Hochschule RheinMain
Fachbereich Design Informatik Medien
Informatik – Smarte Systeme für Mensch und Technik

Master Thesis

to obtain the academic degree

Master of Science

Performance Evaluation of CSIDH on the Surface

Submitted by	Andreas Hellenbrand
on	24. March 2023
Supervisor	Prof. Dr. Steffen Reith
Co-Supervisor	Dr. Michael Meyer

Abstract

Post Quantum Cryptography (PQC) is of great interest among the cryptographic research community. It is motivated by the need to find new schemes to keep cryptographic applications secure, even after large-scale quantum computers become available. For this reason, the National Institute of Standards and Technology (NIST) is currently running the PQC competition to standardize new PQC schemes. Even though not part of the competition, one promising candidate is the isogeny-based *Commutative Supersingular Isogeny Diffie Hellman* (CSIDH) key exchange scheme. CSIDH offers small keys and can be used as *drop-in* replacement in Diffie-Hellman-based protocol, such as *Signals* end-to-end encryption protocol for mobile messenger. The downside of CSIDH is its rather expensive performance. To this end, Castryck and Decru published *CSIDH on the surface* (CSURF) in 2020, which gives an 5.68% improvement. The same authors later revisited CSURF with some small tweaks to the arithmetic of the scheme.

In this work, we present the first implementation of the updated CSURF using optimized $\mathbb{F}_p$ arithmetic. While the results can not confirm the improvements of 5.68% by Castryck and Decru, we propose new parameters for a 511-bit prime that can tie the performance of a state-of-the-art CSIDH implementation.

Contents

Chapter 1

Introduction

Cryptography is derived from the ancient Greek word for *secret writing*. It is the art of encoding a message, such that even if the message ends up in the *wrong* hands, its content remains unknown. Only the correct recipient should be able to decode the message, also known as *ciphertext*, and gain access to its *plaintext*. This encoding and decoding of the message is called encryption and decryption, respectively, and requires a *secret* shared between the sender and receiver.

An early cryptographic scheme is the *scytale* used in ancient Greece. The syctale uses a rod of a given diameter, where a slice of paper is wound around. After the message is written on the paper and is unwrapped, the text is scrambled. To decode the message, a stick of the same diameter is needed. Such a permutation of the letters of the message is called *transposition cipher*. The Romans used a different method attributed to Julius Caesar. The *Ceaser Cipher* is a *substitution cipher*. Each letter is replaced by the letter next to it in the alphabet, shifted by a fixed number. For example, if the key is *3 steps to the left*, D gets substituted with A, E with B, F with C, and so on, until A gets Y, B gets X, and C gets Z. To decipher the message, the substitution is used in reverse.

Such early schemes were mainly used for military or political communications. While the methods used were constantly updated and improved, for example, the *Vigenère cipher* which uses multiple Ceaser ciphers, this use case did not change until after World War II and the use of the *Enigma* machine. These schemes offer no security by today's standard; for example, the *scytale* and *Ceaser cipher* can be broken by brute-force, e.g., by trying out different keys. The *Enigma* was broken using the polish *bomba kryptologiczna* machine.[1] However, like the early examples, modern encryption schemes like AES still rely on a shared secret key between the sender and receiver. Due to this property, the term *symmetric cryptography* is used

[1] Alan Turning's *Bombe* machine built at Bletchley Park was based on this design.

for such schemes.

In the following decades, the use of cryptography experienced growing usage in other fields. While this can partially be attributed to the increasing availability of computing devices, the invention of *public-key* or *asymmetric cryptography* by Diffie and Hellman in 1976 [38] had the most significant impact. Their protocol allowed the exchange of a private key over an insecure channel, overcoming the major downside of symmetric cryptographic systems. Rivest, Shamir, and Adleman released RSA shortly afterwards in 1978 [61]. RSA allows the encryption of a message towards a public key, which can only be decrypted with a corresponding private key, eliminating the need for a shared key. The scheme can also be used in reverse, where the private key is used on a message so that it can be verified with the corresponding public key. Such an application is called a *digital signature*. Many more public-key schemes were proposed since. For example, ElGamal encryption [41], Schnorr signatures [63], or Petersen commitments [58], allowing for even more cryptographic applications, such as TLS, anonymous authentication [25], Zero Knowledge Proof [13] for online voting [26], and cryptocoins using Blockchains [53].

Such public-key schemes rely on computational trap door functions, where it is easy to compute the function in one direction but infeasible in the other. Such trap doors are the *factorization problem* or the *discrete logarithm problem*.

By now, cryptography plays a vital role in everyones everyday life, even if mostly hidden away from the users. A significant role in boosting cryptography usage can be attributed to the Snowden leaks, which showed the large scale of surveillance capabilities of the NSA and similar actors such as the GCHQ and BND across all areas of the internet. As a response, usage of HTTPS to encrypt web traffic increased massively in the following years [42], primarily thanks to the letsencypt.org[2] project. Messengers like Signal introduced end-to-end encrypted communications on mobile devices [50], removing the hurdle of systems like PGP. Major platforms like WhatsApp followed this direction. Applications like the Tor Browser allow users to browse the web anonymously and allow access to information, where such access is suppressed by the government otherwise.

One can conclude that cryptography enables technology and new applications to enforce and protect the fundamental human right for privacy in the digital world.

1.1 Post Quantum Cryptography

While the aforementioned trap-door functions used in public-key cryptography are assumed to be computationally *hard* on classic computers, Shor showed that they

[2] https://letsencrypt.org/

could be solved efficiently on a quantum computer [64]. An attacker with access to a large-scale quantum computer could break current public-key schemes. Grover's algorithm [44] gives a speedup for symmetric cryptography compared to attacks on traditional computers; however, doubling the key-size restores the desired security level.

Quantum computers of the required size are not known to be available. While that implies that current public-key cryptography is secure for now, it is unknown for how long. By that, it is worth investigating new cryptographic schemes that rely on different problems *now*. The research into *Post-Quantum Cryptography* (PQC) has been of great interest in recent years and is motivated to ensure that these schemes are readily available and well understood when quantum computers reach the required size. Further, if quantum save cryptography is deployed now, data collected and stored by adversaries can not be decrypted when quantum computers become available.

1.1.1 NIST Competition

In 2016 the National Institute of Standards and Technology (NIST) of the United States called for submissions of Post-Quantum schemes for the standardization of *key encapsulation mechanism* (KEM), and digital signature schemes [54]. NIST previously held such competitions to standardize AES and SHA. Researchers submitted 69 proposals in round 1, of which 15 were left in round 3 in 2020 [55]. The submissions can be differentiated into five categories based on the underlying computational hard problem; *hash-based*, *code-based*, *multivariate cryptography*, *lattice-based* and *isogeny-based*. *Rainbow* [39], a multivariate cryptography scheme that made into round 3 was broken in 2022 [7].

In 2022, the first candidates were selected for standardization. The lattice-based CRYSTALS-KYBER [11] is selected as KEM, while CRYSTALS-DILITHIUM [40], FALCON (both lattice-based) and the hash-based SHINCS+ [4] are selected for digital signatures. BIKE (code-based), Classic McEliece [67] (hash-based), HQC (code-based), and SIKE [46] (isogeny-based) were moved into a fourth round for further research [56].

1.1.2 Isogeny-based Cryptography

Of the five categories in the NIST competition, isogeny-based cryptography is the *newest* and was the focus of much research over the last decade. The idea to use isogenies between elliptic curves was first presented by Couveignes in 1997 [31]. However, as it was rejected at *crypto'97*, it was only released in 2006 after a similar scheme was proposed by Rostovtsev and Stolbunov [62]. These schemes are often referred to as *CRS*. CRS has impractical performance drawbacks, so it is not considered a PQC alternative. Jao and De Feo presented *Supersingular Isogeny Diffie Hellman* (SIDH) in 2011 [45], which uses *supersingular* elliptic curves, opposed to the *ordinary* elliptic curves from CRS. SIKE, a KEM version of SIDH, was later accepted in the NIST PQC competition. Castryck and Decru released a paper in 2022 [20] that completely breaks SIDH and its derivations such as SIKE or B-SIDH [28].

In 2018 Castryck et al. improved the performance of CRS by also using supersingular elliptic curves in a scheme called *Commutative Supersingular Diffie Hellman* (CSIDH). Follow-up work improved the performance further, e.g., [6] improved the isogeny computation, and [2] presented a constant-time version called CTIDH. Castryck and Decru presented *CSIDH on the Surface* (CSURF) [18], which is the main topic of this Thesis. As CSIDH and its derivations are quite new, they are not part of the NIST PQC standardization. The attack on SIDH does not apply to CSIDH, due to a different underlying structure, therefore CSIDH is still assumed to be secure.

The great benefit of isogeny-based schemes is their *small* keys, compared to the other PQC schemes. For example, CSIDH-512 has a public key size of just 64 bytes, while CRYSTALS-KYBER requires 1088 bytes and even 2368 bytes for the secret key [11]. However, isogeny computation is relatively expensive, such that isogeny-based schemes can not compete with the performance of other schemes. By this, improving the performance of isogeny computation and isogeny-based schemes is of great research interest.

Supersingular isogenies can also be used to design signature schemes as shown by SeaSign [34], CSI-FiSh [8] or SQISign [36].

1.2 Goals of this Thesis

The aim of this Thesis is a performance evaluation of *CSIDH on the Surface* (CSURF) for the updated algorithms given in [17]. CSURF will be implemented using the C programming language with optimized assembler-based arithmetic. As this is the first implementation of [17] in the literature known to the author and the first implementation of CSURF using optimized arithmetic, the benchmark results are of great interest.

1.3 CSIDH Toy Example

This section gives a high-level intuition of the ideas behind CSIDH. Therefore mathematical details are left out. It is important to point out that this simplification loses the cryptographic properties of CSIDH, and the goal is just a brief introduction to the components.

Assume Alice and Bob each stand in a room, and their only way of communication is to exchange a number once. Their goal is to end up at a shared number. Each room has a circle with a fixed number of points on the floor. Alice and Bob start by picking three random numbers between 0 and 5 and a direction. They walk on the circle in steps of three, five, and seven, depending on their random numbers and direction. For example, in Figure 1.1, Alice picks $(1 \circlearrowleft, 3 \circlearrowright, 3 \circlearrowleft)$ and thus walks one *step* the third point on the left, and proceeds with three *steps* to each fifth point on the right and ends the walk with three steps to each seventh point to the left. Afterwards, Alice and Bob exchange the number of the point they landed on. In the last step, they move to the point they received and repeat the previous walk from the new starting position. As seen in the example, Alice and Bob end up on the same point.

From the example, two core observations can be made. First, the walk on the circle *mixes* well, i.e., each point can be reached with roughly the same small number of steps, and it is equally likely to end up on any possible point. Second, the *action* (here, the walks) is commutative. That is, it does not matter if Alices' path is taken first, followed by Bob's, or if Bob's path is taken first. In both cases, the result is the same.

1.4 Structure of this Thesis

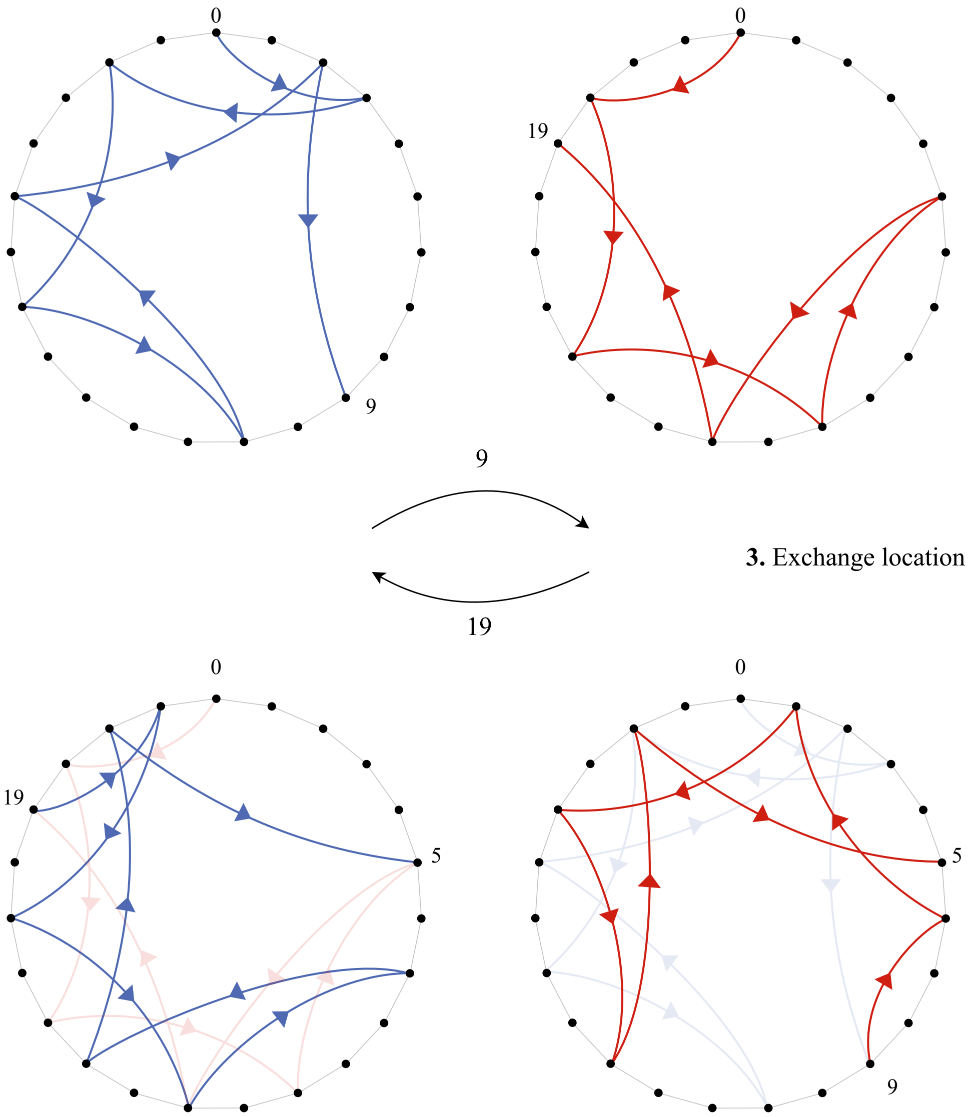
The main topic of this thesis is the performance evaluation of CSURF. To this end, in Chapter 2, we provide the mathematical background of elliptic curves and isogenies. Chapter 3 explains CSIDH and CSURF and their algorithms. The benchmark results for the new implementation produced as part of this work are given in Chapter 4. Finally, we summarize the findings and conclude this thesis in Chapter 5.

1. Alice and Bob select three secret numbers and directions for their number of steps of width 3, 5 and 7.

Alice picks **1** ↻, **3** ↺, **3** ↻

Bob picks **1** ↺, **3** ↺, **2** ↻

2. They both walk their path around the circle, based on their selected numbers and directions



4. They move to the received location and repeat the walk from bevor.

5. Alice and Bob end up on the same location.

Figure 1.1: Simplified toy example for the ideas behind CSIDH.

Chapter 2

Preliminaries

This section presents the mathematical background for the isogenies used in CSIDH and CSURF. We introduce the Diffie-Hellman key exchange and its generalization with Hard Homogeneous Spaces. The following part focuses on elliptic curves and their maps, such as the Frobenius Endomorphism and isogenies, and concludes with the ideal class group.

Notation We use p to denote a prime, q for a prime power $q = p^n$ and $\mathbb{F}_p$ for a finite field (resp. $\mathbb{F}_q$). The symbol ∞ is used for the point at infinity on an elliptic curve E. $\mathcal{O}$ denotes an *order* in an imaginary quadratic field. We use ∞ instead of $\mathcal{O}$ for the point at infinity (as used in most literature) to avoid confusion between these two.

2.1 Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange [38] is one of the first public key schemes introduced in 1976 by Diffie and Hellman.

