem)* As in the notation of the SIDH protocol given in Section 3.6, given

$$E_0, E_A, E_B, P_A, Q_A, P_B, Q_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), l_A^{e_A}, l_B^{e_B}, f,$$

find $j(E_{AB})$, the shared secret.

Remark 4.2.1. In some instantiations of SIDH, such as SIKE [5], the endomorphism ring $\mathrm{End}(E_0)$ of the starting curve is also public information.

These auxiliary points were exploited first by Castryk and Decru [10], then (independently) by Maino and Martindale [46] and improved a few days later by Robert [52]. The current state of SIDH is that it can be broken in polynomial time, even with a starting curve E_0 with an unknown endomorphism ring.

The attacks are beyond the scope of this thesis, but we comment that the main tool used in the attack was based on a result of Kani from 1997 [40] on the reducibility of subgroups of principally polarized abelian surfaces. The fact that this attack was only found about a decade after the publishing of SIDH demonstrates that isogeny-based cryptography is still developing and much more cryptanalysis must be done on the existing protocols.

Chapter 5

SQISign

SQISign is an isogeny-based digital signature scheme, first proposed by De Feo, Kohel, Leroux, Petit and Wesolowski in [29]. For the post-quantum standardization, NIST announced in [49] that after three rounds of evaluation and analysis, the digital signatures that will be standardized are CRYSTALS-Dilithium [6], FALCON [31], and SPHINCS+ [3].

CRYSTALS-Dilithium and FALCON are lattice-based, while SPHINCS+ is hash-based. NIST rejected all other digital signature schemes, and announced a call for additional proposals. NIST stated that they are primarily interested in schemes that are *not lattice-based*. In addition to this desire, they are also interested in signature schemes that have *short signature sizes* and *fast verification times*.

SQISign was the only isogeny-based scheme submitted to NIST, and it currently boasts the smallest combined size of public keys and signatures. It also has relatively fast verification times, hence satisfying NIST's interests.

SQISign is a Σ-protocol (Definition 2.1.3) turned into a digital signature (see Section 2.1) by the Fiat-Shamir transform (Definition 2.1.6). Following the SQISign specification [14], we first take a high-level look at the Σ-protocol. We then follow with a more detailed discussion of the algorithms making up the digital signature: key-generation, signing, and verification. We then end the section with a discussion on the security of SQISign.

Before proceeding to the Σ-protocol, we note that we will only present the main algorithms required in SQISign. A short description of the algorithms referenced can be found in the appendix (Appendix 1). Furthermore, information public to parties participating in the SQISign scheme, such as $E_0, \mathcal{O}_0, T, f, g, D_{\text{chall}}, D_{\text{com}}$, etc., can also be found in the appendix (Appendix 7).

5.1 Σ-Protocol

Suppose a prover, Peggy, wants to prove knowledge of an elliptic curve endomorphism ring to a verifier, Victor. That is, Peggy knows an elliptic curve E_A and its corresponding endomorphism ring $\text{End}(E_A)$. The process is as follows:

1. *(Setup)* Peggy and Victor agree on a public elliptic curve E_0 with known endomorphism ring $\text{End}(E_0)$.

2. *(Commitment)* Peggy generates a random isogeny $\phi_{\text{com}} : E_0 \to E_1$ of degree D_{com}, and sends E_1 to Victor. Since Peggy knows an isogeny $E_0 \to E_1$ and $\text{End}(E_0)$, she can compute $\text{End}(E_1)$ in polynomial time (Theorem 4.1.4).

3. *(Challenge)* Victor generates a random *cyclic* isogeny $\phi_{\text{chall}} : E_1 \to E_2$ of degree

$D_{\text{chall}} = 2^f 3^g$ and sends it to Peggy. A cyclic isogeny is an isogeny where the kernel is cyclic.

4. *(Response)* Peggy, using her knowledge of $\text{End}(E_1)$ and $\phi_{\text{chall}} : E_1 \to E_2$, computes $\text{End}(E_2)$ in polynomial time (Theorem 4.1.4). Then, using knowledge of $\text{End}(E_A)$ and $\text{End}(E_2)$, she can compute in polynomial time an isogeny $\phi_{\text{resp}} : E_A \to E_2$ (Theorem 4.1.4). Peggy computes such isogenies until $\widehat{\phi_{\text{chall}}} \circ \phi_{\text{resp}}$ is cyclic before sending ϕ_{resp} to Victor.

5. *(Verification)* Victor verifies that ϕ_{resp} is indeed an isogeny from E_A to E_2, and $\widehat{\phi_{\text{chall}}} \circ \phi_{\text{resp}}$ is cyclic.

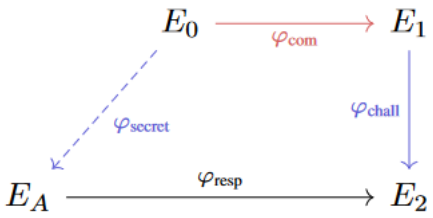

Figure 5.1: [14, Figure 1] The SQISign interactive Σ-protocol.

At the heart of SQISign is the Deuring correspondence (Theorem 2.3.37). It provides an equivalence of supersingular elliptic curves and left ideals for a maximal order of a quaternion algebra. A summary of the Deuring correspondence can be seen in Table 2.1.

The above Σ-protocol is then transformed into a digital signature using the Fiat-Shamir transform.

5.2 Key Generation

The key generation algorithm generates a public key `pk` and secret key `sk` for a party who wishes to sign a message. At a high level, the KeyGen algorithm returns the public key, a random curve E_A, and the secret key, which contains data to compute $\mathrm{End}(E_A)$. The full algorithm can be read in Algorithm 1. We give an overview of the key generation algorithm in what follows.

Lines $1-12$ compute the secret isogeny ϕ_{secret}. First, we sample random ideals I_{secret} of random norm D_{secret} and then run KeyGenKLPT (Algorithm 7) to find a generator α of an equivalent ideal of norm $2^{\texttt{KLPT\_keygen\_length}}$. Assuming a run of KeyGenKLPT is successful, we translate the generator α into the corresponding ideal $J_{\text{secret}} = \chi_{I_{\text{secret}}}(\alpha)$ (Proposition 2.2.19). Finally we use the Deuring correspondence (Table 2.1), and more specifically the algorithm IdealToIsogenyEichler (Algorithm 6) to convert J_{secret} into the corresponding isogeny ϕ_{secret}.

Lines $13-19$ compute additional information to speed up later computations. Firstly, the secret isogeny ϕ_{secret} is normalized using Normalized (Algorithm 8). Then, the basis $B_{0,T}$ of $E_0[T]$ is pushed through ϕ_{secret} to compute the basis $B_{A,T}$ of $E_A[T]$. Then, a basis (P,Q) of $E_0[2^f]$ is computed using CompleteBasis (Algorithm 9). We further push Q through ϕ_{secret}, and the resulting point $Q' = \phi_{\text{secret}}(Q)$ corresponds to the dual of the kernel of the last 2^f isogeny composing ϕ_{secret}. Note that we relabel the point Q' as Q (line 19).

The output of the keygen algorithm is $\texttt{pk} = E_A$ and $\texttt{sk} = (\alpha, B_{A,T}, Q)$.

Remark 5.2.1. A note on lines $7-9$ of the key generation algorithm (Algorithm 1). The

CONTINUE command is used to skip lines 10 and 11 of the while loop; for efficiency purposes. If KeyGenKLPT (line 6) fails, there is no reason to proceed with IdealToIsogenyEichler (line 11). The CONTINUE command is used similarly in the signing algorithm (Algorithm 2), and we will comment on the implications in the corresponding section.

Remark 5.2.2. On line 11 of the key generation algorithm, and line 44 of the signing algorithm (Algorithm 2), we write $(\phi_{\text{secret}}, \_, \texttt{found}) := \text{IdealToIsogenyEichler}_{2^\bullet}(J_{\text{secret}}, \mathcal{O}_0, B_{0,T})$ and $(\_, \textbf{zip}, \texttt{found}) := \text{IdealToIsogenyEichler}_{2^\bullet}(J, J_{\text{secret}}, B_{A,T}, Q)$. The $\_$ signifies that we ignore the respective output of IdealToIsogenyEichler (Algorithm 6).