Let $\mathcal{G}$ be a finite cyclic group of prime order p with generator g. Alice and Bob sample a random secret key a and b from the finite field $\mathbb{F}_p$ and compute their public key as $A = g^a$, and $B = g^b$ respectively. After exchanging their public keys, the shared key can be computed as $S = B^a$ for Alice. Bob computes $S = A^b$. This works as $S = B^a = (g^b)^a = g^{ab} = (g^a)^b = A^b$.

The problem of finding g^{ab} given g^a and g^b is called *Computational Diffie-Hellman* (CDH). *Decisional Diffie-Hellman* (DDH) is the problem of deciding whether for a triplet (g^a, g^b, g^c), if c is randomly sampled from $\mathbb{Z}_p$ or $c = ab$. Triplets of the from (g^a, g^b, g^{ab}) are called DDH triplets (or DDH tuples).

$$\begin{tikzcd} g \arrow[r, "\cdot^{a}"] \arrow[d, "\cdot^{b}"] & g^{a} \arrow[d, "\cdot^{b}"] \\ g^{b} \arrow[r, "\cdot^{a}"] & g^{ab} \end{tikzcd}$$

Figure 2.1: Diffie-Hellman key exchange. Alice moves to the right with g^a and Bob downwards with g^a to compute the shared key g^{ab}.

2.2 Hard Homogeneous Spaces

In [31] Couveignes introduced Hard Homogeneous Spaces (HHS) as a generalization of discrete logarithm-based schemes such as the Diffie-Hellman key exchange. A homogeneous space H for G is a set H which is acted on by a group G, such that the mapping

$$* : G \times H \to H$$

has the following properties:

- (Compatibility): $g * (g'h) = (gg') * h$ for $g, g' \in G$ and $h \in H$.
- (Identity): $g * h = h$ if and only if g is the identity element.
- (Transitivity): there exists an unique $g \in G$, such that $g * h = h'$ for all $h, h' \in H$.

Homogeneous spaces require the following problems to be easy to compute.

Problem 1 (Group Operation) Given $g_1, g_2 \in G$ compute the inverse g_1^{-1}, the group operation $g_1 g_2$ and decide equality $g_1 = g_2$.

Problem 2 (Random Element) Sample $g \in G$ randomly with uniform probability.

Problem 3 (Membership) Given an element h decide if $h \in H$.

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

Problem 5 (Action) Given $g \in G$ and $h \in H$ compute $g * h$. We might use gh as an alternative shorter notation.

To be considered a hard homogeneous space, Problem 6 and 7 need to be difficult.

Problem 6 (Vectorization) Given $h_1, h_2 \in H$ find $g \in G$ such that $gh_1 = h_2$

Problem 7 (Parallelization) Given $h_1, h_1', h_2 \in H$ with $h_1' = gh_1$ for a $g \in G$ compute $h_2' \in H$ such that $h_2' = gh_2$.

Recalling from Section 2.1, we have the group $\mathcal{G}$ of prime order p where the finite field $\mathbb{F}_p$ acts on $\mathcal{G}$ by $h = g^a$, with $h, g \in \mathcal{G}$ and $a \in \mathbb{F}_p$. Here the discrete logarithm problem becomes the Vectorization problem, while the Computational Diffie-Hellman (CDH) assumption is the Parallelization problem. Therefore, groups in which the discrete logarithm problem is assumed to be hard are a Hard Homogeneous Space (HHS).

Couveignes gives another example for HHS, which is different from the discrete logarithm problem using ordinary elliptic curves $\mathcal{E}$ over finite fields and multiplication with a quadratic field of order $\mathcal{O}$. The action is given by an isogeny defined by the ideal $\mathfrak{a}$ from the class group $cl(\mathcal{O})$ (see Section 2.5). Rostovtsev and Stolbunov [62] independently presented a similar key exchange. In subsequent work, performance improvement of the key exchange where presented [45, 35], which led to CSIDH [21] where supersingular elliptic curves are used.

2.3 Elliptic Curves

Elliptic curves over a finite field are interesting for cryptographic applications, as they establish a group where an equivalent to the discrete logarithm problem exists.

Elliptic curves are most commonly denoted as Weierstrass curves.

Definition 2.1. (Weierstrass Curve) Let E be an elliptic curve and $\mathbb{K}$ a field with algebraic closure $\overline{\mathbb{K}}$.[1] Then E is a Weierstrass curve with $a_1, a_2, a_3, a_4, a_6 \in \overline{\mathbb{K}}$ if E has the following form:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

The set of points are the coordinates $(x, y) \in \overline{\mathbb{K}}$ where the equation holds, joint with the point of infinity ∞.

$$E = \{(x, y) \in \overline{\mathbb{K}}^2 | y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\infty\}$$

[1]For a finite field $\mathbb{F}_q$ of prime power order q, the algebraic closure is the union of all extension fields $\mathbb{F}_{q^n}$ with $n \in \mathbb{N}$. ($\overline{\mathbb{F}}_q = \bigcup_{n\in\mathbb{N}} \mathbb{F}_{q^n}$)

$E(\mathbb{K})$ *limits* the elliptic curve over $\mathbb{K}$ rather then $\overline{\mathbb{K}}$, e.g. $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$ and only the $\mathbb{K}$-rational points $(x, y) \in \mathbb{K}^2$ are considered.

The Hasse Theorem approximates the number of points on an elliptic curve over a finite field and thereby its group order.

Theorem 2.2. (Hasse Theorem [66, Theorem V.1.1]) Let $E(\mathbb{F}_q)$ be an elliptic curve over a finite field $\mathbb{F}_q$, then the group order $\#E(\mathbb{F}_q)$ lies within the range:

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}$$

This can be reformulated to $\#E(\mathbb{F}_q) = q + 1 \pm t$, where $|t| \leq 2\sqrt{q}$ is called the Frobenius trace, as it is related to the Frobenius Endomorphism (see 2.6). This is utilized by the Schoof Algorithm to efficiently compute the group order $\#E(\mathbb{F}_q)$ [66, Algorithm XI.3.1].

In most parts of this thesis, Montgomery curves are used, as they offer efficient x-only arithmetic.

Definition 2.3. (Montgomery Curve) A Montgomery Curve E over $\mathbb{K}$ is an elliptic curve of the form

$$E(\mathbb{K}) : By^2 = x^3 + Ax^2 + x \ ,$$

where $A, B \in \mathbb{K}$ and $B(A^2 - 4) \neq 0$.

The B parameter of Montgomery curves can be seen as a twisting factor. Given two elliptic curve $E_{(A,B)}(\mathbb{F}_q) \cong E_{(A,B')}(\mathbb{F}_q)$ via $(x, y) \to (x, \sqrt{B/B'}y)$, then the curves are isomorphic over $\mathbb{F}_q$ if B/B' is a square. Otherwise $E'_{(A,B')}(\mathbb{F}_q)$ is called a *quadratic twist* and only isomorphic over $\mathbb{F}_{q^2}$ (as $\sqrt{B/B'}$ cannot be in $\mathbb{F}_q$). By this, two curves $E(\mathbb{K})$ and $E'(\mathbb{K})$ are each others twist, if they are isomorphic over the algebraic closure $\overline{\mathbb{K}}$ but not $\mathbb{K}$.

If two curves are isomorphic over $\overline{\mathbb{K}}$ can be easily determined with the *j-invariant* as all elliptic curves are isomorphic if and only if they have same j-invariant. For Montgomery curves the j-invariant is given by

$$j(E) = \frac{256(A^2 - 3)^3}{A^2 - 4}$$

In this thesis, supersingular elliptic curves are used primarily. A curve $E(\mathbb{F}_q)$ with characteristic p is *supersingular* if and only if its group order is $\#E(\mathbb{F}_q) \equiv 1 \mod p$ [69, 4.6], otherwise it is called *ordinary*.[2]

[2] A more complete definition of supersingular elliptic curves is given in later in this section (Definition 2.8).

2.3.1 Elliptic Curve Group Law

Each elliptic curve $E(\mathbb{K})$ forms a group, with point addition as the group law. This is given as the following properties hold:

- (Closure) For points $P, Q \in E(\mathbb{K})$, $P + Q = R \in E(\mathbb{K})$
- (Associativity) For points $P, Q, R \in E(\mathbb{K})$, $(P + Q) + R = P + (Q + R)$
- (Neutral Element) For any point $P \in E(\mathbb{K})$, $P + \infty = P \in E(\mathbb{K})$, i.e. the point at infinity ∞ is the neutral element.
- (Inverse) For any point $P \in E(\mathbb{K})$, there is a point $Q \in E(\mathbb{K})$, such that $P + Q = \infty$. Such a Q is also noted as $-P$.

The additive group law can be visualized by the *chord-and-tanget-rule* as shown in Figure 2.2 for an elliptic curve over $\mathbb{R}$. This graphic representation shows that for $P, Q \in E$, $P + Q = Q + P$ holds. By that, the point addition is commutative, and those elliptic curves form an *abelian group*. Addition formulas can be found in [66, III.2.3] or [30] for Montgomery curves.

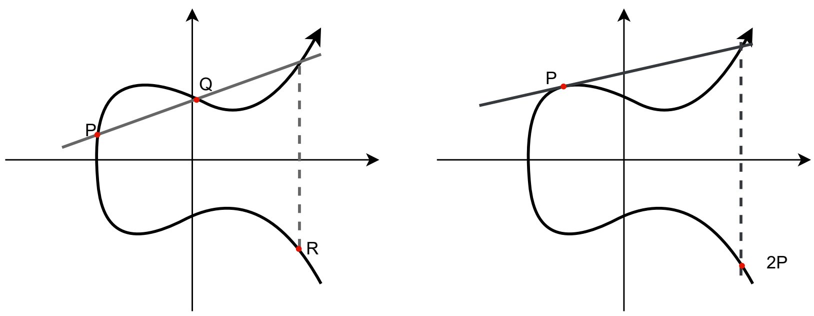


Figure 2.2: Point addition and doubling on an Elliptic Curve over $\mathbb{R}$.

From this, we can repeatedly add a point to itself, leading to scalar multiplication by m ($[m]P = P + P + \cdots + P$, m-times). $[m]P$ for $m < 0$ is given by $[m]P = -[|m|]P$ and for $m = 0$ we use $[0]P = \infty$. The *multiplication-by-m* map $[m] : E \to E$, therefore, denotes a map from a curve to itself. The minimal m that maps $[m]P = \infty$ is called the *order* of P.

It is of interest to look at the points that map to ∞ in the multiplication-by-m map.

Definition 2.4 (Torsion group)**.** Let E be an elliptic curve and $m \in \mathbb{N}$, then $[m]E$ is the set of points over $\mathbb{K}$ that are mapped to ∞ by the scalar multiplication by m.

$$E[m] = \{P \in E \mid [m]P = \infty\}$$

We restrict our definitions to $m \geq 1$, due to the properties of the multiplication by m map for $m \leq 0$ introduced above.

The points in $E[m]$ are the *kernel*[3] of the multiplication-by-m map. As the m-torsion group contains all points that map to ∞ for multiplication-by-m (which also include these of order r, with $r \neq m$ and $r \mid m$), it is different from the set of points of order m.

For elliptic curves $E(\mathbb{F}_q)$ over a finite field, the order r of every point $P \in E(\mathbb{F}_q)$ is finite and divides the group order $r|\#E(\mathbb{F}_q)$ (Lagrange's Theorem [48, §3.3.2]).

2.3.2 Frobenius Endomorphism and the Endomorphism Ring

Other essential maps on elliptic curves are endomorphisms.

Definition 2.5 (Endomorphism)**.** An *endomorphism* is a map φ from an elliptic curve E to itself $\varphi : E \to E$.

As such, the multiplication-by-m gives an endomorphism from $E(\mathbb{K}) \to E(\mathbb{K})$, as the repeated point addition *stays* within $E(\mathbb{K})$. Like isomorphisms that can extend into the algebraic closure $\overline{\mathbb{K}}$, endomorphisms are also not restricted to just $\mathbb{K}$. This allows the definition of the Frobenius Endomorphism.

Definition 2.6 (Frobenius Endomorphism)**.** Given an elliptic curve over a finite field $E(\mathbb{F}_q)$, then the Frobenius Endomorphism π is defined over $\overline{\mathbb{F}}_q$ by the map:

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

If we look at the points in $P \in E(\mathbb{F}_q)$, clearly $\pi(P) = P$ as $x^q \equiv x$ in $\mathbb{F}_q$ by Fermat's Little Theorem. When looking at points from extension fields, this does not hold, such that the Frobenius Endomorphism is indeed different from the identity map. However, for supersingular elliptic curves, the composition $\pi^2 = \pi \circ \pi$ of the Frobenius is $[-p]$, the multiplication by $-p$ (see Appendix A.1).

The observation that $\pi^2 = \pi \circ \pi = [-p]$ implies that $\mathbb{Z}[\pi] := \{[m] + ([n] \circ \pi) | m, n \in \mathbb{Z}\}$ forms a ring with $\mathbb{Z} \subset \mathbb{Z}[\pi]$ as:

[3] The *kernel* of a function $f : E \to E$ is the set of points that are mapped to the point at infinity $ker(f) = \{P \in E \mid f(P) = \infty\}$

- $([m] + [n] \circ \pi) + ([m'] + [n'] \circ \pi) = [m + m'] + [n + n'] \circ \pi$
- $([m] + [n] \circ \pi) \circ ([m'] + [n'] \circ \pi) = [mm' - pnn'] + [mn' + m'n] \circ \pi$

For the parameter choice of CSIDH this ring is known as $\mathbb{F}_p$-rational *endomorphism ring* $\mathrm{End}_{\mathbb{F}_p}(E)$ isomorphic to $\mathbb{Z}[\pi]$. $\mathrm{End}_{\mathbb{F}_p}(E)$ is a subring of $\mathrm{End}(E)$[21].

Definition 2.7 (Endomorphism Ring)**.** Let E be an elliptic curve, then $\mathrm{End}(E)$ is the set of endomorphisms of E. This set forms a *ring* with addition and composition and is called the *Endomorphism Ring.*

Considering elliptic curves over a finite field, the (full) endomorphism ring $\mathrm{End}(E)$ behaves differently for ordinary and supersingular elliptic curves. It is an order (defined below) in an *imaginary quadratic field* for ordinary curves and an order in a *quaternion algebra* if the curve is supersingular[4]. This property defines supersingular (and ordinary) elliptic curves [66, Corollary III.9.4][43, Definiton 9.11.3]. Thus the following statements are equal.

Theorem 2.8 ([43, Theorem 9.11.2])**.** Let $E(\mathbb{F}_q)$ of characteristic p be an supersingular elliptic curve, then:

- $\mathrm{End}(E)$ is an order a in quaterion algebra
- $E[p] = \{\infty\}$
- $\#E(\mathbb{F}_q) \equiv 1 \mod p$

Definition 2.9 (Order [66, III.9])**.** Let $\mathcal{K}$ be a $\mathbb{Q}$-algebra that is finitely generated over $\mathbb{Q}$. An *order* $\mathcal{O}$ of $\mathcal{K}$ is an subring of $\mathcal{K}$ that is finitely generated as a $\mathbb{Z}$-module[5] and satisfies $\mathcal{O} \otimes \mathbb{Q} = \mathcal{K}$.

As mentioned above, CSIDH uses supersingular curves over $\mathbb{F}_p$ where $\mathrm{End}_{\mathbb{F}_p}(E)$ is isomorphic to an order in the *imaginary quadratic field* $\mathrm{End}_{\mathbb{F}_p}(E) \cong \mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ [66, Example III.4.4]. Opposed to $\mathrm{End}(E)$, $\mathrm{End}_{\mathbb{F}_p}(E)$ is commutative.

Further, we let $\mathcal{E}\ell\ell(\mathcal{O}, \pi)$ be the set of all elliptic curves over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$ for Frobenius Endomorphism π. This set has a size of $\#\mathcal{E}\ell\ell(\mathcal{O}, \pi) \approx \sqrt{p}$.

[4] In an imaginary quadratic field a number is composed by two parts, a real and an imaginary part, while in quaternion algebra a number has four parts. While an imaginary quadratic field is commutative, quaternion algebra is not.

[5] A module is a generalization of a *vector space* in which a ring is used for the scalars.

2.4 Isogenies

Another set of maps for elliptic curves are isogenies.