Algorithm 1 SQISign.KeyGen(1^λ)

Input: 1^λ where λ is the security parameter
Output: Secret signing key `sk` and public verification key `pk`
Output: `found` a boolean indicating whether computation succeeded

1: Set `found`:=FALSE, `counter`:= 0
2: **while** `found`=FALSE and `counter` $<$ `SQISign_keygen_attempts` **do**
3: `counter`:=`counter`+1
4: Select a random `KLPT_secret_key_prime_size`-bit prime $D_{\text{secret}} \equiv 3 \pmod 4$
5: Set I_{secret} to be a random ideal of norm D_{secret}
6: $(\alpha, \texttt{found}) := \text{KeyGenKLPT}_{2^\bullet}(I_{\text{secret}})$
7: **if** not `found` **then**
8: CONTINUE
9: **end if**
10: $J_{\text{secret}} := \chi_{I_{\text{secret}}}(\alpha)$
11: $(\phi_{\text{secret}}, \_, \texttt{found}) := \text{IdealToIsogenyEichler}_{2^\bullet}(J_{\text{secret}}, \mathcal{O}_0, B_{0,T})$
12: **end while**
13: **if** `found` **then**
14: $\alpha := \overline{\alpha}$

15: $E_A, \phi_{\text{secret}} := \text{Normalized}(\phi_{secret})$
16: $B_{A,T} := \phi_{\text{secret}}(B_{0,T})$
17: Let P be a point generating $\ker \phi_{\text{secret}} \cap E_0[2^f]$
18: $(P,Q) := \text{CompleteBasis}_{2^f,p+1}(E_0,P)$
19: $Q := \phi_{secret}(Q)$
20: Set $\texttt{pk} := E_A$
21: Set $\texttt{sk} := (\alpha, B_{A,T}, Q)$
22: **end if**
return $\texttt{sk}, \texttt{pk}, \texttt{found}$

5.3 Signing

Once a party has successfully run the key generation algorithm, they can sign a message $\texttt{msg}$ using Algorithm 2 with their secret key $\texttt{sk}$, and produce a signature σ.

The signing algorithm can be divided into three main parts: (1) commitment, (2) challenge, and (3) response, as described by the Fiat-Shamir transform (Definition 2.1.6) for a Σ-protocol. The commitment part generates a random isogeny $\phi_{\text{com}} : E_0 \to E_1$. The challenge part hashes the concatenation of $\texttt{msg}$ and $j(E_1)$, $\texttt{msg} \,||\, j(E_1)$, and maps this hash to a point generating the challenge isogeny $\phi_{\text{chall}} : E_1 \to E_2$. The response part generates the response isogeny $\phi_{\text{resp}} : E_A \to E_2$. The full algorithm can be read in Algorithm 2. We give an overview of the signing algorithm in what follows.

Precomputation. Lines $1-4$ extract the information from the secret key $\texttt{sk} = (\alpha, B_{A,T}, Q)$ and (re-)compute the ideals I_{secret} and J_{secret} from key generation.

Commitment. Lines $5-9$ compute the (random) commitment isogeny $\phi_{\text{com}} : E_0 \to E_1$. This is done by first sampling random integers a, b with $\gcd(a, b, D_{\text{com}}) = 1$, and computing

the kernel $K_{\text{com}} = [a]P_0 + [b]\theta(P_0)$. Finally, the algorithm KernelToIsogeny (Algorithm 10) converts K_{com} into an isogeny, and is normalized using Normalized (Algorithm 8).

The corresponding ideal I_{com} to the isogeny ϕ_{com} is computed as well, and will be used in the challenge and response parts.

Challenge. Lines $10-28$ compute the (random) challenge isogeny $\phi_{\text{chall}} : E_1 \to E_2$, its corresponding ideal I_{chall}, and *compression* information for $\widehat{\phi_{\text{chall}}}$.

First, recall that the challenge in a Σ-protocol (Definition 2.1.3) is randomly chosen by the verifier. The Fiat-Shamir transform (Definition 2.1.6) removes the need for interaction by replacing the randomness from the verifier, with the (pseudo) randomness of a hash function H.

Indeed, for generating the challenge isogeny $\phi_{\text{chall}} : E_1 \to E_2$, we first define the hash $K_{\text{chall}} = H(\texttt{msg}, E_1)$, which computes $a = \texttt{SHAKE256}(\texttt{msg} \,||\, j(E_1))$[1], and then returns $P_1 + [a]Q_1$ where P_1, Q_1 form a basis of $E_1[D_{\text{chall}}]$. Finally, the challenge isogeny is computed from the point K_{chall} using KernelToIsogeny (Algorithm 10) and Normalized (Algorithm 8).

For computing the ideal I_{chall} corresponding to ϕ_{chall}, we compute the pushforward (Definition 2.3.45) $[I_{\text{com}}]_*\text{KernelDecomposedToIdeal}_{D_{\text{chall}}}(a,b)$. Converting from isogenies to ideals using KernelDecomposedToIdeal (Algorithm 15) is done by translating points on E_0 [14, section 2.6.4].

More specifically, the integers a and b correspond to $K_{\text{chall}} = [a]P_1 + [b]Q_1$ and are

[1]The SHAKE256 hash function is described for example in [25].

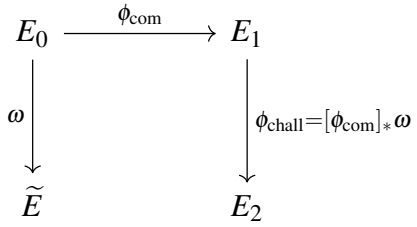

Figure 5.2: Commutative diagram showing the computation of I_{chall}.

computed using NormalizedDlog (Algorithm 11)[2]. Define a point $L = [a]P_{D_{\text{chall}}} + [b]Q_{D_{\text{chall}}}$, where $P_{D_{\text{chall}}}$ and $Q_{D_{\text{chall}}}$ form a basis of $E_0[D_{\text{chall}}]$. Then, the isogeny $\omega : E_0 \to \widetilde{E}$ with kernel $\langle L \rangle$ is "parallel" (see Figure 5.2) to ϕ_{chall}. Thus, the pushforward $[\phi_{\text{chall}}]_*\omega$ is the isogeny ϕ_{chall}. This works because $\gcd(D_{\text{com}}, D_{\text{chall}}) = 1$.

The rest of the challenge phase is responsible for computing the tuple $\mathbf{s} = (b_1, s_1, b_2, s_2)$, and a point Q such that $K_{\text{chall}} = [r]Q$. The tuple $\mathbf{s}$, corresponds to a *compression* of $\widehat{\phi_{\text{chall}}}$ as a composition of two isogenies. One of degree 2^f, and one of degree 3^g, as $D_{\text{chall}} = 2^f 3^g$. Suppose P_2 and Q_2 form a basis of $E_2[D_{\text{chall}}]$, which can be found using TorsionBasis (Algorithm 12). Then the kernel of the 2^f-isogeny is $P_2 + [s_1]Q_2$, and we switch P_2 and Q_2 if $b_1 = 1$. Similarly, the kernel of the 3^g-isogeny is $P_2 + [s_2]Q_2$, where we switch P_2 and Q_2 if $b_2 = 1$.

Response. Lines $29 - 45$ compute the response isogeny ϕ_{resp}, and its compression.

The response isogeny ϕ_{resp} is computed by first computing the ideal $K = \overline{I_{\text{secret}}} \cdot I_{\text{com}} \cdot I_{\text{chall}}$. By the Deuring correspondence (Table 2.1), this corresponds to the isogeny $\widehat{\phi_{\text{secret}}} \cdot \phi_{\text{com}} \cdot \phi_{\text{chall}} : E_A \to E_2$. Then, (repeatedly) using SigningKLPT (Algorithm 16), we find an

[2]Technically, NormalizedDlog is not nessecary. a, b are already known from the computation of K_{chall}, $a = 1$ and $b = \texttt{SHAKE256}(\texttt{msg} \,||\, j(E_1))$, although this information is not explicitly kept.

ideal J equivalent to K, with required norm 2^e, and cyclic. The ideal J then corresponds to the isogeny $\phi_{\text{resp}} : E_A \to E_2$. Instead of using the full isogeny ϕ_{resp} in the signature, we use a compressed version, as was done for the challenge isogeny. To compress the isogeny, this time we use IdealToIsogenyEichler (Algorithm 6), with output $\mathbf{zip} = (b, s_1, s_2, \ldots, s_g)$.

The final output of the signing algorithm is $\sigma, \texttt{found}$ where $\sigma = (\mathbf{zip}, r, \mathbf{s})$.

Remark 5.3.1. Lines $36-38$ again use the CONTINUE command within a while loop for efficiency purposes as was explained in Remark 5.2.1. Lines $40-42$ use CONTINUE in a different manner, it is the check that the ideal J is cyclic.

Algorithm 2 SQISign.Sign(sk, msg)

Input: Secret signing key sk and a message $\texttt{msg} \in \{0,1\}^*$

Output: Signature σ

Output: found a boolean indicating whether computation succeeded

1: Parse sk as $(\alpha, B_{A,T}, Q)$

2: Compute $n(\alpha) := 2^e D_{\text{secret}}$

3: Compute $I_{\text{secret}} := \mathcal{O}_0\langle \overline{\alpha}, D_{\text{secret}} \rangle$

4: Compute $J_{\text{secret}} := \mathcal{O}_0\langle \alpha, 2^e \rangle$

Commitment.

5: Sample a, b at random from interval $[1, D_{\text{com}}]$ until $\gcd(a, b, D_{\text{com}}) = 1$

6: $I_{\text{com}} := \mathcal{O}_0\langle a + b\overline{\theta}, D_{\text{com}} \rangle$

7: $K_{\text{com}} := aP_0 + b\theta(P_0)$, where $P_0 \in B_{0,D_{\text{com}}}$

8: $(E_1, \phi_{\text{com}}) := \text{Normalized}(\text{KernelToIsogeny}(K_{\text{com}}))$, where $\phi_{\text{com}} : E_0 \to E_1$

9: $(B_{1,D_{\text{chall}}}) = \phi_{\text{com}}(B_{0,D_{\text{chall}}})$. Write $B_{1,D_{\text{chall}}} = (P_1, Q_1)$

Challenge

10: $K_{\text{chall}} := H(\texttt{msg}, E_1)$

11: $(E_2, \phi_{\text{chall}}) := \text{Normalized}(\text{KernelToIsogeny}(K_{\text{chall}}))$, where $\phi_{\text{chall}} : E_1 \to E_2$

$(a,b) := \text{NormalizedDlog}_{D_{\text{chall}}}(E_1,(P_1,Q_1),K_{\text{chall}})$
$I_{\text{chall}} := [I_{\text{com}}]_*\text{KernelDecomposedToIdeal}_{D_{\text{chall}}}(a,b)$
$(K,P) := \text{CompleteBasis}_{D_{\text{chall},p+1}}(E_1,K_{\text{chall}})$
$P' := \phi_{\text{chall}}(P)$
$(P_2,Q_2) := \text{TorsionBasis}_{D_{\text{chall}},p+1}(E_2)$
$b_1 := 0$, $b_2 := 0$
$(a_0,a_1) := \text{NormalizedDlog}_{2^f}(E_2,[3^g]P_2,[3^g]Q_2,[3^g]P')$
If $a_0 \equiv 0 \pmod 2$, swap a_0 and a_1 and set $b_2 := 1$
$s_1 = a_0^{-1}a_1 \pmod{2^f}$
$a_0,a_1 := \text{NormalizedDlog}_{3^g}(E_2,([2^f]P_2,[2^f]Q_2),[2^f]P')$
If $a \equiv 0 \pmod 3$, swap a_0 and a_1 and set $b_2 := 1$
$s_2 := a_0^{-1}a_1 \pmod{3^g}$
$(E_1',\widehat{\phi_{\text{chall}}}) := \text{Normalized}(\text{KernelToIsogeny}(P'))$, where $E_1' = E_1$
$\mathbf{s} := (b_1,s_1,b_2,s_2)$
Use $\mathbf{s}$ to compute deterministically a point Q' so that $\langle P',Q'\rangle = E_2[D_{\text{chall}}]$
$Q := \widehat{\phi_{\text{chall}}}(Q')$
Compute r such that $K_{\text{chall}} = [r]Q$
Response
$K := \overline{I_{\text{secret}}} \cdot I_{\text{com}} \cdot I_{\text{chall}}$
$e :=$ `KLPT_signing_klpt_length`
`found` := FALSE
`counter` $:= 0$
while not `found` and `counter` $<$ `SQISign_response_attempts` **do**
    `counter` := `counter` $+ 1$
    J, `found` $:= \text{SigningKLPT}_{2^e}(K,I_{\text{secret}})$
    if not `found` **then**
        CONTINUE
    end if
    Compute α such that $J \cdot \overline{I_{\text{chall}}} = \langle \alpha, 2^e D_{\text{chall}}\rangle$
    if $\alpha/k \in \mathcal{O}_R(I_{\text{secret}})$ for any $k > 1$ **then**
        CONTINUE

42: **end if**
43: **end while**
44: $(\_, \mathbf{zip}, \texttt{found}) := \text{IdealToIsogenyEichler}_{2^\bullet}(J, J_{\text{secret}}, B_{A,T}, Q)$
45: $\sigma := (\mathbf{zip}, r, \mathbf{s})$
return $\sigma, \texttt{found}$

5.4 Verification

Once a signer has successfully signed a message `msg` with their secret key `sk` and produced the signature σ, they send σ and their public key `pk` to the verifier.

In the verification algorithm (Algorithm 3), the verifier first extracts the components of the signature $\sigma = (\mathbf{zip}, r, \mathbf{s})$.

The verification algorithm first reconstructs the response isogeny ϕ_{resp} from **zip**, and verifies it has the correct domain, E_A, and codomain, E_2, and is cyclic. This is done using the algorithm Decompress (Algorithm 13).

Then, the dual of the challenge isogeny $\widehat{\phi_{\text{chall}}}$ is reconstructed from $\mathbf{s}$. The reconstruction is checked against a calculation of ϕ_{chall} by recomputing $K_{\text{chall}} = H(\texttt{msg}, \texttt{pk})$, and following the procedure from the signing process. This is done in the algorithm DecompressAndCheck (Algorithm 14).

Algorithm 3 SQISign.Verify($\texttt{msg}, \sigma, \texttt{pk}$)

Input: A message `msg`
Input: A signature σ
Input: A public key `pk`
Output: `verfied`, a boolean indicating whether the verification passed

1: $\mathbf{zip}, r, \mathbf{s} := \sigma$
2: $E_2, Q_2 := \text{Decompress}_{\text{resp}}(\mathbf{zip}, \texttt{pk})$
3: $\texttt{verified} = \text{DecompressAndCheck}_{\text{chall}}(\mathbf{s}, E_2, Q_2, r, \texttt{msg})$
return `verified`

5.5 Security

In this section, we discuss the *existential unforgeability under chosen message attacks* (EUF-CMA) (Definition 2.1.8) security notion. This is the strongest notion of security for a digital signature. For SQISign, it is proved to satisfy EUF-CMA by first proving the two notions of weak honest-verifier zero-knowledge (wHVZK, Definition 2.1.10) and special soundness (Definition 2.1.9) for the Σ-protocol. Then we apply the following two theorems.

Theorem 5.5.1. [14, Theorem 9.1.4] Any Σ-protocol that satisfies special-soundness and wHVZK also satisfies security under passive impersonation attacks (IMP-PA, Definition 2.1.13).

Theorem 5.5.2. [14, Theorem 9.19] Suppose a Σ-protocol satisfies IMP-PA. Then the resulting digital signature obtained by applying the Fiat-Shamir transform (Definition 2.1.6) satisfies EUF-CMA security.

In [29], the authors prove that if one can break the special soundness of the SQISign Σ-protocol then there is an efficient algorithm for the following problem.

Problem 9. [29, Problem 1] *(Supersingular Smooth Endomorphism Problem)* Given a supersingular elliptic curve E, find a (non-trivial) cyclic endomorphism of E of smooth[3] degree.

[3] An integer x is n-smooth if all its prime factors are less than or equal to n. A number x is "smooth" if it is n-smooth for n relatively small to x

Notice the difference between this problem and the supersingular endomorphism problem (Problem 7). Here, we are only asked to compute a singular non-trivial cyclic endomorphism. Surprisingly, it turns out that this problem is equivalent to the supersingular endomorphism problem [29, Remark 8].