Definition 2.10 (Isogeny)**.** An isogeny is a morphism between two elliptic curves $\varphi : E_1 \to E_2$ with $\varphi(\infty) = \infty$. Elliptic curves connected via an isogeny are called *isogenous*.

For use in cryptography, we focus on *separable* isogenies, for which the degree equals the cardinality of its kernel, $deg(\varphi) = \#ker(\varphi)$. Separable isogenies can be specified by their kernel.

Proposition 2.11. For any finite subgroup $G \subset E_1$, there is a unique elliptic curve E_2 and an isogeny $\varphi : E_1 \to E_2$, such that $ker(\varphi) = G$. The codomain curve is often written as $E_2 = E_1/G$.

We can compute an isogeny of degree m (written as m-isogeny) away from E_1 with a point from the m-torsion group $P \in E_1[m]$ and use Vélus formulas (see Section 2.4.2) to obtain φ and codomain curve $E_2 = E_1/G$ from the subgroup G generated by $G = \langle P \rangle$.

For each isogeny $\varphi : E_1 \to E_2$ there exists a unique *dual isogeny* $\hat{\varphi} : E_2 \to E_1$ with $deg(\varphi) = deg(\hat{\varphi})$. The composition $\hat{\varphi} \circ \varphi$ is the multiplication-by-$deg(\varphi)$ map on E_1 (respectively on E_2 for $\varphi \circ \hat{\varphi}$).

The $\mathbb{F}_q$-isogeny class describes the set of elliptic curves isogenous over $\mathbb{F}_q$. From Tate's isogeny theorem, it follows that two elliptic curves E_1, E_2 over $\mathbb{F}_q$ are isogenous over $\mathbb{F}_q$ if and only if $\#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q)$ [66, Theorem 7.7][43, 25.3].

2.4.1 Isogeny Example

Figure 2.3 shows an example for a 2-isogeny φ from $E_1 : y^2 = x^3 + x$ to $E_2 : y^2 = x^3 - 4x$ over $\mathbb{F}_11$ with $\varphi(x, y) = \left(\frac{x^2+1}{x}, \frac{x^2-1}{x^2}y\right)$. Supersingular elliptic curves over $\mathbb{F}_11$ have 11 $\mathbb{F}_11$-regular points together with the point at infinity, such that $\#E(\mathbb{F}_11) = p + 1$. From Lagrange's theorem, we know that each point order devides 12, and as such, we have points of order 2,3,4 and 6. The red point $K = (0, 0)$ on E_1 is the generator for the subgroup $\mathcal{K} = \langle K \rangle = \{(0, 0), \infty\}$ used as kernel of the isogeny ($ker(\varphi) = \mathcal{K}$). As $\mathcal{K}$ has order 2, the isogeny φ has degree 2. From this, one observes that K on E_1 gets mapped to the point at infinity on E_2 and that if a point is *pushed* through the isogeny, its order is reduced by the degree of the isogeny.

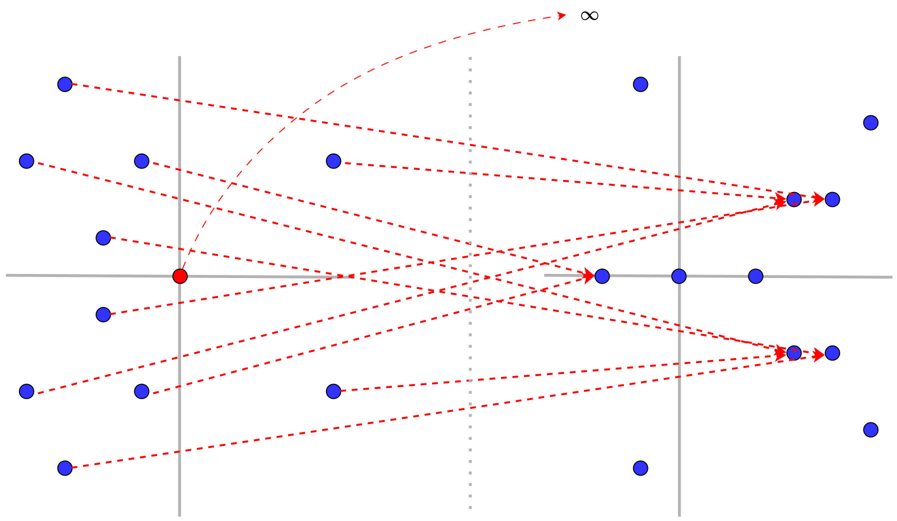


Figure 2.3: Degree 2-isogeny between two elliptic curves over $\mathbb{F}_{11}$ (based on [33]).

2.4.2 Vélu and $\sqrt{}$élu

To compute an isogeny $\varphi : E_1 \to E_2$ with kernel $ker(\varphi) = \langle K \rangle$, one wants to evaluate $\varphi(P)$ for points $P \in E_1$ and obtain the codomain curve E_2, which can be done with Vélu type formulas.

In [68] Vélu gives formulas to compute an isogeny of prime degree ℓ for Weierstrass Curves in runtime $\tilde{O}(\ell)$, where $\tilde{O}$ is a variation of the big-O notation, with logarithmic factors disregarded, to simplify the notation. Costello and Hisil [29] provide formulas to compute odd-degree isogenies on Montgomery curves, which was improved upon in [60] by Renes. Work by Bernstein, De Feo, Leroux, and Smith [6] brings the runtime down to $\tilde{O}(\sqrt{\ell})$ and therefore is called *vélusqrt* (or $\sqrt{}$élu).

The expensive part of these formulas is the evaluation of a polynomial of the form

$$f(X) = \prod_{s \in S} (X - x([s]K)) \ ,$$

where X is the x-coordinate of the point to evaluate, $S = \{1, \ldots, (l-1)/2\}$, K is the kernel generator, and $x([s]K)$ returns the x-coordinate of K added s-times. *Vélusqrt* optimizes the evaluation of this polynomial by replacing S with an *index-system*, where the idea is to split up S such that $S = (I + J) \cup K$, which allows the polynomial to be evaluated separately for each set I, J and K and joined back together at the end. This mechanism is also refered to as *babystep-giantstep*.

2.5 Ideal Class Group

To obtain the structure of a Hard Homogeneous Space, the order $\mathcal{O} \cong \mathbb{Z}[\pi]$ needs to be linked to an action on the set of supersingular elliptic curves over $\mathbb{F}_p$, denoted as $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$. For an order $\mathcal{O}$ of the field $\mathbb{Z}[\sqrt{-p}]$, the *ideal class group* $cl(\mathcal{O})$ is a finite abelian group of invertible principal ideals of $\mathcal{O}$.

Definition 2.12 (Ideal)**.** Let R be a commutative ring. Then an ideal $\mathfrak{I}$ is an additive subring $\mathfrak{I} \subset R$ with

- $\mathfrak{I} \neq \emptyset$
- If $\mathfrak{a}, \mathfrak{b} \in \mathfrak{I}$ then $\mathfrak{a} + \mathfrak{b} \in \mathfrak{I}$
- If $\mathfrak{a} \in \mathfrak{I}$ and $r \in R$ than $\mathfrak{a} * r = r * \mathfrak{a} \in \mathfrak{I}$

An ideal $\mathfrak{I}$ is called *principal ideal* if there exists an element $a \in R$ such that

$$\mathfrak{I} = \langle a \rangle = aR = \{ar \mid r \in R\}$$

The even numbers in $\mathbb{Z}$ with principal ideal $\langle 2 \rangle$ give an simple example for an ideal.

If R is a commutative ring and $\mathfrak{a} \subset R$, we call $\mathfrak{a}$ a *prime ideal* [48, Chapter 15.7], if

$$\alpha, \beta \in R; \alpha * \beta \in \mathfrak{a} \Rightarrow \alpha \in \mathfrak{a} \textit{ or } \beta \in \mathfrak{a}$$

The *norm* ($N()$) of an ideal $\mathfrak{a} \subseteq \mathcal{O}$ can be computed with $N(\mathfrak{a}) = gcd(\{N(\alpha) | \alpha \in \mathfrak{a}\})$.

As an ideal $\mathfrak{a}$ describes endomorphisms on a curve E, it can be used to define a $\mathfrak{a}$-torsion group:

$$E[\mathfrak{a}] = \{P \in E(\overline{\mathbb{F}_q}) | \alpha(P) = \infty \text{ for all } \alpha \in \mathfrak{a}\}$$

The $\mathfrak{a}$-torsion can also be seen as the intersection of the set of kernel points from all elements α of the ideal $\mathfrak{a}$ ($E[\mathfrak{a}] = \bigcap_{\alpha \in \mathfrak{a}} ker(\alpha)$).

From this torsion group $E[\mathfrak{a}]$ on E, a unique isogeny $\varphi_{\mathfrak{a}} : E \to E/E[\mathfrak{a}]$ can be defined, where the degree of $\varphi_{\mathfrak{a}}$ is the norm of the ideal $\mathfrak{a}$. The isogeny can also be written as $\mathfrak{a} * E = E/E[\mathfrak{a}]$, which is called the *class group action* of $cl(\mathcal{O})$ on $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$. The size of the class group is $\#cl(\mathcal{O}) \approx \sqrt{p} \approx \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$ [21].

The *class group action* of $cl(\mathcal{O})$ gives the *action* for our Hard Homogeneous Space. We use $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$ to denote the set of elliptic curves defined over $\mathbb{F}_p$ with an order $\mathcal{O}$

in an imaginary quadratic field with $\mathrm{End}_{\mathbb{F}_p}(E) \cong \mathcal{O}$ and the Frobenius Endomorphism $\pi \in \mathcal{O}$ of E. The action described above is commutative, i.e. for $\mathfrak{a} * E = E_{\mathfrak{a}}$ and $\mathfrak{b} * E = E_{\mathfrak{b}}$, $\mathfrak{ab} * E = \mathfrak{b} * E_{\mathfrak{a}} = \mathfrak{a} * E_{\mathfrak{b}}$.

Remark. This section only gives an introduction to the mathematical background and leaves out some details. For a more in-depth view we refer to [21, 37, 35, 32] and [66] for a more complete introduction to elliptic curves and their maps. Details for Montgomery curves and their arithmetic can be found in [30].

Chapter 3

CSIDH and CSURF

This Chapter describes *Commutative Supersingular Isogeny Diffie-Hellmann* (CSIDH) and *CSIDH on the surface* (CSURF) and outlines their structure, parameters and algorithms.

3.1 CSIDH

Castryck, Lange, Martindale, Panny, and Renes introduced a key exchange scheme called *Commutative Supersingular Isogeny Diffie-Hellmann* (CSIDH) in 2018 [21]. As mentioned above, it is based on work by Couveignes [31], and Rostovtsev and Stolbunov [62] (commonly referred to as CRS) but uses supersingular elliptic curves instead of ordinary curves, making the performance feasible. As Hard Homogeneous Space, CSIDH uses the ideal class group $cl(\mathcal{O})$ and the set of supersingular elliptic curves $\mathcal{E\ell\ell}(\mathcal{O}, \pi)$ over $\mathbb{F}_p$ with a $\mathbb{F}_p$-rational endomorphism ring $\mathrm{End}_{\mathbb{F}_p}(E) = \mathcal{O} \cong \mathbb{Z}[\sqrt{-p}]$ in an imaginary quadratic field and Frobenius Endomorphism $\pi \cong \sqrt{-p}$. Then we have the following action:

$$\begin{aligned} cl(\mathcal{O}) \times \mathcal{E\ell\ell}(\mathcal{O}, \pi) &\to \mathcal{E\ell\ell}(\mathcal{O}, \pi) \\ \mathfrak{a} * E &\to E/E[\mathfrak{a}] \end{aligned}$$

The prime p is chosen to be built with *small* prime factors ℓ_i:

$$p = f \prod_{i=1}^{n} \ell_i - 1$$

By this, all ℓ_i divide the group order of $E(\mathbb{F}_p)$, as supersingular elliptic curves have

$\#E(\mathbb{F}_p) = p+1$ and thereby $E(\mathbb{F}_p)$ has a subgroup of order ℓ_i, which gives an isogeny of degree ℓ_i.

All ℓ_i in $p+1$ split into prime ideals $(\ell_i) = \mathfrak{l}_i\bar{\mathfrak{l}_i} = \langle \ell_i, \pi - 1\rangle\langle \ell_i, \pi+1\rangle$ in $\mathbb{Z}[\pi]$ where the Frobenius Endomorphism is given as $\sqrt{-p}$. The ideal $\mathfrak{l}_i$ gives a torsion subgroup $E[\mathfrak{l}_i] = E[\ell_i]$ with group order ℓ_i, as $ker(\pi - 1) = E(\mathbb{F}_p)$. The torsion subgroup for $\bar{\mathfrak{l}}_i$ is given by

$$E[\bar{\mathfrak{l}}_i] = E[\ell_i] \cap ker(\pi + 1) = \{P \in E[\ell_i] | \pi(P) = -P\}$$

For CSIDH Montgomery curves and $p \equiv 3 \mod 4$ (as in CSIDH) this corresponds to:

$$E[\bar{\mathfrak{l}}_i] = \{(x, y) \in E[\ell_i] \mid x \in \mathbb{F}_p, y \notin \mathbb{F}_p\} \cup \{\infty\}$$

Here, points (x, y) are in $\mathbb{F}_{p^2}$ with $x \in \mathbb{F}_p$ and $y \in \mathbb{F}_{p^2}/\mathbb{F}_p$. However, due to the x-only arithmetic of Montgomery curves, implementations can still work in $\mathbb{F}_p$ only. One says that, if a $\mathfrak{l}_i$-isogeny moves in one direction, then a $\bar{\mathfrak{l}}_i$-isogeny moves in the other; i.e. an isogeny $\varphi : E_1 \to E_2$ with $E_2 = E_1/E_1[\mathfrak{l}_i]$ has a dual isogeny $\hat{\varphi} : E_2 \to E_1$ with $E_1 = E_2/E_2[\bar{\mathfrak{l}}_i]$.

For CSIDH, one wants to evaluate a large degree isogeny corresponding to an ideal $\mathfrak{a}$. As seen in Section 2.4.2, the isogeny computation is quite expensive, so directly sampling $\mathfrak{a}$ is infeasible. However, due to the choice of the prime p we can *build* the ideal as an product of prime ideals $\mathfrak{l}\bar{\mathfrak{l}} = (\ell)$ with $\mathfrak{l}^{-1} = \bar{\mathfrak{l}} \in cl(\mathcal{O})$

$$\mathfrak{a} = \prod_i \mathfrak{l}_i^{e_i}$$

where e_i are small exponents. Rather than sampling such an ideal $\mathfrak{a}$ as a secret key and decomposing it into ℓ_i with exponents e_i [1], CSIDH samples the exponents e_i from a small interval $[-B; B]$ directly. The interval needs to be picked such that the class group is used optimally to give a keyspace of size $(2 * B + 1)^n \approx \#cl(\mathcal{O}) \approx \sqrt{p}$ [21].

All ℓ_i-isogenies for all elliptic curves in $\mathcal{E\ell\ell}(\mathcal{O}, \pi)$ unified form a graph. Figure 3.1 shows the structure of the CSIDH isogeny graph for $\mathcal{E\ell\ell}(\mathcal{O}, \pi)$ over $\mathbb{F}_{419}$ for $3, 5$ and 7-isogenies. The vertices of this graph are the elliptic curves in $\mathcal{E\ell\ell}(\mathcal{O}, \pi)$. Each edge represents two ℓ_i degree isogenies, e.g., for $\mathfrak{l}_i$ in one direction and $\bar{\mathfrak{l}}_i$ in the other. Each ℓ_i-isogeny forms a cycle in $\mathcal{E\ell\ell}(\mathcal{O}, \pi)$, by which, after a certain number of ℓ_i-isogenies, one ends up on the starting point after visiting each elliptic curve in $\mathcal{E\ell\ell}(\mathcal{O}, \pi)$.

This CSIDH isogeny graph differs from the supersingular isogeny graphs used in SIDH; instead, it is similar to a horizontal layer of an isogeny volcano of ordinary elliptic curves.