Theorem 5.5.3. [14, Theorem 9.1.12] If there is a probabilistic polynomial adversary that breaks the soundness of the protocol with probability w and expected running time r for the public key E_A, then there is an algorithm for the supersingular smooth endomorphism problem on E_A with expected running time $O(r/(w-1/c))$, where c is the size of the challenge space.

In [29], the authors assume that the set of response isogenies is computationally indistinguishable from random isogenies of equal degree from E_A. [29, Proposition 11 and Lemma 12] shows that under this assumption, the Σ-protocol satisfies wHVZK.

5.6 Key Sizes and Performace

SQSign has some of the smallest public key and signature sizes compared to other digital signature candidates. The key sizes are summarized in Table 5.1. Note that there are standard notions of security, NIST levels I, III, and V. A NIST level I has security against attacks requiring 2^{128} operations. Similarly, levels III and V have security against attacks requiring 2^{192} and 2^{256} operations respectively.

Performance is measured in units of CPU cycles. CPUs (central processing units) process information in cycles, and the speed of a CPU is measured in cycles per second. For example, a CPU with a rating of 1 GHz (Giga-Hertz) means that the CPU can perform

Parameter Set	Secret Key Size	Public Key Size	Signature Size
NIST I	782	64	177
NIST III	1138	96	263
NIST V	1509	128	335

Table 5.1: [14, Section 6.1, Table 1] SQISign key and signature sizes in bytes for each security level.

one billion cycles per second.

For SQISign, the authors tested the digital signature on an Intel Xeon Gold 6338 CPU (Ice Lake), compiled on Ubuntu with clang version 14. Their results are the median of 10 benchmark runs [14, Section 6.5], and can be viewed in Table 5.2.

Parameter Set	Key Generation	Sign	Verify
NIST I	2834	4781	103
NIST III	21359	38884	687
NIST	84944	160458	2051

Table 5.2: [14, Section 6.5, Table 2] SQISign performance of the reference implementation (with default GMP installing) in units of 10^6 CPU cycles, for each security level.

For example, a CPU with a speed rated at 1 GHz, one would expect the running time of the key generation algorithm at security level I to take 2.834 seconds. This applies to the current implementation of SQISign[4], although many improvements can be made. Improvements are possible as the mathematical operations used in SQISign, such as isogeny evaluations of points, are not optimized for efficiency, and is an open area of research.

[4]The current fastest implementation of SQISign can be found at `https://github.com/SQISign/the-SQISign`.

Chapter 6

Beyond UnForgeability Features (BUFF)

The standard security notion for digital signatures is *existential unforgeability under chosen message attacks* (EUF-CMA) (Definition 2.1.8). In most scenarios, EUF-CMA is sufficient, but depending on the use case of a digital signature, other attacks may be possible even with the guarantee of EUF-CMA. In [17], Cremers, Düzlü, Fiedler, Fischlin, and Janson define several security notions, the BUFF (Beyond UnForgeability Features) security, to protect against these attacks. Their work stems from various attacks on real-world protocols ("real-world" meaning protocols that are used in practice). Furthermore, the authors define a generic transformation, the BUFF transform, that achieves the security notions they define.

In this section, we introduce the BUFF security notions, and the BUFF transform. Then we discuss the work in [2] where Aulbach, Düzlü, Meyer, Struck, and Weishäupl analyze BUFF security of the additional signatures submitted to NIST, and in particular take a look at each security notion for SQISign.

We end the section with an implementation of the BUFF transform to a SageMath

implementation of SQISign by Santos, Eriksen, Meyer, and Pope[54], and discuss the performance results.

6.1 Security Notions and Transformation

Exclusive Ownership (S-CEO and S-DEO). An adversary is given a valid message-signature pair (msg,sig) for some public key pk, and asked to find a different public key $\overline{\texttt{pk}}$ which is still valid for (msg,sig). If this situation is hard, then we say the protocol satisfies the *conservative exclusive ownership* (S-CEO) property.

A related situation is when the adversary is also asked to find a different message $\overline{\texttt{msg}}$ such that $(\overline{\texttt{msg}}, \texttt{sig})$ verifies for $\overline{\texttt{pk}}$. If this situation is hard, we say the protocol satisfies the *destructive exclusive ownership* (S-DEO) property.

Lack of the S-CEO property would allow, for example, an adversary to claim a message-signature pair as their own by providing $\overline{\texttt{pk}}$. An attack [4] on an earlier version of the ACME protocol used by Let's Encrypt [7] exploited the lack of the S-CEO property.

Lack of the S-DEO property would demonstrate that there are signatures which are not unique to the message and public key; which is an undisireable property of digital signature schemes for various reasons, see e.g. [63].

Message-Bound Signatures (MBS). An adversary is asked to generate two distinct messages msg and $\overline{\texttt{msg}}$ with a signature sig and public key pk such that (msg,sig) and ($\overline{\texttt{msg}}$,sig) are valid for pk. If this situation is hard, we say the protocol satisfies the *message-bound signatures* (MBS) property.

Lack of this property allows, for example, users who have previously signed a message $\mathtt{msg}$, to claim later that they did not sign $\mathtt{msg}$, but instead had signed $\overline{\mathtt{msg}}$. The impossibility of disputing the authorship or validity of a statement is called *non-repudiation*, which has many implications in law, see for example [22, Page 10].

Weak Non Re-signability (wNR). An adversary is given a signature $\mathtt{sig}$ and public key $\mathtt{pk}$ for some *unknown* message $\mathtt{msg}$. The adversary is asked to find a different signature $\overline{\mathtt{sig}}$ and different public key $\overline{\mathtt{pk}}$ for which the *unknown* message $\mathtt{msg}$ still verifies. f this situation is hard, we say the protocol satisfies the (weak) Non Re-signability (wNR) property.

Lack of this property allows, for example, an adversary to claim knowledge of $\mathtt{msg}$ without in fact knowing it. This attack strategy can be employed in schemes that use message-hiding signatures, i.e. schemes that only transmit the signature and public key, and expect the verifier to already know the message. In [38], they exploit the lack of wNR to attack the DRKey protocol [41].

BUFF Transform. The BUFF transform is a generic transform that uses a hash function to turn a signature scheme into one that achieves the above security notions. It makes modifications to the signing algorithm $\mathtt{Sign}$ and verification algorithm $\mathtt{Verify}$ of the signature scheme. The key generation algorithm is left unchanged. We label these new signing and verification algorithms as $\mathtt{Sign}^*$ and $\mathtt{Verify}^*$ respectively (see Algorithms 4 and 5).

Theorem 6.1.1. [17, Theorem 5.5] Let Π be an EUF-CMA-secure signature scheme. Then the application of the BUFF transformation produces an EUF-CMA-secure scheme Π^* that additionally provides the properties of S-CEO, S-DEO, MBS, and NR assuming that the

Algorithm 4 $\texttt{Sign}^*(\texttt{sk},\texttt{msg})$

Input: Secret key `sk`.
Input: A message `msg`.
Output: The tuple $(\texttt{sig},\texttt{h})$.

1: $\texttt{h} := \texttt{H(msg,pk)}$
2: $\texttt{sig} := \texttt{Sign(sk,h)}$

return $(\texttt{sig},\texttt{h})$

Algorithm 5 $\texttt{Verify}^*(\texttt{pk},\texttt{msg},(\texttt{sig},\texttt{h}))$

Input: A public key `pk`.
Input: A message `msg`.
Input: A BUFFed signature tuple $(\texttt{sig},\texttt{h})$
Output: A bit, indicating PASS or FAIL

1: $\overline{\texttt{h}} := \texttt{H(msg,pk)}$
2: $\texttt{v} := \texttt{Verify(sk},\overline{\texttt{h}},\texttt{sig})$

return $(\texttt{v} = 1 \wedge \texttt{h} = \overline{\texttt{h}})$

hash function `H` used in the transformation is collision-resistant.

6.2 BUFF Security of SQISign

In [2], the authors analyze the BUFF security of most[1] of the additional digital signatures submitted to NIST. Their results [2, Table 1], show that only 4 of the 17 schemes that they analyzed satisfied full BUFF security. In particular, for SQISign, they detected an (impractical) attack against the S-CEO property, an attack against the wNR property, and claimed SQISign satisfies the S-DEO and MBS properties. In the rest of this subsection, we discuss each of the BUFF security properties for SQISign.

[1] They did not analyze schemes for which an attack against EUF-CMA was detected.

6.2.1 S-CEO

Before we describe the attack, we describe in more detail how the response isogeny ϕ_{resp} is computed from a signature `sig`. Recall from the signing algorithm (Algorithm 2) that the signature consists of a compressed description of the isogenies ϕ_{resp} and ϕ_{chall}. It takes the form

$$\texttt{sig} = \sigma = (\mathbf{zip}, r, \mathbf{s}) = (b, s^{(1)}, \ldots, s^{(n)}, r, b_2, s_2, b_3, s_3).$$

We are interested in the part of the signature corresponding to ϕ_{resp}, which is $b, s^{(1)}, \ldots, s^{(n)}$. These integers correspond to the kernels of the isogenies in the decomposition of ϕ_{resp},

$$E_A = E^{(1)} \xrightarrow{\phi^{(1)}} E^{(2)} \xrightarrow{\phi^{(2)}} \ldots \xrightarrow{\phi^{(n)}} E^{(n+1)} = E_2, \tag{6.1}$$

where $\deg(\phi^{(i)}) = 2^f$ (note that $\deg(\phi_{\text{resp}}) = 2^e = 2^{nf}$).

More specifically, the subgroup $E_A[2^f] = E^{(1)}[2^f]$ is generated by two elements

$$E^{(1)}[2^f] = \langle P^{(1)}, Q^{(1)} \rangle.$$

The kernel of $\phi^{(1)}$ is then specified by b and $s^{(1)}$,

$$\ker(\phi^{(1)}) = \langle P^{(1)} + [s^{(1)}]Q^{(1)} \rangle,$$

where we swap $P^{(1)}$ and $Q^{(1)}$ if $b = 1$ and do not if $b = 0$.

To construct the remaining $n-1$ isogenies, we know (Proposition 2.3.31) that $\langle Q^{(2)} \rangle$, where $Q^{(2)} = \phi^{(1)}(Q^{(1)})$, is the kernel of the dual $\widehat{\phi^{(1)}}$. As the dual has degree 2^f, we know

$Q^{(2)}$ is one of the generators of $E^{(2)}[2^f]$. We find another generator $P^{(2)}$ of $E^{(2)}[2^f]$ using `CompleteBasis` (Algorithm 9). Then the kernel of $\phi^{(2)}$ is specified by $s^{(2)}$, $\ker(\phi^{(2)}) = \langle P^{(2)} + [s^{(2)}]Q^{(2)} \rangle$. This process is continued until we have the full decomposition of ϕ_{resp}.

Now that we have discussed the computation of ϕ_{resp} from `sig`, we can proceed with the attack. We take the role of the adversary and are provided a valid signature `sig` for some public key $\texttt{pk} = E_A$ and message `msg`. We are tasked to find a different public key $\overline{\texttt{pk}} = E_{A'} \neq E_A$ such that $\texttt{Verify}(\overline{\texttt{pk}}, \texttt{msg}, \texttt{sig}) = 1$.

In other words, the goal of the attack is to produce a curve $E_{A'}$ such that the compression described in `sig` constructs an isogeny $\psi_{\text{resp}} : E_{A'} \to E_2$. In general, the codomain of the isogeny described by `sig` from an arbitrary curve $E_{A'}$ will not be E_2.

One method of selecting a candidate curve $E_{A'}$ is to first choose a random 2^e-isogeny $\psi' : E_2 \to E_{A'}$, and then test `sig` for this $E_{A'}$. This is not a good approach, because there are 2^e possible choices for ψ'.

A better method of selecting $E_{A'}$, is to use the intermediate curves we know from the compression of ϕ_{resp} (Equation 6.1). We choose a random 2^f-isogeny $\psi' : E^{(2)} \to E_{A'}$ and then test `sig` for this $E_{A'}$. Again, there are 2^f such possible isogenies, and if f is very large, then the attack is impractical, but this approach is much better than the $2^e = 2^{nf}$ space before.

The authors in [2] implement this attack in Python and experimentally claim that the probability of SQISign being vulnerable to this attack is below 2^{-f}. Because this attack is highly impractical, SQISign could still be considered S-CEO secure, but better attack

avenues may exist, and hence the need for applying a BUFF transform.

6.2.2 S-DEO

Similar to S-CEO we are provided a valid signature $\texttt{sig}$ and public key $\texttt{pk}$ for some message $\texttt{msg}$. We are again tasked with finding a different public key $\overline{\texttt{pk}}$, but also a different message $\overline{\texttt{msg}}$ which are valid for $\texttt{sig}$.

In other words, signing $\texttt{msg}$ for $\texttt{pk}$ should produce the same signature as when signing $\overline{\texttt{msg}}$ for $\overline{\texttt{pk}}$. But recall that in the verification algorithm (Algorithm 3), we must check both $[r]Q' = H(\texttt{msg}, E_1)$ and $[r]Q' = H(\overline{\texttt{msg}}, E_1)$, i.e. a hash collision. This should not happen assuming the hash function is properly chosen. **Thus, SQISign is S-DEO secure.**

6.2.3 MBS

Suppose we are provided with a valid signature $\texttt{sig}$ for public key $\texttt{pk}$ and message $\texttt{msg}$. We are tasked with finding a different public key $\overline{\texttt{pk}}$ and signature $\overline{\texttt{sig}}$ which still verify for $\texttt{msg}$. Referring back to the verification algorithm (Algorithm 3) we need that $[r]Q' = H(\texttt{msg}, E_1)$ and $[r]Q' = H(\overline{\texttt{msg}}, E_1)$, again a hash collision. **Thus, SQISign is MBS-secure.**

6.2.4 wNR

We are given a public key $\texttt{pk}$ and signature $\texttt{sig}$ for some *unknown* message $\texttt{msg}$. We are asked to find different public key $\overline{\texttt{pk}}$ and signature $\overline{\texttt{sig}}$ which still verify with the unknown message $\texttt{msg}$. The outline of the attack is as follows.

The information contained in $\texttt{sig}$ gives us information about constructing $\phi_{\text{resp}} : E_A \to$

E_2 and $\phi_{\text{chall}} : E_1 \to E_2$. For the first part of the new signature $\overline{\texttt{sig}}$, we pick a random 2^f-isogeny $\widehat{\psi^{(n)}} : E_2 \to \widetilde{E^{(n)}}$ such that $\phi_{\text{chall}} \circ \widehat{\psi^{(n)}}$ is cyclic. The cyclic requirement is imposed because otherwise, the verification algorithm would fail, see the description of the Σ-protocol (Section 5.1), more specifically, the verification step (Step 5).

For the rest of the new signature, we continue constructing random 2^f-isogenies $\widehat{\psi^{(j)}} : \widetilde{E^{(j+1)}} \to \widetilde{E^{(j)}}$ for $j = n-1, \ldots, 1$, such that the composition $\widehat{\psi} = \widehat{\psi^{(1)}} \circ \cdots \circ \widehat{\psi^{(n)}}$ is cyclic. We have thus constructed a path from E_2 to some curve $\widetilde{E^{(1)}}$

$$E_2 = \widetilde{E^{(n+1)}} \xrightarrow{\widehat{\psi^{(n)}}} \widetilde{E^{(n)}} \xrightarrow{\widehat{\psi^{(n-1)}}} \ldots \xrightarrow{\widehat{\psi^{(1)}}} \widetilde{E^{(1)}} = E_{A'}. \tag{6.2}$$

Finally, for the new signature $\overline{\texttt{sig}}$ we use the compression of $\widehat{\psi}$, describing $\psi_{\text{resp}} = \psi^{(n)} \circ \cdots \circ \psi^{(1)}$, and the new public key is $\overline{\texttt{pk}} = E_{A'} = \widetilde{E^{(1)}}$.

We have thus presented an attack on the wNR property, although implementation of this attack has not been yet done.

6.2.5 Post-Transformation

After applying the BUFF transform, the attacks on S-CEO and wNR would not be possible, and the scheme would remain S-DEO and MBS secure. In the BUFF transform, specifically the signing algorithm (Algorithm 4), we perform the hash $\texttt{h} = \texttt{H}(\texttt{msg}, \texttt{pk})$ in the signing algorithm, and sign the hash $\texttt{h}$ instead of the message. Later in the verification algorithm (Algorithm 5), we check that the public key and message inputs to the verification algorithm produce the same hash.