[1]As the structure of $cl(\mathcal{O})$ is unknown in general, such a decomposition is not possible.

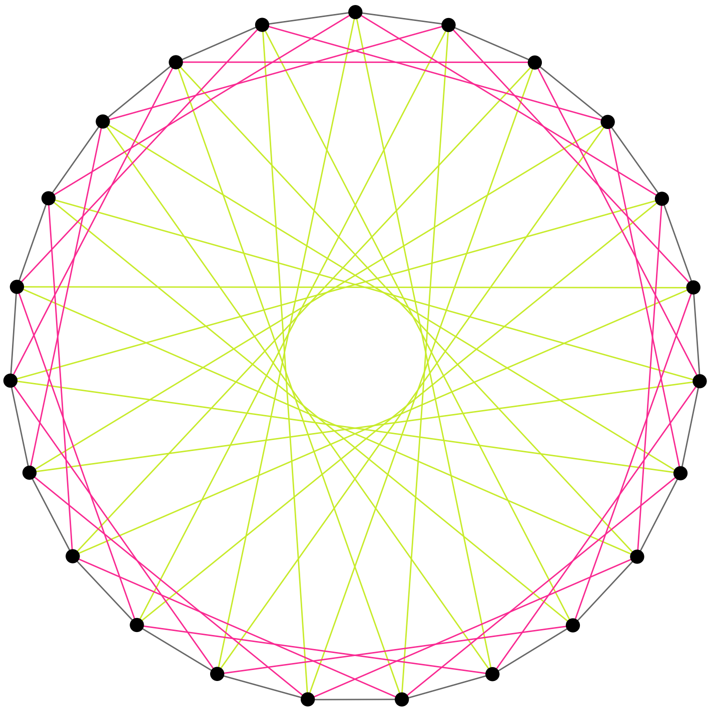


Figure 3.1: CSIDH isogeny graph over $\mathbb{F}_{419}$ with **3**-isogenies, **5**-isogenies, and **7**-isogenies.

Remark. For simplification, the above Section only considers the parameters and structure of CSIDH. The changes implied by the CSURF parameters are introduced in Section 3.2.

To efficiently compute the action [21] proposed the CSIDH-512 parameters where the 511-bit prime p is composed of the 73 first odd primes and 587, such that $p \equiv 3 \mod 4$.

$$p = 4 * \underbrace{(3 * 5 * \cdots * 373)}_{\text{73 first odd primes}} * 587 - 1$$

By that we can compute the action $\mathfrak{a} * E$ as $\mathfrak{a} = \prod \mathfrak{l}_i^{e_i}$ for the 74 different ℓ_i. The private key exponents $e_1, \ldots, e_{74}$ are sampled from the interval $[-5, 5]$, such that $(2 * 5 + 1)^{74} \approx \sqrt{p} \approx \#cl(\mathcal{O}) \approx 2^{256}$.

The CSIDH key exchange scheme can now be given by straightforward adaptation of the Diffie-Hellmann key exchange (Section 2.1). By sampling their exponents, Alice and Bob obtain their secret $\mathfrak{a}$ and $\mathfrak{b}$. They exchange the public keys via

$E_A = \mathfrak{a} * E$ and respectively $E_B = \mathfrak{b} * E$ and conclude by computing the shared key $\mathfrak{a}\mathfrak{b} * E = \mathfrak{a} * E_b = \mathfrak{b} * E_a$, where $E : y^2 = x^3 + x$ is used as the starting curve.

As Montgomery curves, $E_A : y^2 = x^3 + Ax^2 + x$ can be represented with the single parameter A, CSIDH allows for very small public keys (64 bytes in the case of CSIDH-512), as only this parameter needs to be exchanged. Further, CSIDH allows for efficient public key validation, as one just needs to check that A represents a supersingular elliptic curve E_A.

3.1.1 Implementation

This Section outlines the algorithmic approach to compute the CSIDH action. For details we refer to [21] and the optimization in [51].

As given above, the public key is computed by sampling n exponents $e_1, \ldots, e_n \in [-5, 5]$ and walking $|e_i|$ ℓ_i-isogenies adound the isogeny-graph, where the sign of e_i determines the *direction*, e.g., if $\mathfrak{l}_i$ or $\bar{\mathfrak{l}}_i$ is used.

The simplest approach is to search a point generating the kernel for the required isogeny, compute the isogeny, move to the next elliptic curve using Vèlu's formulas, and search the next point until all required isogenies are computed. However, [21] gives a more efficient algorithm where a single point is used for multiple isogeny computations. For simplicity, we start by considering all $e_i > 0$ and generalize later. The idea is to sample a random point $P = (x, y) \in \mathbb{F}_p$ and update it with $P \leftarrow [4]P$ such that the order of P divides $\prod \ell_i$. In a loop, we pick an ℓ_i and compute a kernel point K of order ℓ_i by computing $K = [k/\ell_i]P$ where $k = \prod_j \ell_j$ is the product of yet unprocessed ℓ's. There is a $1/\ell_i$ probability of finding $K = \infty$, in which case one proceeds to the next ℓ_i. After computing the isogeny φ from $\langle K \rangle$ and updating P by pushing it through the isogeny $\varphi(P)$, one repeats the loop with the next ℓ_i. After the loop is completed, a new point P is sampled until all isogenies are computed. To cover $e \in [-B, B]$, one defines two sets S_+, S_-, splitting the ℓ_i's based on the sign of e_i. Instead of sampling a point P, a coordinate $x \in \mathbb{F}_p$ is picked, and the corresponding y^2 is computed via the curve equation. One checks if $y^2 \in \mathbb{F}_p$ via a square-root check and picks the correct set S accordingly. That is S_+ if $y \in \mathbb{F}_p$ and S_- otherwise. After removing the unwanted factors from $P = (x, y)$ by setting $P = [t]P$ where $t = \prod_{i \in S'} \ell_i$, i.e., the product of the primes in the other set S', one proceeds with the loop as described above.

Algorithm 1 shows how to compute the class group action as described above while Algorithm 2 shows how to check if an elliptic curve is supersingular or ordinary, which is required for the key verification.

Algorithm 1: Evaluating the class group action.

Input : $A \in \mathbb{F}_p$ on $E_A : y^2 = x^3 + Ax^2 + x$ and secret exponents $(e_1, \ldots, e_n)$
Output : B on $E_B : y^2 = x^3 + Bx^2 + x$ such that $[\mathfrak{l}_1^{e_1} \ldots \mathfrak{l}_n^{e_n}]E_A = E_B$

while *some* $e_i \neq 0$ **do**
  Sample a random $x \in \mathbb{F}_p$;
  $y^2 \leftarrow x^3 + Ax^2 + x$;
  $P \leftarrow [4](x, y)$;
  if y^2 *is a square in* $\mathbb{F}_p$ **then**
    $s \leftarrow +1$;
  else
    $s \leftarrow -1$;
  end
  $S = \{i \mid e_i \neq 0, sign(e_i) = s\}$;
  if $S = \emptyset$ **then**
    continue;
  end
  $k \leftarrow \prod_{i \in S} \ell_i$;
  $P \leftarrow [(p+1)/k]P$;
  foreach $i \in S$ **do**
    $K \leftarrow [k/\ell_i]P$;
    if $K = \infty$ **then**
      continue;
    end
    Compute isogeny $\varphi : E_A \rightarrow E_B$ with $ker(\varphi) = K$;
    $A \leftarrow B$;
    $P \leftarrow \varphi(P)$;
    $k \leftarrow k/\ell_i$;
    $e_i \leftarrow e_i - s$;
  end
end
return A

Algorithm 2: Verifying supersingarlity

Input : Public key $A \in \mathbb{F}_p$, where $p = 4 * \ell_1 \dots \ell_n - 1$
Output : *supersingular* or *ordinary*

Sample a random $x \in \mathbb{F}_p$;
$y^2 \leftarrow x^3 + Ax^2 + x$;
$P \leftarrow (x, y)$;
$d \leftarrow 1$;
foreach ℓ_i **do**
  $Q_i \leftarrow [(p+1)/\ell_i]P$;
  if $[\ell_i]Q_i \neq \infty$ **then**
    return *ordinary*
  end
  if $Q_i \neq \infty$ **then**
    $d \leftarrow \ell_i * d$;
  end
  if $d > 4\sqrt{p}$ **then**
    return *supersingular*
  end
end

3.1.2 Security

The security of CSIDH relies on a pure isogeny problem. Given two elliptic curves E_1, E_2, find the isogeny $\varphi : E_1 \to E_2$. On classical computers, this can be attacked with a generic *meet-in-the-middle* with complexity $O(\sqrt[4]{p})$, while on a quantum computer, the *claw finding algorithm* of complexity $O(\sqrt[6]{p})$ can be used. However, the additional structure introduced by the class group action allows for a subexponential quantum attack based on an *abelian hidden-shift problem* [21]. [9] and [5] analyzed the quantum attack in more detail. While [21] argues the CSIDH-512 parameters reach NIST level 1, [10], and [59] argue the 511-bit prime does not meet the requirement and needs to be chosen significantly bigger. The exact quantum security of CSIDH remains open for now.

Obviously, CSIDH, as described above, is not *constant-time*, as the number of isogenies to compute depends on the private key. This, and other side-channel and fault injection attacks, are subject of ongoing research [15, 24, 14, 1, 3]. The state-of-the-art constant-time version of CSIDH is CTIDH [2], which uses batching to hide the number of isogeny computations. Recent work [16] exploits the fact that CSIDH uses the elliptic curve $E_0 : y^2 = x^3 + x$ to construct a zero-value and correlation attack.

As stated in Section 1.1.2, SIDH, the basis for the NIST PQC candidate SIKE, was broken by Castryck and Decru [20]. SIDH requires *torsion points* as part of the public key. The attack is based on *glue-and-split* by Kani [47], where two elliptic curves are combined into a Jacobian curve. An efficient *oracle* can be constructed for such a curve that predicts the correct next step in the isogeny-walk based on the torsion points contained in the public key. CSIDH, and derived schemes, are not affected by this attack, as the additional *torsion points* are not required.

3.2 CSURF

CSIDH on the surface (CSURF) was presented in [18] by Castryck and Decru. It is motivated by making 2-isogenies available for the walk through the CSIDH isogeny graph. As degree-2 isogenies can be efficiently computed, this was found to give a performance benefit of 5.68% compared to CSIDH, where only odd-degree isogenies can be used. To use 2-isogenies, CSURF uses supersingular elliptic curves over $\mathbb{F}_p$ with a prime $p \equiv 7 \mod 8$ and $\mathbb{F}_p$-rational endomorphism ring $\mathrm{End}_{\mathbb{F}_p}(E) = \mathbb{Z}[\frac{1+\sqrt{-p}}{2}]$, i.e. supersingular elliptic curves on the *surface*. Section 3.2.1 introduces the terminology of the floor and the surface of an isogeny volcano.

To compute the 2-isogenies, [18] proposed the use of Montgomery$^-$ curves ($E : y^2 = x^3 + Ax^2\text{-}x$) where a single sign is flipped compared to regular Montgomery form. However, after comments by De Feo that imply less efficient scalar multiplication on Montgomery$^-$ curves, Castryck proposed to compute 2-isogenies as a chain on curves of the form $E : y^2 = x^3 + ax^2 + bx$ in [17]. This work focuses on the new approach, and details on the 2-action are given in Section 3.2.2.

3.2.1 The Floor and the Surface

As shown above, Figure 3.1 gives the graph of horizontal ℓ-isogenies in an isogeny volcano, which implies that there exists other horizontal layers, as well as vertical isogenies. These layers are given by their endomorphism rings. Over $\mathbb{F}_p$, with $p \equiv 3 \mod 4$, two endomorphism rings are present in the volcano, such that the following two layers can be defined:

Definition (Floor and Surface [37])**.** Let E be a supersingular elliptic curve over $\mathbb{F}_p$ with $p \equiv 3 \mod 4$ and π the Frobenius Endomorphism with $\pi = \sqrt{-p}$. We say E is *on the floor* if $\mathrm{End}_{\mathbb{F}_p}(E) = \mathbb{Z}[\pi]$ and is *on the surface* if $\mathrm{End}_{\mathbb{F}_p}(E) = \mathbb{Z}[(1+\pi)/2]$.

On the floor the ideals of $\mathbb{Z}(\pi)$ split into $(\ell_i) = \mathfrak{l}_i\overline{\mathfrak{l}_i} = \langle \ell_i, \pi - 1\rangle\langle \ell_i, \pi + 1\rangle$ for all ℓ_i, ideals on the surface $\mathbb{Z}[(1+\pi)/2]$ are of the same form (expect for $\ell_0 = 2$). [17]

$$\ell_i\mathbb{Z}[(1+\pi)/2] = \mathfrak{l}_i\overline{\mathfrak{l}_i} = \langle \ell_i, \sqrt{-p} - 1\rangle \mathfrak{l}_i \langle \ell_i, \sqrt{-p} + 1\rangle$$

We represent the concept of these isogeny volcanos by figures based on [19].[2]

Figure 3.2 shows the floor and surface. For simplification, only a single odd ℓ-isogeny circle is shown. For the case with $p \equiv 3 \mod 8$, as in CSIDH, it can be observed that the surface has, by a factor of 3, fewer elliptic curves in $\mathcal{E}\ell\ell(\mathbb{Z}[(1+\pi/2], \pi)$ than the floor.

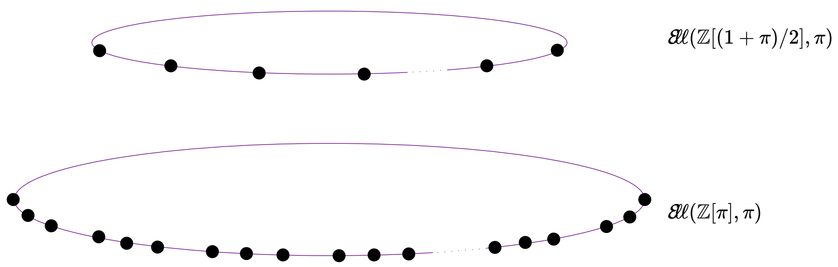


Figure 3.2: Odd ℓ-isogenies in the isogeny volcano with horizontal odd degree isogenies over $\mathbb{F}_p$ with $p \equiv 3 \mod 8$.

Via the following Lemma, the floor and surface are connected by degree-2 isogenies.

Lemma ([37, Lemma 2.2])**.** Let φ be a non-horizontal isogeny between supersingular elliptic curves over $\mathbb{F}_p$. Then the degree of φ is divisible by 2.

As shown in Figure 3.3, each curve on the surface has three vertical 2-isogenies to the floor.

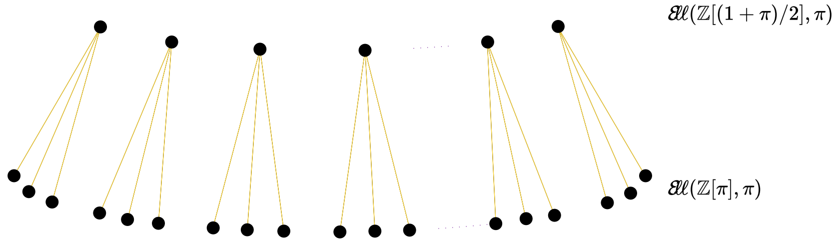


Figure 3.3: Isogeny volcano with vertical 2-isogenies over $\mathbb{F}_p$ with $p \equiv 3 \mod 8$.

[2] Also known as: "How to draw a circus tent" - Savina D.

By this, Figure 3.4 gives the combined isogeny volcano. As the three 2-isogenies are not connected to three *consecutive* curves on the floor[3], the ℓ-isogenies on the floor are displayed as a spiral.

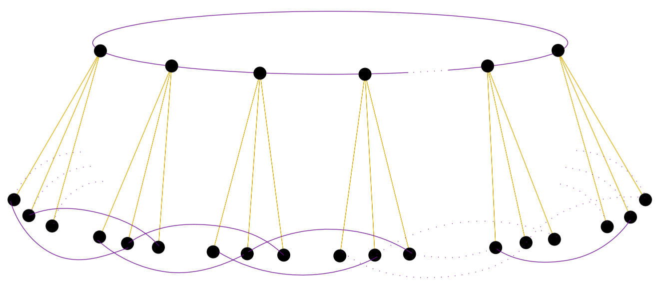


Figure 3.4: Isogeny volcano over $\mathbb{F}_p$ with $p \equiv 3 \mod 8$.

However, CSURF aims to include the 2-isogenies in the horizontal layer. For these we move to curves over $\mathbb{F}_p$with $p \equiv 7 \mod 8$, where $\#\mathcal{E}\ell\ell(\mathbb{Z}[\pi],\pi) = \#\mathcal{E}\ell\ell(\mathbb{Z}[(1+\pi)/2],\pi)$ (Figure 3.5).

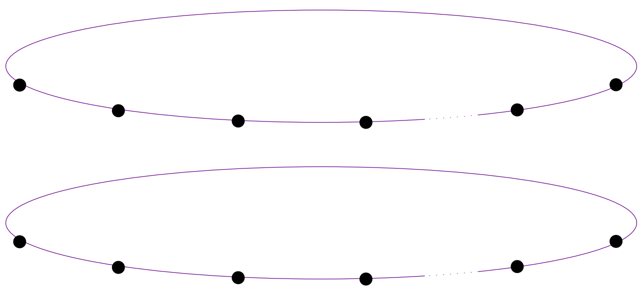


Figure 3.5: Isogeny volcano with horizontal odd degree isogenies over $\mathbb{F}_p$ with $p \equiv 7 \mod 8$.

This moves two of the three 2-isogenies to the surface (Figure 3.6), making 2-isogenies available for CSURF if p is chosen accordingly.

[3]I.e. three curves E_1, E_2, E_3, such that $\varphi_{1,2} : E_1 \to E_2$ and $\varphi_{2,3} : E_2 \to E_3$ both have degree ℓ.

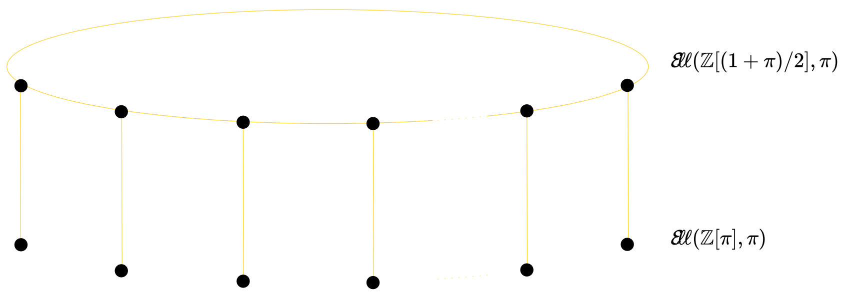


Figure 3.6: Isogeny volcano with 2-isogenies over $\mathbb{F}_p$ with $p \equiv 7 \mod 8$.

3.2.2 Computing 2-Isogenies

In Figure 3.6, one sees that each curve on the surface must have three points of order 2, one for each isogeny *direction*.

Lemma 3.1 ([17])**.** Let $p \equiv 7 \mod 8$ and an elliptic curve $E \in \mathcal{E}\ell\ell(\mathbb{Z}[(1+\pi)/2], \pi)$. Its three points of order 2 can be classified as follows:

1. $P_{\rightarrow}$ whose four halves[4] belong to $E(\mathbb{F}_p)$
2. $P_{\leftarrow}$ whose four halves have $x \in \mathbb{F}_p$ and $y \notin \mathbb{F}_p$
3. $P_{\downarrow}$ whose four halves have $x \notin \mathbb{F}_p$

Furthermore, for $\mathfrak{l}\bar{\mathfrak{l}} = (2)$ we have $E[\mathfrak{l}] = \langle P_{\rightarrow} \rangle$ and $E[\bar{\mathfrak{l}}] = \langle P_{\leftarrow} \rangle$, while $\langle P_{\downarrow} \rangle$ takes us to the floor.

One could use the same approach to find the points $P_{\rightarrow}$ and $P_{\leftarrow}$ as for the other ℓ_i isogeny computations in CSIDH; that is, sampling a random point Q, quotenting out the unwanted factors by $P = \left[\frac{p+1}{\ell_i}\right] Q$ and check whether P has the correct order or $P = \infty$, if valid, compute the isogeny using Vèlu's formulas. This approach is not optimal because there is a 50% chance that $P = \infty$ for $\ell_0 = 2$. However, changing the curve model allows for the computation of *isogeny-chains* and starting from a curve where the points $P_{\rightarrow}$ or $P_{\leftarrow}$ are known, this can be avoided altogether.

[17] uses curves in the form of

$$E : y^2 = x^3 + ax^2 + bx \qquad , P_{\rightarrow} = (0, 0) \ , \tag{3.1}$$

[4]Point halving is the opposite to point doubling; that is, for a point P, we call H in $2H = P$ a half of P.

where the point $P_{\rightarrow}$ is positioned at the origin $(0,0)$, from which it follows that b is a square in $\mathbb{F}_p$ (by Lemma 3.1). For such an elliptic curve E_1, one finds that the isogenous curve $E_2 = [\mathfrak{l}_0]E_1 = E_1/E[\langle P_{\rightarrow}\rangle]$ is given by

$$E_2 : y^2 = x^3 + 2ax^2 + (a^2 - 4b)x$$

where the point $(0,0)$ generates the dual isogeny with $E_1 = [\bar{\mathfrak{l}}_0]E_2 = E_2/\langle P_{\leftarrow}\rangle$ [65, Proposition 3.7 and Chapter 3.5]. The next step is to shift $P_{\rightarrow}$ to the origin $(0,0)$. For this, one needs to find the roots of $x(x^2 + 2ax + (a^2 - 4b))$ which are given by 0 and $(a \pm 2\beta)$ with $\beta = \sqrt{b} \in \mathbb{F}_p$, to find the points $((a \pm 2\beta), 0)$. The points $((a \pm 2\beta), 0)$ correspond to $P_{\rightarrow}$ or $P_{\downarrow}$. Following Lemma 3.1, we can check which point it is by validating if $(a \pm 2\beta) \in \mathbb{F}_p$. As b is a square in $\mathbb{F}_p$, $\beta = \sqrt{b} \in \mathbb{F}_p$, we can shift $P_{\rightarrow} = ((a + 2\beta), 0)$ to $(0,0)$ and obtain:

$$E_2 : y^2 = x^3 + (6a + \beta)x^2 + 4\beta(a + 2\beta)x \qquad P_{\rightarrow} = (0,0)$$

Lemma 3.2 ([17, Lemma 4])**.** Assume $p \equiv 7 \mod 8$ and an elliptic curve $E : y^2 = x^3 + ax^2 + bx$ on the surface such that b is a square and $\beta = \sqrt{b} \in \mathbb{F}_p$. Then $P_{\rightarrow} = (0,0)$ if and only if $a \pm 2\beta$ are both squares in $\mathbb{F}_p$.

Thereby, to compute the 2-isogeny-chain, starting from a curve in form of Equation 3.1, we can interatively compute $\beta = b^{(p+1)/4} = \sqrt{b}$ (see Appendix A.2 for a proof) and update $a \leftarrow (6a + \beta)$, $b \leftarrow 4\beta(a + 2\beta)$. It remains to return the elliptic curve $x^3 + ax^2 + bx$ to Montgomery form $x^x + Ax^2 + x$, such that the other odd ℓ_i-isogenies can be computed, which is done via $A = a/\sqrt{b}$.

So far, this approach only concerns the $\mathfrak{l}_0$ direction for $P_{\rightarrow}$. For the other direction, e.g., $\overline{\mathfrak{l}_0}$, we need to move $P_{\leftarrow}$ to the origin. This can be done utilizing the quadratic twist.

Theorem 3.3. Let E be a Montgomery curve $E : y^2 = x^3 + Ax^2 + x$ where $P_{\rightarrow} = (0,0)$, then it's quadratic twist $E' : y^2 = x^3 - Ax^2 + x$ has $P_{\leftarrow} = (0,0)$.[5]

Proof. As -1 is not a square in $\mathbb{F}_p$, $E : y^2 = x^3 + Ax^2 + x$ has a quadratic twist $-y^2 = x^3 + Ax^2 + x$, which can be reformed to $E' : y^2 = x^3 - Ax^2 + x$ using the substitution $x \leftarrow -x$. If $P_{\rightarrow} = (0,0)$ on E, then $A \pm 2$ are both squares in $\mathbb{F}_p$ (Lemma 3.2). But then $-A \pm 2$ for E' can not be squares, so $P_{\rightarrow}$ is not at $(0,0)$, nor can $P_{\downarrow}$,[6] thereby $P_{\leftarrow}$ must be at the origin. $\square$

[5] $E : y^2 = x^3 + Ax^2 + x$ is in the form of Equation 3.1 with $b = 1$.
[6] As $P_{\downarrow}$ has halves outside $\mathbb{F}_p$.

However, to use the isogeny chain described above, $P_{\rightarrow}$ needs to be at $(0,0)$ rather than $P_{\leftarrow}$, making a shift necessary. To this end, we again need to find $P_{\rightarrow}$ on $E' : y^2 = x^3 - Ax^2 + x$ by finding the roots $0, x_1, x_2$ of $x(x^2 - Ax^2 + x)$, move them to the origin by replacing $x \leftarrow x + x_1$ (resp. $x \leftarrow x + x_2$) and check whether the coefficient b at x is a square in $\mathbb{F}_p$. The other root must be picked if b is not a square. The substitution $x \leftarrow x + x_1$ gives an elltipic curve of form $y^2 = x^3 + ax^2 + b + c$ with $a = 3x_1 - A$, $b = 3x_1^2 - 2Ax_1 + 1$ and $c = x_1^3 + Ax_x^2 + x_1$, i.e. a curve not in the form of 3.1. However, one finds that always $x_1^3 + Ax_1^2 + x_1 = 0$. Therefore the new a and b for $x^3 + ax^2 + bx$ where $P_{\rightarrow} = (0,0)$ are found, and the loop to compute the isogeny chain can be entered. In the last step, we need to twist the curve back and shift $P_{\rightarrow}$ back to the origin by the same means.

As the 2-action requires an elliptic curve E_A where $P_{\rightarrow}$ is at the origin, it is necessary to ensure that for E_A, $A \pm 2$ is both a square in the key verification step.

We conclude this Section with the algorithms for the 2-isogeny-chain.

Algorithm 3: Evaluating the class group action for $\ell_0 = 2$

Input : $A_1 \in \mathbb{F}_p$ on $E_A : y^2 = x^3 + A_1x^2 + x$ and exponent $e_0 \in \{-B, \dots, B\}$

Output : $A_2 \in \mathbb{F}_p$ on $E_2 : y^2 = x^3 + A_2x^2 + x$

1 $a \leftarrow A$;
2 $b \leftarrow 1$;
3 **if** $e_0 < 0$ **then**
4     $a, b \leftarrow twist\_and\_shift(A)$;
5 **end**
6 **for** *1..*$|e_0|$ **do**
7     $\beta \leftarrow b^{p+1/4}$;
8     $b \leftarrow 4\beta(a + 2\beta)$;
9     $a \leftarrow a + 6\beta$;
10 **end**
11 $A \leftarrow a/\sqrt{b}$;
12 **if** $e_0 < 0$ **then**
13     $a, b \leftarrow twist\_and\_shift(A)$;
14     $A \leftarrow a/\sqrt{b}$;
15 **end**
16 **return** A

Algorithm 3 takes the parameter A of the starting curve (resp. the public key of the other party) as well as the private key exponent e_0 and outputs the public key A_2 (resp. the shared key). Lines 1 and 2 set the parameters for the curve to form $y^2 = x^3 + ax^2 + bx$. In lines 3 to 5, the curve is twisted to use $-A$ and then shifted such that $P_{\rightarrow} = (0, 0)$. The loop in lines 6 to 10 computes the isogeny for each e_0. Afterward, the elliptic curve is transformed back to form $y^2 = x^3 + Ax^2 + x$ (line 11) and twisted again if required (lines 12 to 15). Finally, the new elliptic curve is returned via its parameter A.

Algorithm 4: twist and shift

Input : $A \in \mathbb{F}_p$ on $E_1(\mathbb{F}_p) : y^2 = x^3 + Ax^2 + x$

Output : $a, b \in \mathbb{F}_p$ on $E_2(\mathbb{F}_p) : y^2 = x^3 + ax^2 + bx$ where $P_{\rightarrow} = (0, 0)$

```
1  A ← −A;
2  t ← sqrt((A/2)^2 − 1);
3  z ← −A/2 + t;
4  b ← 3z^2 + 2Az + 1;
5  if b is not square then
6  |   z ← −A/2 − t;
7  |   b ← 3z^2 + 2Az + 1;
8  end
9  a ← 3z + A;
10 return a,b
```

Algorithm 4 computes the twist of a curve $E_1(\mathbb{F}_p) : y^2 = x^3 + Ax^2 + x$ and transforms it into form $y^2 = x^3 + ax^2 + bx$ with $P_{\rightarrow} = (0, 0)$. Line 1 *twists* the curve be setting $A \leftarrow -A$. In line 2, a temporary variable is computed to the find a root z (line 3). Line 4 sets the parameter b on $x^3 + ax^2 + bx$. If b is not a square in $\mathbb{F}_p$ (line 5), the other root z needs to be used (line 6). Finally, a on $x^3 + ax^2 + bx$ is computed and returned together with b.

3.2.3 CSURF Parameters

It remains to define the CSURF parameters. [18] suggested the following 513-bit prime p, which is left unchanged in [17]:

$$p = 4 * 3 * \underbrace{(2 * 3 * 5 * \cdots * 389)}_{\substack{\text{75 first consecutive primes,} \\ \text{skip 347 and 359}}} - 1$$

As stated above, the starting curve E_0 is chosen such that $P_{\rightarrow}$ is at the origin

$$E_0 : y^2 = x^3 + 3x^2 + 2x \ ,$$

which has an equivalent Montgomery form

$$E_0 : y^2 = x^3 + (3/\sqrt{2})x^2 + x \ .$$

As now that the additional faster 2-action is available, the range for the private key exponents is adjusted to

$$\underbrace{[-137; 137]}_{e_0} \times \underbrace{[-4; 4]^3}_{e_{1,2,3}} \times \underbrace{[-5; 5]^{46}}_{e_{4\ldots49}} \times \underbrace{[-4; 4]^{25}}_{e_{50\ldots74}}$$

The notation for the keyspace is adapted from [22]. Here, e_0 for the 2-action is sampled from the range $[-137; 137]$ and the exponents e_1, e_2, e_3 for the $3, 5$ and 7 action, as well as the last 25 exponents, from the range $[-4; 4]$. All other 46 exponents are sampled from $[-5; 5]$. This keyspace is choosen such that $(2*137+1)(2*4+1)^{28}(2*5+1)^{46} \approx 2^{256}$. The reduced range $[-4; 4]$ is split between the start and the end to reduce the number of low degree isogenies, as finding a fitting point is more expensive ($1/\ell_i$ chance of missing), and high degree isogenies, which are more expensive to compute ($\tilde{O}(\sqrt{\ell_i})$). In the reference implementation[7] from [18] with Montgomery^{-} curves, these parameter gave an impovement of $\approx 5\%$ over CSIDH.