In S-CEO and wNR, the attacker is either asked to find a new public key or new signature. Hence for the above attacks to work, we would require a hash collision. The security arguments for S-DEO and MBS remain the same.

6.3 Modifications to SQISign

Results. The results of testing after applying[2] the BUFF transform yielded minimal changes in signing and verification time. The tests were run on the SageMath implementation of SQISign ([54]), using a laptop running Windows with an i7-1065G7 @1.30 GHz and 8 GB of RAM. The hash function in the BUFF transform (Algorithms 4 and 5) was chosen to be `SHA-256` [48].

The test over the finite field $\mathbb{F}_{p_{\text{toy}}}$ yielded an average of 3.57% increase in running time after applying the BUFF transform (6.3). Here,

$$p_{\text{toy}} = 9568331647090687$$

is a $\log_2(p_{\text{toy}}) \approx 53$ bit prime.

The test over the finite field $\mathbb{F}_{p_{6983}}$ yielded an average of 0.47% decrease in running

[2]The modified SageMath code can be found at: `https://github.com/tomadia/SQISign-SageMath-BUFFed`

time after applying the BUFF transform (6.3). Here,

$$p_{6983} = 737430436214997974490748205438634569979446953723240325119999999999999999999$$

is a $\log_2(p_{6983}) \approx 256$ bit prime. This decrease in running time is unexpected because by applying the BUFF transform, we are performing more computations. A possible explanation for this is that SQISign uses probabilistic algorithms. Running more tests most likely would show a slight increase in running time.

Remark 6.3.1.

- The optimized code for SQISign is written in the C language[3]. Implementation of the BUFF transform in C has not yet been done, and is a crucial step for determining the performance change to a real-world application of SQISign.
- The official SQISign specification lists 11 potential different primes for use [14, Section 5.2.3]. We have tested only 2 primes, with one being unsuitable for real-world cryptography.
- By appending a `SHA-256` hash to the signature, we increase the signature size by 256 bits or 32 bytes. One of the features of SQISign is the very small signature sizes. By security level, the signature sizes are: NIST-I (177 bytes), NIST-III (263 bytes), NIST-V (335 bytes) [14, Section 6.1]. A 32 byte increase in signature size is relatively large but acceptable. Other hash functions may provide similar security to `SHA-256` with a smaller increase in signature size.

[3]The code can be found at: `https://github.com/SQISign/the-SQISign`

- There are many probabilistic algorithms in SQISign, such as KeyGenKLPT (Algorithm 7). For an analysis of the failure cases, see [14, Chapter 8]

Run	Original [s]	BUFFed [s]
1	12.815246	13.451124
2	12.500575	13.615909
3	12.812785	13.032886
4	13.227243	13.402713
5	12.953201	13.577151
6	13.321484	13.327887
7	12.822679	12.955670
8	13.069933	14.352298
9	12.716149	13.206865
10	13.313216	13.249065
Average	12.9552511	13.4171568

Table 6.1: SQISign signing and verifying times for $\mathbb{F}_{p_{toy}}$. 10 tests were made (1st column) for both the unmodified code (2nd column) and the modified code with the BUFF transform applied (3rd column).

Remark 6.3.2. In Table 6.1, there was a failure for the BUFFed test on trial 7 (after test 6). We did not include it.

Run	Original [s]	BUFFed [s]
1	642.504599	595.978784
2	606.042313	617.852270
3	610.230417	638.746618
4	630.444214	615.686984
5	617.314850	608.645813
6	620.210331	645.065727
7	619.302924	603.580329
Average	620.8642354	617.9366464

Table 6.2: SQISign signing and verifying times for $\mathbb{F}_{p_{6983}}$. 7 tests were made (1st column) for both the unmodified code (2nd column) and the modified code with the BUFF transform applied (3rd column).

Chapter 7

Conclusions

Post-quantum cryptography may one day be called simply cryptography if quantum computers and quantum algorithms become simple to implement. The process of creating new cryptographic protocols and attempting to break them is standard practice. We believe that one general avenue of further research should be to focus more on cryptanalysis. This is especially true within the subfield of isogeny-based cryptography. There are many variations and different cryptosystems, all relying on the difficulty of the isogeny problem (or some variation). While creating variations of protocols is useful, for example improving efficiency, it misses the underlying point of cryptography, security.

Isogeny-based cryptography, as we have seen, is a relatively new field of research. For all isogeny-based protocols, there is no security reduction to a known NP-hard problem. While a lack of such reduction does not necessarily imply isogeny-based cryptography can be seen as less secure than other NIST proposals, which do have NP-hard reductions, it does decrease the likelihood of adoption.

In this thesis, we implemented the BUFF transform to the SQISign digital signature, and analyzed the performance impact. Our results (Tables 6.2 and 6.1) showed that there is a very low increase in running time; although our use of the SHA-256 hash function increases the signature size by a relatively substantial amount.

We provide some more concrete avenues of research following this thesis.

- Efficient implementation of SQISign: Algorithmic changes, coding techniques, selection of parameter sizes, etc.
- Implementation of the BUFF transform in the current fastest implementation of SQISign.
- Using different hash functions for the BUFF transform, and observing the difference in performance and increase in signature sizes.
- Side-channel attacks for SQISign. For example, see [43] where the authors describe a fault attack and counter-measures. Also, see [39] where the authors work towards a constant-time implementation of SQISign to avoid timing attacks.

Bibliography

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009. ISBN 0521424267.

[2] Thomas Aulbach, Samed Düzlü, Michael Meyer, Patrick Struck, and Maximiliane Weishäupl. Hash your keys before signing: BUFF security of the additional NIST PQC signatures. Cryptology ePrint Archive, Paper 2024/591, 2024. URL `https://eprint.iacr.org/2024/591`.

[3] Jean-Philippe Aumasson, Daniel J. Bernstein, Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. SPHINCS+: Submission to the NIST post-quantum project, 2022. URL `https://sphincs.org/data/sphincs+-r3.1-specification.pdf`.

[4] Andrew Ayer. Duplicate signature key selection attack in Let's Encrypt. Blog post, 2015. URL `https://www.agwa.name/blog/post/duplicate_signature_key_selection_attack_in_lets_encrypt`.

[5] Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Aaron Hutchinson, Amir Jalali, Koray Karabina, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Geovandro Pereira, Joost Renes, Vladimir Soukharev, and David Urbanic. Supersingular isogeny key encapsulation, 2022. URL `https://sike.org/files/SIDH-spec.pdf`.

[6] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: Algorithm specifications and supporting documentation, 2021. URL `https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf`.

[7] Richard Barnes, Jacob Hoffman-Andrews, and James Kasten. Automatic certificate management environment (ACME). Internet-Draft draft-barnes-acme-04, Internet Engineering Task Force, 2015. URL `https://datatracker.ietf.org/doc/draft-barnes-acme/04/`.

[8] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *Advances in Cryptology – ASIACRYPT 2019*, pages 227–247. Springer International Publishing, 2019. ISBN 978-3-030-34578-5.

[9] Jean-François Biasse, David Jao, and Anirudh Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves. In *Progress in Cryptology – INDOCRYPT 2014*, pages 428–442. Springer International Publishing, 2014. ISBN 978-3-319-13039-2.

[10] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In *Advances in Cryptology – EUROCRYPT 2023*, pages 423–447. Springer Nature Switzerland, 2023. ISBN 978-3-031-30589-4.

[11] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. Cryptology ePrint Archive, Paper 2018/383, 2018. URL https://eprint.iacr.org/2018/383.

[12] Wouter Castryck, Ann Dooms, Carlo Emerencia, and Alexander Lemmens. A fusion algorithm for solving the hidden shift problem in finite abelian groups. Cryptology ePrint Archive, Paper 2021/562, 2021. URL https://eprint.iacr.org/2021/562.

[13] Denis X. Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *Journal of Cryptology*, 22(1):93–113, September 2007. ISSN 1432-1378. doi: 10.1007/s00145-007-9002-x. URL http://dx.doi.org/10.1007/s00145-007-9002-x.