[7] https://github.com/TDecru/CSURF

New Parameters

In this Thesis, we propose a new parameter set for CSURF, which involves a new 511-bit prime p and adjusted keyspaces. This step is required to overcome slowdowns due to the 513-bit prime in combination with the $\mathbb{F}_p$-arithmetic used in this work. The new prime is of the following form:

$$p = 4 * 53 * \underbrace{(2 * 3 * 5 * \cdots * 401)}_{\substack{\text{74 first consecutive primes,} \\ \text{skip 239, 277, 281 and 367}}} - 1$$

The prime with all factors can be found in Appendix B.1. In the following, we refer to CSURF using this new prime as CSURF-512 and use CSURF-513 for the original prime. The CSURF-512 prime has 74 factors that can be used for isogeny computation, while CSURF-513 offers 75 factors. Sadly, no 511-bit prime with cofactor 4 fulfills $p \equiv 7 \mod 8$ with 74 odd prime factors besides the factor 2. Thereby, as new prime p only has 74 factors available for the isogeny computation, the keyspace needs to be adapted. For the evaluation in this work, we propose five new keyspaces:

CSURF-512(137):

$$\underbrace{[-137; 137]}_{e_0} \times \underbrace{[-4; 4]^3}_{e_{1,2,3}} \times \underbrace{[-5; 5]^{57}}_{e_{4\ldots60}} \times \underbrace{[-4; 4]^{16}}_{e_{61\ldots73}}$$

CSURF-512(75A):

$$\underbrace{[-75; 75]}_{e_0} \times \underbrace{[-4; 4]^3}_{e_{1,2,3}} \times \underbrace{[-5; 5]^{63}}_{e_{4\ldots63}} \times \underbrace{[-4; 4]^{10}}_{e_{64\ldots73}}$$

CSURF-512(75B):

$$\underbrace{[-75; 75]}_{e_0} \times \underbrace{[-3; 3]^2}_{e_{1,2}} \times \underbrace{[-4; 4]^1}_{e_3} \times \underbrace{[-5; 5]^{16}}_{e_{4\ldots19}} \times \underbrace{[-6; 6]^2}_{e_{20,21}} \times \underbrace{[-5; 5]^{42}}_{e_{22\ldots63}} \times \underbrace{[-4; 4]^{10}}_{e_{64\ldots73}}$$

CSURF-512(50A):

$$\underbrace{[-50; 50]}_{e_0} \times \underbrace{[-4; 4]^3}_{e_{1,2,3}} \times \underbrace{[-5; 5]^{62}}_{e_{4\ldots65}} \times \underbrace{[-4; 4]^8}_{e_{66\ldots73}}$$

CSURF-512(50B):

$$\underbrace{[-50; 50]}_{e_0} \times \underbrace{[-4; 4]^3}_{e_{1,2,3}} \times \underbrace{[-5; 5]^{16}}_{e_{4\ldots19}} \times \underbrace{[-6; 6]^4}_{e_{20\ldots23}} \times \underbrace{[-5; 5]^{36}}_{e_{24\ldots61}} \times \underbrace{[-4; 4]^{12}}_{e_{62\ldots73}}$$

While CSURF-512(137) retains the original range of $[-137; 137]$ for the 2-isogenies, CSURF-512(75A) and CSURF-512(50A) reduce this range accordingly. As we have one less high degree isogeny available, the range $[-4; 4]$ can only be used for fewer isogenies while holding the desired key size of 2^{256}. This motivates the introduction of CSURF-512(50B), where the range $[-6; 6]$ is introduced for the 20th to 24th isogenies, such that the range $[-4; 4]$ can be used for the last 12 expensive high-degree isogenies rather than only last 8. For CSURF-512(75B), the usage of $[-6; 6]$ at the 20th and 21st isogeny allows for the range $[-3; 3]$ for the 3- and 5-isogeny, where there is a higher probability of $1/\ell_i$ that the point is not of the required order. Similar keyspace optimizations were proposed in [57] for a faster constant-time algorithm form CSIDH.

Chapter 4

Implementation and Evaluation

In this work, CSURF was implemented in the C language, based on the CSIDH version of the CTIDH implementation from `http://ctidh.isogeny.org/software.html`, which in turn is based on the reference implementation of CSIDH [21] with improvements from [51] and [6] to use $\sqrt{\text{é}}$lu for the isogeny computation. The core of this version is its assembler based $\mathbb{F}_p$-arithmetic.

The code written in this Thesis is released into the public domain and can be found at:

`https://github.com/andhell/CSURF`

The software can be compiled on Intel and AMD CPUs of generation Broadwell and Zen or newer, where the `adcx/adox` instruction is available. The newly implemented algorithms for the 2-action are allready given in Algorithm 3 and 4 above, such that we refer to Section 3.2.2 for details.

In this chapter, we analyze the cost of the algorithms from Section 4. We use **a, M, S, I, Sqrt** to denote the number of additions, multiplications, quarings, inversions and square root computations.

$\mathbb{F}_p$ inversion and square root computation use a *square-and-multiply* algorithm. The cost of one $\mathbb{F}_p$ inversion is $100\mathbf{M} + 505\mathbf{S}$ for the CSURF-512 prime, while square root computation needs $100\mathbf{M} + 504\mathbf{S}$. The isogeny computation for odd degree isogenies with $\sqrt{\text{é}}$lu can be optimized by precomputing the number of baby and giant steps needed for each ℓ_i, which can be calculated based on the number of multiplications needed or the number of CPU cycles. Optimizing for multiplications can be done platform-independent, while the result of cycle-based optimizations depends on the microarchitecture.

4.1 2-Action Cost

Given the implementation of the algorithm described above (Algorithm 3), the 2-action has setup cost of $1\mathbf{M} + 1\mathbf{I} + 1\mathbf{Sqrt}$ computation, while the cost for each 2-isogeny of the isogeny chain is $5\mathbf{a} + 1\mathbf{M} + 1\mathbf{Sqrt}$. However, as for $e_0 < 0$, the curve needs to be twisted and $P_{\to}$ shifted to the origin, Algorithm 4 gives additional setup cost of $7\mathbf{a} + 6\mathbf{M} + 2\mathbf{S} + 3\mathbf{I} + 3\mathbf{Sqrt}$ or $10\mathbf{a} + 9\mathbf{M} + 3\mathbf{S} + 3\mathbf{I} + 3\mathbf{Sqrt}$ if the first guess for the root was wrong.[1]

For the CSURF parameter sets, that gives an average cost of $34600\mathbf{S}$ for CSURF-513 and CSURF-512(137) with a maximal of 137 2-isogenies, $18900\mathbf{S}$ for CSURF-512(75) and $12600\mathbf{S}$ for CSURF-512(50) respectively, due to the cost of the **Sqrt** per 2-isogeny. The cost for additions and multiplications are negligible in this context.

The 2-isogeny chain with Montgomery^{-} has setup cost of $9\mathbf{a}+5\mathbf{M}+2\mathbf{S}+2\mathbf{I}+3\mathbf{Sqrt}$ for $e_0 > 0$ with the addition of $2\mathbf{a}$ for $e_0 < 0$ based on the algorithm given in [18]. Thereby the new approach is much faster if $e_0 > 0$ and comparable otherwise. Cost per 2-isogeny are about the same, with Montgomery^{-} having costs of $4\mathbf{a}+1\mathbf{M}+1\mathbf{S}+1\mathbf{Sqrt}$. By this, each 2-isogeny needs one addition less but one squaring more.

Table 4.1 summerizes aboves findings.

	Setup	**2-isogeny**
$x^3 + ax^2 + bx$	$1\mathbf{M} + 1\mathbf{I} + 1\mathbf{Sqrt}$ to $10\mathbf{a}+9\mathbf{M}+3\mathbf{S}+3\mathbf{I}+3\mathbf{Sqrt}$	$5\mathbf{a} + 1\mathbf{M} + 1\mathbf{Sqrt}$
Montgomery^{-}	$9\mathbf{a} + 5\mathbf{M} + 2\mathbf{S} + 2\mathbf{I} + 3\mathbf{Sqrt}$ to $11\mathbf{a} + 5\mathbf{M} + 2\mathbf{S} + 2\mathbf{I} + 3\mathbf{Sqrt}$	$4\mathbf{a} + 1\mathbf{M} + 1\mathbf{S} + 1\mathbf{Sqrt}$

Table 4.1: 2-isogeny-chain cost.

As public key verification in CSURF requires checking that the curve is on the surface with $A \pm 2$ is a square in $\mathbb{F}_p$, verification has extra cost of $2\mathbf{a} + 2\mathbf{Sqrt}$ over CSIDH.

4.2 Benchmark Results

To compare the different parameter sets from Section 3.2.3, experiments are run on a dual socket system with AMD EPYC 7643 48-Core CPUs with 20.000 runs, each of 65 keys, totaling 1.300.000 experiments. Further, we include the results for CSIDH-512 and CTIDH-512 and the original CSURF parameters for the 513-bit

[1] While Algorithm 3 indicates 4 multiplications per 2-isogenies, this can be reduced to 3 additions and one multiplication, as all multiplications of β are with a small factor and thous can be computed by additions only.

prime. Besides the number of CPU cycles, addition, multiplication, and squarings were counted. Combo is a weighted sum where multiplications are accounted for with 100%, squarings with 105%, and additions with 15%, to reflect the different costs of each $\mathbb{F}_p$ operation. We include results for multiplication-tuned and untuned runs in Table 4.2, and 4.3 as well as Figure 4.2, 4.1, 4.4, and 4.3. Each data entry is the arverge over all 1.300.000 experiments together with the mean squared error (MSE). Table 4.4 gives the benchmark results for public key verification. Here, we only include the multiplication-tuned run, as the result is the same as for the untuned benchmarks. The complete benchmark data can be found in Appendix B.2. We do not inlcude the cycle-tuned results mentioned above, to allow better platform independent comparability.

Tuning the multiplications for $\sqrt{\text{é}}$lu gives a small advantage of 0.32% in Mcyc and 0.75% in *combo* over all parameter sets. However, CSIDH-512 and CSURF-512(137) show higher costs in the tuned run, which can result from the variable time constraints of these implementations. Excluding these results, the gain rises to 0.95% and 1.36%.

The results for CSURF-513 parameters allow no comparison regarding Mcyc, as the $\mathbb{F}_p$ arithmetic for the 513-bit prime requires nine 64-bit blocks versus eight in CSIDH-512. This leads to an increase in the computational cost of $\approx 69\%$ per multiplication and $\approx 6\%$ per addition, which gives an overall increase of $\approx 36\%$ in Mcyc for CSURF-513 over CSIDH-512. However, in terms of cost per $\mathbb{F}_p$ arithmetic, CSURF-513 ties CSIDH. One observes a shift between additions and squarings (133382 ± 20113 vs. 106178 ± 9550 squarings and 394807 ± 33988 vs. 440878 ± 36632 additions) due to each CSURF 2-isogeny square root computation having costs of 505 squarings.

The above Section (4.1) predicted additional squarings of 34600 for parameter set with a maximum of 137 2-isogenies, 18900 for a max range of 74, and 12600 for a maximum of 50 2-isogenies. Given the baseline of 108306 squarings in CSIDH-512 (multiplication-tuned), the benchmark results confirm these numbers. For CSURF-513 and CSURF-512(137), we have 34406 and 31344 additional squarings. The CSURF-512(75) parameters give a difference of 13801 (75A) and 14236 (75B), and CSURF-512(50) ends up with only 7725 (50A) and 9506 (50B) additional squarings. That these numbers are lower is expected, as the adjusted keyspace still saves isogeny computation of a higher degree and thereby also saves some squaring computations compared to CSIDH-512.

Moving to the new 511-bit prime for CSURF-512 lifts the cost per $\mathbb{F}_p$ computation to the same level as CSIDH-512. However, retaining the range $[-137; 137]$ in CSURF-512(137) for the 2-isogenies increases the cost in terms of $\mathbb{F}_p$ operations over CSURF-513 by $\approx 10000\mathbf{M}$ and $\approx 15000\mathbf{a}$, as only 73 odd factors are available, and therefore only 19 secret exponents can be sampled from $[-4; 4]$ vs. 28 in CSURF-513 with 74 odd factors. While reducing the number of 2-isogenies to the range $[-75; 75]$ in CSURF-512(75A) (resp. $[-50; 50]$ in CSURF-512(50A)) requires even more isogeny computations of higher degree, it brings down the number of squaring computations by 20605 (resp. 26681) for the tuned and 14710 (resp. 19236) for the untuned run. These parameters end up slightly faster compared to CSIDH-512 in the multiplication-tuned-run. In 75B and 50B, the number of high-degree isogenies with the reduced range of $[-4; 4]$ is increased by using $[-6; 6]$ for some lower-degree isogenies. However, the benchmark results indicate no benefit over their CSURF-512(75A) and CSURF-512(50A) counterparts, as both are within ≈ 1000 $\mathbb{F}_p$-operations via *combo* (resp. ≈ 4000 for the untuned-run), which is negligible.

CSURF-512(50A) is slightly faster than CSURF-512(75A) with 0.3 Mcyc less and 0.71% better *combo* results in the multiplication-tuned-run. The untuned run is faster by 1.74 Mcyc and 1.25% in *combo*. Compared to CSIDH-512, CSURF-512(50A) is faster by 2.32 Mcyc (3.09%) and 2.68% by instructions (resp. 0.74 Mcyc (0.99%) and 0.89% for the untuned run). The multiplication-tuned CSURF-512(75A) results only gain 2.02 Mcyc (2, 66%) and 1.99% by instructions. These numbers are quite small compared to the margin of error between 8-9% of these variable-time runs, such that it can only be concluded that the new parameters can tie the performance of CSIDH-512 and, at most offer a slight benefit. However, the parameters for CSURF-512(50A) look most promising, such that we recommend these for new implementations where the usage of a 511-bit prime is beneficial.

Public key verification is slightly more expensive for CSURF, as stated above in Section 4.1. The two additional square root computations cost 505 squarings each, which is reflected in the results as the increased number of $\mathbb{F}_p$ operations (17271 vs. 18388). The extra operations require an additional 0.22 Mcyc for the CSURF-512 parameters.

	Mcyc	combo
CSIDH-512	75.07±6.77	497247±41198
CTIDH-512	79.12±3.46	516167±12564
CSURF-513	103.08±9.78	497382±43540
CSURF-512(137)	76.52±7.52	511501±47709
CSURF-512(75A)	73.05±5.96	487358±35786
CSURF-512(75B)	72.93±5.60	488334±34824
CSURF-512(50A)	72.75±6.28	483922±39014
CSURF-512(50B)	73.06±7.01	482305±43746

Table 4.2: Average cost and mean squared error for 1.300.000 multiplication-tuned runs.

	Mcyc	combo
CSIDH-512	74.14±6.46	493063±40900
CTIDH-512	80.40±3.48	527304±12544
CSURF-513	101.01±9.19	491724±42919
CSURF-512(137)	74.88±6.13	505190±39147
CSURF-512(75A)	75.14±6.09	494899±36119
CSURF-512(75B)	75.04±6.44	498616±39407
CSURF-512(50A)	73.40±6.46	488699±39260
CSURF-512(50B)	73.55±5.84	494859±37456

Table 4.3: Average benchmark results for 1.300.000 untuned runs.

	Mcyc	combo
CTIDH-512	2.62±0.11	17022±220
CSURF-513	3.87±0.18	18208±220
CSURF-512(137)	2.83±0.09	18388±38
CSURF-512(75A)	2.85±0.11	18388±38
CSURF-512(75B)	2.84±0.09	18388±38
CSURF-512(50A)	2.85±0.09	18388±38
CSURF-512(50B)	2.86±0.10	18388±38

Table 4.4: Average public key verification benchmark results for 1.300.000 untuned runs.

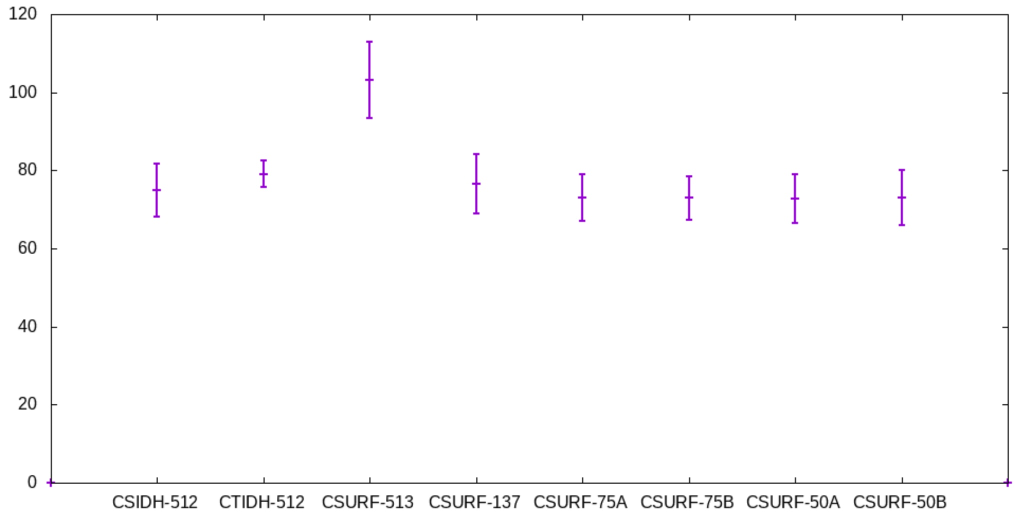


Figure 4.1: Average Mcyc cost of 1.300.000 tested keys (multiplication-tuned).

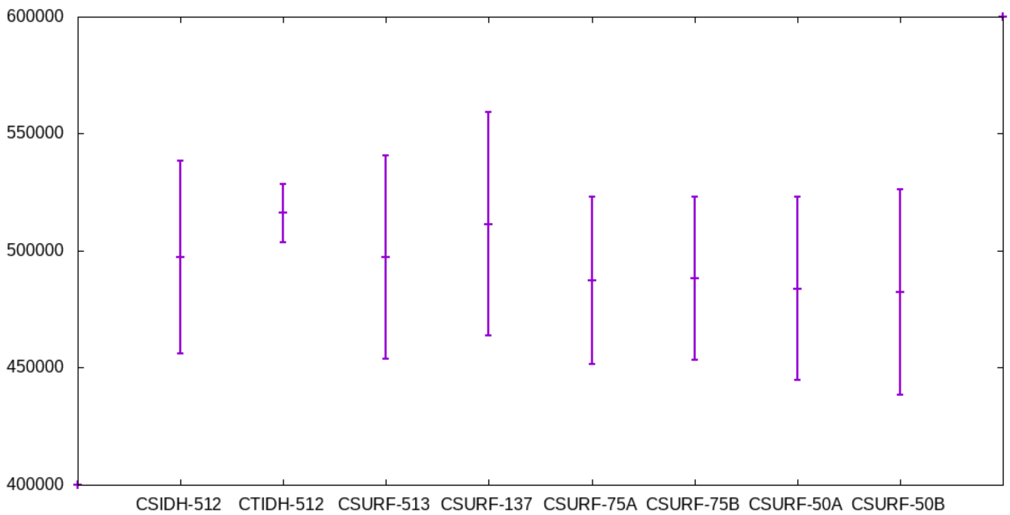


Figure 4.2: Average instruction cost of 1.300.000 tested keys (multiplication-tuned).

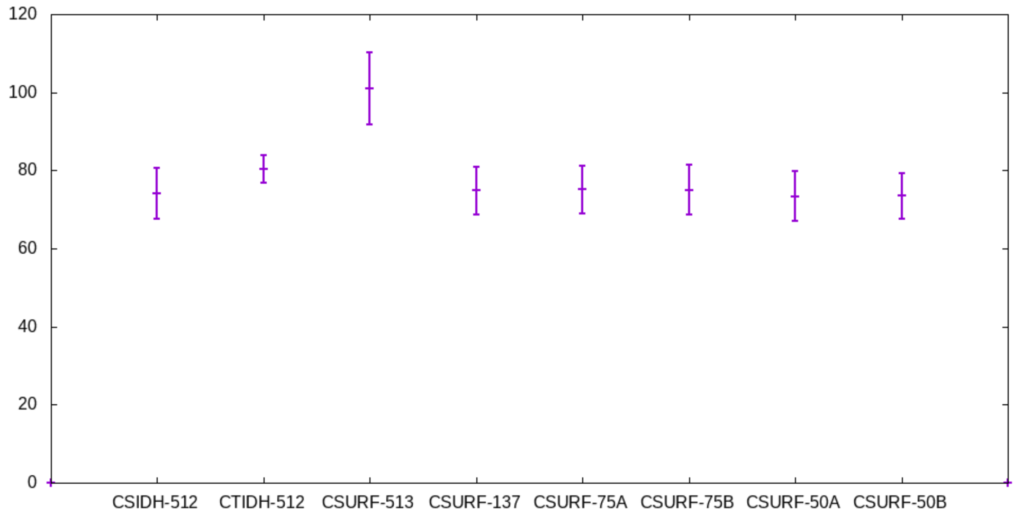


Figure 4.3: Average Mcyc cost of 1.300.000 tested keys (untuned).

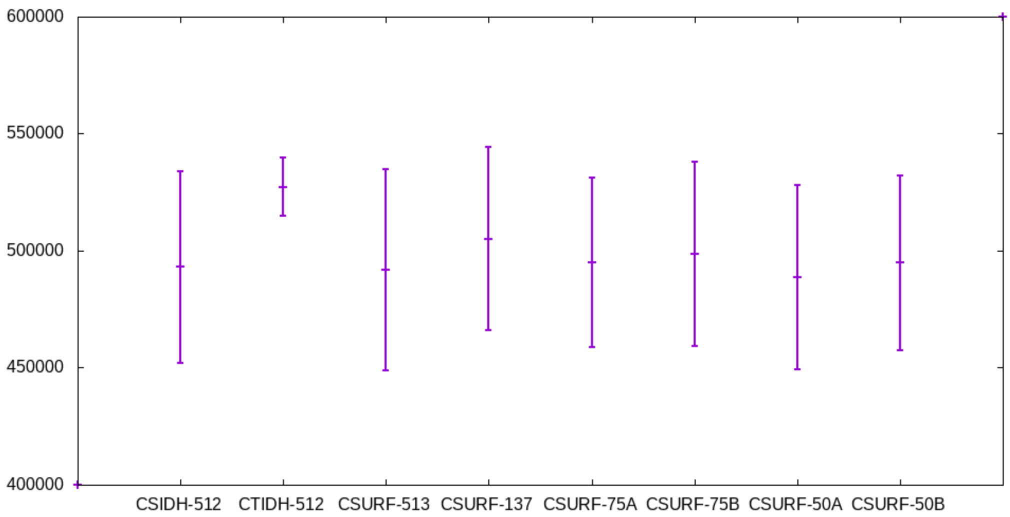


Figure 4.4: Average instruction cost of 1.300.000 tested keys (untuned).

4.2.1 Comparsion with Reference Implementation

The reference implementation from [18] using the Magma algebra system [12] can be found at https://github.com/TDecru/CSURF. This implementation gave a 5.68% improvement of CSURF-513 over CSIDH-512 for 25.000 tested keys. An explanation, comparing the different types of computing 2-isogeny-chains, would not explain the results. The use of elliptic curves with form $x^3 + ax^2 + bx$ over Montgomery$^-$ has negligible cost impact as outlined in Section 4.1. Rather, the difference in the results observed in this work could be explained as both systems use different $\mathbb{F}_p$-arithmetic. Using the assembler optimized $\mathbb{F}_p$-arithmetic gave a $\approx 36\%$ performance penalty of CSURF-513 over CSIDH-512 due to the 513-bit prime, which apparently is not an issue using Magma. Further, in this work, the $\mathbb{F}_p$ instructions are fine-tuned to use the least amount of assembler instructions for the specific prime, which could explain that the 5.68% improvement could also not be confirmed in *combo*, where the number of $\mathbb{F}_p$ instructions are evaluated. Suppose addition and/or multiplication are significantly more expensive in Magma. In that case, that will *smooth out* the high cost of the square root computations in the 2-isogeny chain observed in the assembler-based implementation, as the impact of the following isogenies is more relevant. On this note, the reference Magma implementation is not using $\sqrt{\text{é}}$lu for isogeny computation, as these improvements were released later. However, the results from the vélusqrt paper confirmed the $\approx 5\%$ improvement for an updated Magma version [6, Figure 5.], which also could be explained with the arguments given before.

Chapter 5

Conclusion

It remains to summarize the contents of this thesis and contextualize the results in relation to further research.

In this work, we introduced the mathematics of elliptic curves with isgoenies, endomorphism rings and *ideal class groups* requiered for CSIDH in Chapter 2. Chapter 3 gives the CSIDH-isogeny graph and explained the algorithms required for the key exchange, before moving to the *surface* with CSURF. After proposing new optimized paramters, we presented bechnmark resultes of the first implementation of *CSIDH on the surface* using elliptic curves of form $x^3 + ax^2 + bx$ instead of Montgomery$^-$ in Chapter 4. This is the implementation of this scheme in the literature known to the author. Further, it is the first implementation using optimized $\mathbb{F}_p$ arithmetic, which allows for better comparability to other isogeny-based schemes, such as CSIDH or CTIDH, as well as an in-depth view into the costs of different parameters, by allowing the analysis of $\mathbb{F}_p$ calculations.

While this work could not confirm the improvements observed in the reference implementation, we could tie the performance of the optimized CSIDH implementation by introducing a new 511-bit prime with adjusted keyspaces. Of which, CSURF-512(50) with the new 511-bit prime and a maximum of 50 2-isogenies vs. the 137 of the orginal parameters, gave the most promising results. However, better parameters may be found in further research.

In Section 3.1.2 we mentioned the *zero-value and correlation* side-channel against SIKE and CSIDH by Campos, Meyer, Reijnders, and Stöttinger [16]. This attack exploits the parameter choice of the CSIDH starting curve $E_0 : y^2 = x^3 + x$. However, as this curve is located on the *floor* and CSURF only uses the *surface*, CSURF offers a natural mitigation against this side channel. Following the attacks against SIDH and SIKE, trust in isogeny-based cryptography might be weakened. However, as CSIDH and CSURF use a strictly different structure that does not need to rely on torsion-points as part of the public key, these schemes are assumed to be secure, even though the exact quantum security is currently debated.

A great benefit of CSIDH based-schemes is that they follow the notion of Hard Homogeneous Spaces (Section 2.2), by which they can be used as *drop in* replacement for the Diffie-Hellman-Key-Exchange. For example, CSIDH based-schemes can be used in Signal Messengers 3-way Diffie-Hellman Key-Exchange (X3DH) [49] in order to make the application post-quantum secure. While such mobile applications can benefit from the small keysizes of just a few bytes, the slow performance of CSIDH is a major hurdle.

To this end, *radical isgoenies* by Castryck et al. [22] suggest promising improvements, as isogeny-chains, similar to the 2-isogenies in CSURF are made available to isogenies up to degree 13. In 2022 [23] this was improved up to isogenies of degree 37. It is worth investigating the performance benefits from *radical isogenies* using optimized implementation to allow for comparison against CSIDH and CTIDH. The CSURF implementation from this work gives a natural entry point for such work.

Constant-time implementation are a requirement if CSIDH-based schemes should be considerd for real-world usage. Initial work to turn CSIDH/CSURF with radical isgoenies into constant-time schemes was presented in 2022 [27] using dummy instructions as in [52]. A combination of CSURF, or radical isogenies respectively, and CTIDH seems promising, if the performance gains translate well into an optimized implementation.

In [17], Castryck concludes with "*Overall, there seems little reason not to work on the surface, but don't expect a dramatic impact.*", which is a statement confirmed and strengthened by the findings of this thesis.

Bibliography

[1] Gora Adj, Jesús-Javier Chi-Domínguez, Víctor Mateu, and Francisco Rodríguez-Henríquez. Faulty isogenies: a new kind of leakage, 2022. URL https://arxiv.org/abs/2202.04896.

[2] Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, Tung Chou, Tanja Lange, Michael Meyer, Benjamin Smith, and Jana Sotáková. CTIDH: faster constant-time CSIDH. volume 2021, pages 351–387, 2021. doi: 10.46586/tches.v2021.i4.351-387. URL https://doi.org/10.46586/tches.v2021.i4.351-387.

[3] Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska. Disorientation faults in CSIDH. Cryptology ePrint Archive, Paper 2022/1202, 2022. URL https://eprint.iacr.org/2022/1202. https://eprint.iacr.org/2022/1202.

[4] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. Sphincs: Practical stateless hash-based signatures. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 368–397, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. ISBN 978-3-662-46800-5.

[5] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 409–441. Springer, 2019. doi: 10.1007/978-3-030-17656-3_15. URL https://doi.org/10.1007/978-3-030-17656-3_15.

[6] Daniel J Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster

computation of isogenies of large prime degree. *Open Book Series*, 4(1):39–55, 2020. URL https://velusqrt.isogeny.org/.

[7] Ward Beullens. Breaking Rainbow Takes a Weekend on a Laptop. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2022. doi: 10.1007/978-3-031-15979-4_16. URL https://doi.org/10.1007/978-3-031-15979-4_16.

[8] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient Isogeny Based Signatures Through Class Group Computations. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2019. doi: 10.1007/978-3-030-34578-5_9. URL https://doi.org/10.1007/978-3-030-34578-5_9.

[9] Jean-François Biasse, Annamaria Iezzi, and Michael J. Jacobson. A note on the security of CSIDH. In Debrup Chakraborty and Tetsu Iwata, editors, *Progress in Cryptology – INDOCRYPT 2018*, pages 153–168, Cham, 2018. Springer International Publishing. ISBN 978-3-030-05378-9.

[10] Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 493–522. Springer, 2020. doi: 10.1007/978-3-030-45724-2_17. URL https://doi.org/10.1007/978-3-030-45724-2_17.

[11] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - Kyber: A cca-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 353–367. IEEE, 2018. doi: 10.1109/EuroSP.2018.00032. URL https://doi.org/10.1109/EuroSP.2018.00032.

[12] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. ISSN 0747-

7171. doi: 10.1006/jsco.1996.0125. URL http://dx.doi.org/10.1006/jsco.1996.0125. Computational algebra and number theory (London, 1993).

[13] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical report, 1997.

[14] Fabio Campos, Matthias J. Kannwischer, Michael Meyer, Hiroshi Onuki, and Marc Stöttinger. Trouble at the CSIDH: Protecting CSIDH with Dummy-Operations Against Fault Injection Attacks. In *2020 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*, pages 57–65, 2020. doi: 10.1109/FDTC51366.2020.00015.

[15] Fabio Campos, Juliane Krämer, and Marcel Müller. Safe-Error Attacks on SIKE and CSIDH. In Lejla Batina, Stjepan Picek, and Mainack Mondal, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 104–125, Cham, 2022. Springer International Publishing. ISBN 978-3-030-95085-9.

[16] Fabio Campos, Michael Meyer, Krijn Reijnders, and Marc Stöttinger. Patient Zero and Patient Six: Zero-Value and Correlation Attacks on CSIDH and SIKE. Cryptology ePrint Archive, Paper 2022/904, 2022. URL https://eprint.iacr.org/2022/904. https://eprint.iacr.org/2022/904.

[17] Wouter Castryck. CSIDH On The Surface (CSURF). Isogeny-based Cryptography School - July 2021, 2021. URL https://homes.esat.kuleuven.be/~wcastryc/summer_school_csurf.pdf.

[18] Wouter Castryck and Thomas Decru. CSIDH on the surface. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, volume 12100 of *Lecture Notes in Computer Science*, pages 111–129. Springer, 2020. doi: 10.1007/978-3-030-44223-1_7. URL https://doi.org/10.1007/978-3-030-44223-1_7.

[19] Wouter Castryck and Thomas Decru. PQCrypto 2020 | CSIDH on the surface (Video), 2020. URL https://www.youtube.com/watch?v=hXyE3qg09gA.

[20] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. URL https://eprint.iacr.org/2022/975. https://eprint.iacr.org/2022/975.

[21] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In Thomas

Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018. doi: 10.1007/978-3-030-03332-3_15. URL https://doi.org/10.1007/978-3-030-03332-3_15.

[22] Wouter Castryck, Thomas Decru, and Frederik Vercauteren. Radical isogenies. *IACR Cryptol. ePrint Arch.*, page 1108, 2020. URL https://eprint.iacr.org/2020/1108.

[23] Wouter Castryck, Thomas Decru, Marc Houben, and Frederik Vercauteren. Horizontal racewalking using radical isogenies. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part II*, volume 13792 of *Lecture Notes in Computer Science*, pages 67–96. Springer, 2022. doi: 10.1007/978-3-031-22966-4_3. URL https://doi.org/10.1007/978-3-031-22966-4_3.