[14] Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, Sikhar Patranabix, Christophe Petit, Francisco Rodríguez Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQISign algorithm specifications and supporting documentation, 2023. URL https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/sqisign-spec-web.pdf#cite.AC%3ADKLPW20.

[15] Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, October 2013. ISSN 1862-2976. doi: 10.1515/jmc-2012-0016. URL http://dx.doi.org/10.1515/jmc-2012-0016.

[16] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Paper 2006/291, 2006. URL https://eprint.iacr.org/2006/291.

[17] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. Cryptology ePrint Archive, Paper 2020/1525, 2020. URL https://eprint.iacr.org/2020/1525. Major revision of the published version.

[18] Luca De Feo. Mathematics of isogeny based cryptography. Lecture notes, 2017. URL https://arxiv.org/abs/1711.04062.

[19] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In *Advances in Cryptology – EUROCRYPT 2019*, pages 759–789. Springer International Publishing, 2019. ISBN 978-3-030-17659-4.

[20] Luca De Feo, Jean Kieffer, and Benjamin Smith. Towards practical key exchange from ordinary isogeny graphs. In *Advances in Cryptology – ASIACRYPT 2018*, pages 365–394. Springer International Publishing, 2018. ISBN 978-3-030-03332-3.

[21] Christina Delfs and Steven D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$, 2013. URL https://arxiv.org/abs/1310.7789.

[22] Tom St Denis. *Cryptography for Developers*. Syngress, 2007. ISBN 9781597491044. doi: 10.1016/b978-1-59749-104-4.x5000-6. URL http://dx.doi.org/10.1016/B978-1-59749-104-4.X5000-6.

[23] Max Deuring. Die typen der multiplikatorenringe elliptischer funktionenkörper: G. herglotz zum 60. geburtstag gewidmet. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 14(1):197–272, December 1941. ISSN 1865-8784. doi: 10.1007/bf02940746. URL http://dx.doi.org/10.1007/BF02940746.

[24] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976. ISSN 1557-9654. doi: 10.1109/tit.1976.1055638. URL http://dx.doi.org/10.1109/TIT.1976.1055638.

[25] Morris J. Dworkin and NIST. SHA-3 standard: Permutation-based hash and extendable-output functions, August 2015. URL http://dx.doi.org/10.6028/nist.fips.202.

[26] Luca De Feo. Exploring isogeny graphs. Habilitation manuscript, 2018. URL https://defeo.lu/hdr/#manuscript.

[27] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Cryptology ePrint Archive, Paper 2011/506, 2011. URL https://eprint.iacr.org/2011/506. This paper is an extended version of the published version.

[28] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay

functions from supersingular isogenies and pairings. Cryptology ePrint Archive, Paper 2019/166, 2019. URL https://eprint.iacr.org/2019/166.

[29] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: compact post-quantum signatures from quaternions and isogenies. Cryptology ePrint Archive, Paper 2020/1240, 2020. URL https://eprint.iacr.org/2020/1240.

[30] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in Cryptology—CRYPTO '86*, page 186–194. Springer-Verlag, 1987. ISBN 0387180478.

[31] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON: Fast-fourier lattice-based compact signatures over NTRU, 2022. URL https://falcon-sign.info/falcon.pdf.

[32] Steven D. Galbraith. Constructing isogenies between elliptic curves over finite fields. *LMS Journal of Computation and Mathematics*, 2:118–138, 1999. ISSN 1461-1570. doi: 10.1112/s1461157000000097. URL http://dx.doi.org/10.1112/S1461157000000097.

[33] Steven D. Galbraith and Frederik Vercauteren. Computational problems in supersingular elliptic curve isogenies. Cryptology ePrint Archive, Paper 2017/774, 2017. URL https://eprint.iacr.org/2017/774.

[34] Oded Goldreich. Randomized methods in computation - lecture notes, 2001. URL https://www.wisdom.weizmann.ac.il/~/oded/rnd01.html.

[35] Michael Gora, Eric Simpson, and Patrick Schaumont. Intellectual property protection for embedded sensor nodes. In *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 289–298. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-70550-5.

[36] Noam D. Elkies (https://mathoverflow.net/users/14830/noam-d elkies). Why do we get a connected 2-regular graph? MathOverflow. URL https://mathoverflow.net/q/467743. URL:https://mathoverflow.net/q/467743 (version: 2024-03-26).

[37] Alex Youcis (https://math.stackexchange.com/users/16497/alex youcis). Can a separable isogeny of elliptic curves have an inseparable dual? Mathematics Stack Exchange. URL https://math.stackexchange.com/q/1763893. URL:https://math.stackexchange.com/q/1763893 (version: 2016-04-29).

[38] Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. Seems legit: Automated analysis of subtle attacks on protocols that use signatures. Cryptology ePrint Archive, Paper 2019/779, 2019. URL https://eprint.iacr.org/2019/779.

[39] David Jacquemin, Anisha Mukherjee, Péter Kutas, and Sujoy Sinha Roy. Ready to SQI? safety first! towards a constant-time implementation of isogeny-based signature, SQIsign. Cryptology ePrint Archive, Paper 2023/807, 2023. URL https://eprint.iacr.org/2023/807.

[40] Ernst Kani. The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1997(485):93–122, 1997. ISSN 1435-5345. doi: 10.1515/crll.1997.485.93. URL http://dx.doi.org/10.1515/crll.1997.485.93.

[41] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Jia, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. Lightweight source authentication and path validation. *ACM SIGCOMM Computer Communication Review*, 44(4):271–282, August 2014. ISSN 0146-4833. doi: 10.1145/2740070.2626323. URL http://dx.doi.org/10.1145/2740070.2626323.

[42] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem, 2004. URL https://arxiv.org/abs/quant-ph/0302112.

[43] Jeonghwan Lee, Donghoe Heo, Hyeonhak Kim, Gyusang Kim, Suhri Kim, Heeseok Kim, and Seokhie Hong. *Fault Attack on SQIsign*, page 54–76. Springer Nature Switzerland, 2024. ISBN 9783031627460. doi: 10.1007/978-3-031-62746-0_3. URL http://dx.doi.org/10.1007/978-3-031-62746-0_3.

[44] Christopher Leonardi. A note on the ending elliptic curve in SIDH. Cryptology ePrint Archive, Paper 2020/262, 2020. URL https://eprint.iacr.org/2020/262.

[45] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8 (3):261–277, September 1988. ISSN 1439-6912. doi: 10.1007/bf02126799. URL http://dx.doi.org/10.1007/BF02126799.

[46] Luciano Maino and Chloe Martindale. An attack on SIDH with arbitrary starting

curve. Cryptology ePrint Archive, Paper 2022/1026, 2022. URL https://eprint.iacr.org/2022/1026.

[47] Maruti Ram Pedaprolu Murty. Ramanujan graphs. Survey, 2003. URL https://mast.queensu.ca/~murty/ramanujan.pdf.

[48] National Institute of Standards and Technology. FIPS PUB 180-4 secure hash standard, August 2015. URL http://dx.doi.org/10.6028/NIST.FIPS.180-4.

[49] National Institute of Standards and Technology. Call for additional digital signature schemes for the post-quantum cryptography standardization process, 2022. URL https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf.

[50] Arnold K. Pizer. Ramanujan graphs and Hecke operators. *Bulletin of the American Mathematical Society*, 23:127–137, 1990.

[51] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. ISSN 1557-7317. doi: 10.1145/359340.359342. URL http://dx.doi.org/10.1145/359340.359342.

[52] Damien Robert. Breaking SIDH in polynomial time. In *Advances in Cryptology – EUROCRYPT 2023*, pages 472–503. Springer, 2023. ISBN 978-3-031-30589-4.

[53] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Paper 2006/145, 2006. URL https://eprint.iacr.org/2006/145.

[54] Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Giacomo Pope. Learning to SQI: Implementing SQISign in SageMath. GitHub, 2023. URL https://github.com/LearningToSQI/SQISign-SageMath/.

[55] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. ISSN 1095-7111. doi: 10.1137/s0097539795293172. URL http://dx.doi.org/10.1137/S0097539795293172.

[56] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*. Springer New York, 2009. ISBN 9780387094946. doi: 10.1007/978-0-387-09494-6. URL http://dx.doi.org/10.1007/978-0-387-09494-6.