[24] Daniel Cervantes-Vázquez, Mathilde Chenu, Jesús-Javier Chi-Domínguez, Luca De Feo, Francisco Rodríguez-Henríquez, and Benjamin Smith. Stronger and Faster Side-Channel Protections for CSIDH. In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology – LATINCRYPT 2019*, pages 173–193, Cham, 2019. Springer International Publishing. ISBN 978-3-030-30530-7.

[25] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, October 1985. ISSN 0001-0782. doi: 10.1145/4372.4373. URL https://doi.org/10.1145/4372.4373.

[26] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, January 2004. ISSN 1540-7993. doi: 10.1109/MSECP.2004.1264852. URL https://doi.org/10.1109/MSECP.2004.1264852.

[27] Jesús-Javier Chi-Domínguez and Krijn Reijnders. Fully projective radical isogenies in constant-time. In Steven D. Galbraith, editor, *Topics in Cryptology - CT-RSA 2022 - Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1-2, 2022, Proceedings*, volume 13161 of *Lecture Notes in Computer Science*, pages 73–95. Springer, 2022. doi: 10.1007/978-3-030-95312-6_4. URL https://doi.org/10.1007/978-3-030-95312-6_4.

[28] Craig Costello. B-SIDH: Supersingular Isogeny Diffie-Hellman Using Twisted Torsion. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology -*

ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II, volume 12492 of *Lecture Notes in Computer Science*, pages 440–463. Springer, 2020. doi: 10.1007/978-3-030-64834-3_15. URL https://doi.org/10.1007/978-3-030-64834-3_15.

[29] Craig Costello and Hüseyin Hisil. A simple and compact algorithm for SIDH with arbitrary degree isogenies. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 303–329. Springer, 2017. doi: 10.1007/978-3-319-70697-9_11. URL https://doi.org/10.1007/978-3-319-70697-9_11.

[30] Craig Costello and Benjamin Smith. Montgomery curves and their arithmetic - The case of large characteristic fields. volume 8, pages 227–240, 2018. doi: 10.1007/s13389-017-0157-6. URL https://doi.org/10.1007/s13389-017-0157-6.

[31] Jean-Marc Couveignes. Hard Homogeneous Spaces. Cryptology ePrint Archive, Paper 2006/291, 2006. URL https://eprint.iacr.org/2006/291. https://eprint.iacr.org/2006/291.

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

[33] Luca De Feo. Isogeny Based Cryptography: an Introduction (Slides), 2019. URL https://defeo.lu/docet/assets/slides/2019-11-28-ntnu.pdf.

[34] Luca De Feo and Steven D. Galbraith. Seasign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 759–789. Springer, 2019. doi: 10.1007/978-3-030-17659-4_26. URL https://doi.org/10.1007/978-3-030-17659-4_26.

[35] Luca De Feo, Jean Kieffer, and Benjamin Smith. Towards Practical Key Exchange from Ordinary Isogeny Graphs. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 365–394. Springer, 2018. doi:

10.1007/978-3-030-03332-3_14. URL https://doi.org/10.1007/978-3-030-03332-3_14.

[36] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2020. doi: 10.1007/978-3-030-64837-4_3. URL https://doi.org/10.1007/978-3-030-64837-4_3.

[37] Christina Delfs and Steven D Galbraith. Computing isogenies between supersingular elliptic curves over Fp. *Designs, Codes and Cryptography*, 78(2):425–440, 2016.

[38] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi: 10.1109/TIT.1976.1055638.

[39] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 164–175, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31542-1.

[40] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018 (1):238–268, 2018. doi: 10.13154/tches.v2018.i1.238-268. URL https://doi.org/10.13154/tches.v2018.i1.238-268.

[41] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *ADV. IN CRYPTOLOGY, SPRINGER-VERLAG*, 1985.

[42] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. Measuring HTTPS adoption on the web. In Engin Kirda and Thomas Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, pages 1323–1338. USENIX Association, 2017. URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/felt.

[43] Steven D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012. doi: 10.1017/CBO9781139012843.

[44] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. doi: 10.1145/237814.237866. URL https://doi.org/10.1145/237814.237866.

[45] David Jao and Luca De Feo. Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011. doi: 10.1007/978-3-642-25405-5_2. URL https://doi.org/10.1007/978-3-642-25405-5_2.

[46] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE – Supersingular Isogeny Key Encapsulation, 2017. URL https://sike.org.

[47] Ernst Kani. The number of curves of genus two with elliptic differentials. 1997 (485):93–122, 1997. doi: doi:10.1515/crll.1997.485.93. URL https://doi.org/10.1515/crll.1997.485.93.

[48] C. Karpfinger and K. Meyberg. *Algebra: Gruppen - Ringe - Körper*. SpringerLink : Bücher. Springer Berlin Heidelberg, 2012. ISBN 9783827430120. URL https://books.google.de/books?id=EV7hEcOR1WQC.

[49] Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol. URL https://signal.org/docs/specifications/x3dh/. Accessed Juni 24, 2019.

[50] Moxie Marlinspike and Trevor Perrin. The Double Ratchet Algorithm, 2016. URL https://signal.org/docs/specifications/doubleratchet/. Accessed Feb 01, 2023.

[51] Michael Meyer and Steffen Reith. A faster way to the CSIDH. In Debrup Chakraborty and Tetsu Iwata, editors, *Progress in Cryptology - INDOCRYPT 2018 - 19th International Conference on Cryptology in India, New Delhi, India, December 9-12, 2018, Proceedings*, volume 11356 of *Lecture Notes in Computer Science*, pages 137–152. Springer, 2018. doi: 10.1007/978-3-030-05378-9_8. URL https://doi.org/10.1007/978-3-030-05378-9_8.

[52] Michael Meyer, Fabio Campos, and Steffen Reith. On lions and elligators: An efficient constant-time implementation of CSIDH. In Jintai Ding and Rainer

Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*, volume 11505 of *Lecture Notes in Computer Science*, pages 307–325. Springer, 2019. doi: 10.1007/978-3-030-25510-7_17. URL https://doi.org/10.1007/978-3-030-25510-7_17.

[53] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. URL http://www.bitcoin.org/bitcoin.pdf.

[54] NIST. Post-Quantum Cryptography Standardization, 2016. URL https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization. Accessed March 10, 2023.

[55] NIST. Post-Quantum Cryptography, Round 3 Submissions, 2020. URL https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. Accessed March 10, 2023.

[56] NIST. Post-Quantum Cryptography, Round 4 Submissions, 2022. URL https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions. Accessed March 10, 2023.

[57] Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi. (short paper) A faster constant-time algorithm of CSIDH keeping two points. In Nuttapong Attrapadung and Takeshi Yagi, editors, *Advances in Information and Computer Security - 14th International Workshop on Security, IWSEC 2019, Tokyo, Japan, August 28-30, 2019, Proceedings*, volume 11689 of *Lecture Notes in Computer Science*, pages 23–33. Springer, 2019. doi: 10.1007/978-3-030-26834-3_2. URL https://doi.org/10.1007/978-3-030-26834-3_2.

[58] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. ISBN 978-3-540-46766-3.

[59] Chris Peikert. He gives c-sieves on the csidh. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 463–492, Cham, 2020. Springer International Publishing. ISBN 978-3-030-45724-2.

[60] Joost Renes. Computing isogenies between montgomery curves using the action of (0, 0). In *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*, volume 10786 of *Lecture Notes in Computer Science*, pages 229–247. Springer, 2018. doi:

10.1007/978-3-319-79063-3_11. URL https://doi.org/10.1007/978-3-319-79063-3_11.

[61] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978. ISSN 0001-0782. doi: 10.1145/359340.359342. URL https://doi.org/10.1145/359340.359342.

[62] 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. https://eprint.iacr.org/2006/145.

[63] C. P. Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York. ISBN 978-0-387-34805-6.

[64] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. doi: 10.1109/SFCS.1994.365700.

[65] J.H. Silverman and J.T. Tate. *Rational Points on Elliptic Curves*. Undergraduate Texts in Mathematics. Springer New York, 1994. ISBN 9780387978253.

[66] Joseph H Silverman. *The Arithmetic of Elliptic Curves*. Graduate texts in mathematics. Springer, 2009. doi: 10.1007/978-0-387-09494-6.

[67] Harshdeep Singh. Code based Cryptography: Classic McEliece. *CoRR*, abs/1907.12754, 2019. URL http://arxiv.org/abs/1907.12754.

[68] Jacques Vélu. Isogenies between elliptic curves. *l'Académie, Sci. Paris, Sér. A*, 273:1–5, 1971.

[69] L.C. Washington. *Elliptic Curves: Number Theory and Cryptography*. Discrete Mathematics and Its Applications. CRC Press, 2003. ISBN 9780203484029.

Appendix A

Proofs

A.1 $\pi \circ \pi = [-p]$

Theorem. Let $E(\mathbb{F}_p)$ be a supersingular elliptic curve over $\mathbb{F}_p$ with Frobenius Endomorphism π. Then the composition of $\pi \circ \pi = \pi^2$ gives the multiplication map $[-p]$.

Proof. Hasse Theorem (Theorem 2.2) states $\#E(\mathbb{F}_p) = p + 1 \pm t$, where $|t|$ is the Frobenius trace, which can be reformulated to $t = p + 1 - \#E(\mathbb{F}_p)$. As $E(\mathbb{F}_p)$ is supersingular we have $\#E(\mathbb{F}_p) = p + 1$ and thereby $t = p + 1 - \#E(\mathbb{F}_p) = 0$ [69, Proposition 4.31]. [69, Theorem 4.10] states that $\pi^2 - t\pi + p = 0$ is an endomorphism of $E(\mathbb{F}_p)$. As $t = 0$, $\pi^2 + p = 0$ and thereby $\pi^2 = [-p]$. □

A.2 $a^{(p+1)/4} = \sqrt{a}$

Theorem. Let a be a square in $\mathbb{F}_p$, then $a^{(p+1)/4} \equiv \sqrt{a}$, if and only if $p \equiv 3 \mod 4$.

Proof. Fermats' little theorem gives $a^{p-1} \equiv 1 \mod p$ and from Euler's Criterion we have for an a of form $a = x^2$,

$$a^{(p-1)/2} \equiv (x^2)^{(p-1)/2} \equiv x^{p-1} \equiv 1$$

Extening $a^{(p-1)/2} \equiv 1$ with a gives $a^{(p+1)/2} \equiv a$, which spits into $a^{(p+1)/2} \equiv a^{(p+1)/4} * a^{(p+1)/4} \equiv (a^{(p+1)/4})^2 \equiv a$, thus $a^{(p+1)/4} \equiv \sqrt{a}$, as $(p+1)/4$ is an integer if $p \equiv 3 \mod 4$.

□

Appendix B

Parameters and Benchmarks

B.1 New CSURF-512 Prime

$$
\begin{aligned}
p = 4 * 53 * \\
(2 * 3 * 5 * 7 * 11 * 13 * 17 * 19 * 23 * \\
29 * 31 * 37 * 41 * 43 * 47 * 53 * 59 * 61 * 67 * \\
71 * 73 * 79 * 83 * 89 * 97 * 101 * 103 * 107 * \\
109 * 113 * 127 * 131 * 137 * 139 * 149 * 151 * 157 * \\
163 * 167 * 173 * 179 * 181 * 191 * 193 * 197 * 199 * \\
211 * 223 * 227 * 229 * 233 * 241 * 251 * 257 * 263 * \\
269 * 271 * 283 * 293 * 307 * 311 * 313 * 317 * 331 * \\
337 * 347 * 349 * 353 * 359 * 379 * 383 * 389 * 397 * \\
401) - 1
\end{aligned}
$$

$$
\begin{aligned}
p = 3395414068631999943740201092484489266336316861763180222 \\
2139218327163673764877085916413118523116122461965397847319 \\
63674499489965070706606658106846726347395 9
\end{aligned}
$$

$$
\begin{aligned}
p = 0x40d4703145d63961020936b7b52ec19596552e0e2a1360214ad \\
eb896376b07955ee8aa9f9e03818bf4a5aa25285fc714680d6665c95 \\
ec4733d1777e13d738927
\end{aligned}
$$

B.2 Benchmark Results

Multiplication-tuned

	CSIDH-512	**CTIDH-512**
Mcyc	75.07±6.77	79.12±3.46
mulsq	424402±35347	437984±10950
sq	108306±9889	116790±4336
addsub	449532±37095	482292±9321
mul	316096±26020	321194±6620
combo	497247±41198	516167±12564
	CSURF-512(137)	**CSURF-513**
Mcyc	76.52±7.52	103.08±9.78
mulsq	442442±42094	431058±38934
sq	142712±21044	139650±21849
addsub	412824±36116	395606±31187
mul	299731±26003	291408±22754
combo	511501±47709	497382±43540
	CSURF-512(75A)	**CSURF-512(75B)**
Mcyc	73.05±5.96	72.93±5.60
mulsq	418940±30918	419841±30299
sq	122107±11988	122542±12650
addsub	415412±32198	415776±31108
mul	296833±22755	297299±21979
combo	487358±35786	488334±34824
	CSURF-512(50A)	**CSURF-512(50B)**
Mcyc	72.75±6.28	73.06±7.01
mulsq	415099±33506	414365±37726
sq	116031±11430	117812±12145
addsub	420143±35766	413663±37886
mul	299068±24624	296553±27146
combo185	483922±39014	482305±43746

Untuned

	CSIDH-512	CTIDH-512
Mcyc	74.14±6.46	80.40±3.48
mulsq	421622±35108	446405±10934
sq	106178±9550	116699±4333
addsub	440878±36632	500429±9295
mul	315445±26152	329706±6607
combo	493063±40900	527304±12544
	CSURF-512(137)	**CSURF-513**
Mcyc	74.88±6.13	101.01±9.19
mulsq	436963±34835	425834±37983
sq	135815±19848	133382±20113
addsub	409572±31364	394807±33988
mul	301148±22613	292452±24240
combo	505190±39147	491724±42919
	CSURF-512(75A)	**CSURF-512(75B)**
Mcyc	75.14±6.09	75.04±6.44
mulsq	425918±31446	429381±34116
sq	121105±13319	123475±13526
addsub	419510±31306	420411±34757
mul	304812±22162	305906±24611
combo	494899±36119	498616±39407
	CSURF-512(50A)	**CSURF-512(50B)**
Mcyc	73.40±6.46	73.55±5.84
mulsq	420103±33814	425878±32329
sq	116579±11407	119994±10658
addsub	418444±34836	419877±32694
mul	303525±24464	305884±23602
combo	488699±39260	494859±37456

Notation

$[m]$	the multiplication-by-m map for point on elliptic curves.
$\mathfrak{a}, \mathfrak{b}, \mathfrak{c}, \ldots$	ideals.
a,M,S,I,Sqrt	cost of addition, muliplication, squaring, inversion and square root computation.
p	a prime number.
q	a prime power p^n.
$cl(\mathcal{O})$	the ideal class group of $\mathcal{O}$.
E	an elliptic curve.
$E(\mathbb{K})$	an elliptic curve over $\mathcal{K}$.
$E[m]$	the m-torsion of an elliptic curve E.
$\mathrm{End}(E)$	the endomorphism ring of E.
$\mathrm{End}_{\mathbb{F}_p}(E)$	the $\mathbb{F}_p$-rational endomorphism ring of E.
$\mathcal{E}\ell\ell$	a set of elliptic curves.
$\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$	the set of supersingular elliptic curves over $\mathbb{F}_p$ with $\mathrm{End}_{\mathbb{F}_p}(E) \cong \mathcal{O}$.
$\mathbb{F}_p$	a finite field with p elements.
G	a group action on a hard homogeneous space.
H	a hard homogeneous space.
$\mathbb{K}$	a field.
$\overline{\mathbb{K}}$	the algebraic closure of a field $\mathbb{K}$.
$ker(\varphi)$	the kernel of the map φ.
$O, \tilde{O}$	Big-O notation.
$\mathcal{O}$	an order of an imaginary quadratic field.
P, Q, R	points on an elliptic curve.
∞	the point at infinity on an elliptic curve.
φ	an isogeny.
π	the Frobenius Endomorphism.
$\mathbb{Z}/p\mathbb{Z}$	the ring of integers modulo p.