[57] Anton Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *Advances in Mathematics of Communications*, 4(2):215–235, May 2010. ISSN 1930-5346. doi: 10.3934/amc.2010.4.215. URL http://dx.doi.org/10.3934/amc.2010.4.215.

[58] John Tate. Endomorphisms of abelian varieties over finite fields. *Inventiones Mathematicae*, 2(2):134–144, 1966. ISSN 1432-1297. doi: 10.1007/bf01404549. URL http://dx.doi.org/10.1007/BF01404549.

[59] Edlyn Teske. An elliptic curve trapdoor system. *Journal of Cryptology*, 19(1):115–133, March 2005. ISSN 1432-1378. doi: 10.1007/s00145-004-0328-3. URL http://dx.doi.org/10.1007/s00145-004-0328-3.

[60] David Urbanik and David Jao. SoK: The problem landscape of SIDH. Cryptology

ePrint Archive, Paper 2018/336, 2018. URL https://eprint.iacr.org/2018/336.

[61] John Voight. *Quaternion Algebras*. Springer International Publishing, 2021. ISBN 9783030566944. doi: 10.1007/978-3-030-56694-4. URL http://dx.doi.org/10.1007/978-3-030-56694-4.

[62] John Voight. Quaternion algebras, post-publication version (version 1.0.6u), 2024. URL https://math.dartmouth.edu/~jvoight/quat-book.pdf. The corresponding references for this book might differ from the official publication of the book.

[63] S. William. *Cryptography and Network Security - Principles and Practice, 7th Edition*. Pearson Education India, 2017. ISBN 9789353942564.

Public Precomputed Information for SQISign

Here we list information publicly available to parties participating in the SQISign digital signature scheme. For a complete description on how these values are selected see the SQISign specification document [14].

- A prime $p \equiv 3 \pmod 4$.
- The supersingular elliptic curve $E_0 : y^2 = x^3 + x$.
- The endmorphism ring of E_0 which is isomorphic to $\mathcal{O}_0 = \mathbb{Z} \oplus i\mathbb{Z} \oplus \frac{i+j}{2}\mathbb{Z} \oplus \frac{1+k}{2}\mathbb{Z}$.
- The security parameter λ which satisfies $\log(p) \approx 2\lambda$.
- The odd, smooth value T determining the "accessible torsion", satisfying $T|p^2 - 1$, and $T > p^{5/4}$.
- the largest integer f such that $2^f | p + 1$.
- The largest integer g such that $3^g | T$.

- The degree of the commitment isogeny D_{com}, satisfying $D_{\text{com}}|T$. For the provided parameter set, we define it as $D_{\text{com}} = T/3^g$.
- The degree of the challenge isogeny D_{chall}, satisfying $D_{\text{chall}}|T$ and $\gcd(D_{\text{chall}}, D_{\text{com}}) = 1$. For the provided parameter sets we define it as $D_{\text{chall}} = 2^f 3^g$.
- $B_{0,T} = [P_T, Q_T]$, a basis of $E_0[T]$.
- $B_{0,D_{\text{com}}} = [P_{D_{\text{com}}}, Q_{D_{\text{com}}}]$, a basis of $E_0[D_{\text{com}}]$.
- $B_{0,D_{\text{chall}}} = [P_{D_{\text{chall}}}, Q_{D_{\text{chall}}}]$, a basis of $E_0[D_{\text{chall}}]$.
- The action of $\theta = j + (1+k)/2$ on $B_{0,D_{\text{com}}}$
- The action of $\alpha \in \{i, j, k, \frac{i+j}{2}, \frac{1+k}{2}\}$ on $B_{0,T}$ and $B_{0,D_{\text{chall}}}$.
- A list `extremal_order_list` of alternate p-extremal maximal orders.
- `SQISign_keygen_attempts` is a constant that limits the number of trials for the keygen computation.
- `KLPT_secret_key_prime_size` is a constant that describes the bitsize of the secret key small prime, which should be $\lceil \log(p)/4 \rceil$.
- `KLPT_signing_klpt_length` is a constant equal to a multiple of f.
- `SQISign_response_attempts` is a constant that limits the number of trials for the response phase in the signing algorithm.

Extra Algorithms

Here we list some of the auxiliary algorithms that were referenced throughout the thesis. Note that we only include the input and output descriptions for the algorithms. See the SQISign specification [14] for the full details on all algorithms.

Algorithm 6 IdealToIsogenyEichler$_{l^\bullet}(I, J, B_{A,T}, Q)$

Input: I a left $\mathcal{O}$-ideal of norm l^e, where $e = fg$
Input: J a $(\mathcal{O}_0, \mathcal{O})$-ideal of norm $l^\bullet$
Input: $B_{A,T}$, a basis of $E_A[T]$
Input: Q, a point of $E_A[l^f]$
Output: ϕ_I corresponding to I
Output: **zip** $= (b, s_1, s_2, \ldots, s_g)$ compressed ϕ_I
Output: `found` a boolean indicating that the computation was successful

Algorithm 7 KeyGenKLPT(I)

Input: I a left $\mathcal{O}_0$-ideal of small prime norm (bitsize is `KLPT_secret_key_prime_size` $\approx \log(p)/4$)
Output: Generator β of equivalent ideal $J \sim I$ of norm $2^{\texttt{KLPT\_keygen\_length}}$
Output: `found` a boolean indicating whether a solution was found

Algorithm 8 Normalized(ϕ)

Input: An isogeny $\phi : E_1 \to E_2$
Output: A normalized curve E_2' isomorphic to E_2
Output: An isogeny $\phi' : E_1 \to E_2'$ equal to ϕ up to post-composition by an isomorphism

Algorithm 9 CompleteBasis$_{m,N}(E_{A,B}, R, [x = 1])$

Input: A Montgomery curve $E_{A,B}$ of order N^2, with $m|N$
Input: A point $R = (x_R, y_R \in E_{A,B}(\mathbb{F}_{p^2})$ of order m
Input: A sarting value $x \in \mathbb{F}_{p^2}$, by default $x = 1$
Output: A basis (R,S) of $E_{A,B}[m]$

Algorithm 10 KernelToIsogeny(E,P)

Input: An elliptic curve E
Input: A point $P \in E$
Output: The unique (up to isomorphism) isogney $\phi : E \to E/\langle P \rangle$

Algorithm 11 NormalizedDlog$_{l^f}(E,(R,s),P)$

Input: An elliptic curve E
Input: A basis (R,S) of $E[l^f]$
Input: A point $P \in E[l^f]$
Output: Normalized integers (a,b) such that $[a]R + [b]S = \pm P$ using a Pohlig-Hellman-style algorithm

Algorithm 12 TorsionBasis$_{m,N}(E_{A,B})$

Input: A Montgomery curve $E_{A,B}$ with order N^2, with $m|N$
Output: A basis (R,S) of $E_{A,B}[m]$

Algorithm 13 Decompress$_{\text{resp}}(E,s)$

Input: E a normalized Montgomery curve
Input: $\mathbf{s} = (b_1, s_1, s_2, \ldots, s_e)$ a compression of the isogeny ϕ
Output: E the codomain of ϕ
Output: $Q \in E$ a point generating the 2-isogeny ϕ' such that $\phi = \widehat{\phi'} \circ \phi''$

Algorithm 14 DecompressAndCheck$_{\text{chall}}(E,s,Q,r,\texttt{msg})$

Input: E a normalized Montgomery curve
Input: $\mathbf{s} = (b_1, s_1, b_2, s_2)$ a compression of the isogeny ϕ_{chall}
Input: $Q \in E$ a point of order 2
Input: r an integer
Input: `msg` a message
Output: `verified` a boolean indicating whether verification should pass

Algorithm 15 KernelDecomposedToIdeal$_D(a,b)$

Input: $a,b \in \mathbb{Z}$ defining a point $[a]P_D + [b]P_D$ on E_0 of order D generating an isogeny ϕ
Output: I_ϕ a left $\mathcal{O}_0$-ideal

Algorithm 16 SigningKLPT$_{2^e}(K, I_\tau)$

Input: K a left $\mathcal{O}$-ideal
Input: I_τ a $(\mathcal{O}_0, \mathcal{O})$-ideal of prime norm N_τ, coprime to nrd(K)
Output: Generator β of equivalent ideal $J \sim I$ of norm 2^e
Output: `found` a boolean indicating whther a solution was found