Computing Quotient Groups of Smooth Order with Applications to Isogenies over Higher-Dimensional Abelian Varieties

Jesús-Javier Chi-Domínguez[1], Amalia Pizarro-Madariaga[2], and Edgardo Riquelme[3]

[1] Cryptography Research Center, Technology Innovation Institute, Abu Dhabi, UAE
[2] Instituto de Matemáticas, Universidad de Valparaíso, Chile
[3] Departamento de Ciencias Básicas, Universidad del Bío Bío, Chile
jesus.dominguez@tii.ae,amalia.pizarro@uv.cl,edriquelme@ubiobio.cl

Abstract. There is an increasing interest in efficiently computing isogenies with a kernel of large-smooth size, for instance, as a building block for building secure Proof-of-Knowledge (PoK) with isogenies of degree equals a power of a small prime number. Another example corresponded to the attacks started by Castyck and Decru and followed up by Maino-Martindale and Robert, which require calculating isogenies over superspecial principally polarized abelian surfaces (superspecial PPAS). On the opposite side of cryptanalysis, some of the current state-of-the-art on safe isogeny-based PoK constructions extends to the case of superspecial PPAS, with the property that one could use smaller fields (e.g., 128, 192, and 256 bits).

This work presents a general framework that generalizes the situation of computing isogenies of the large-smooth degree to the context of quotient groups. More precisely, we abstract and propose a generalization of the strategy technique by Jao, De Feo, and Plût. Such a framework provides an efficient generic algorithm that easily applies to computing isogenies over superspecial PPAS when given the isogeny kernel. Additionally, our algorithm induces an efficient algorithm to perform the KernelToIsogeny procedure required in SQISignHD.

To illustrate the impact of optimal strategies, we draft our experiments on the isogenies over superspecial PPAS required in the Castryck-Decru attack (powers of two and three). Our experiments illustrate a decent speed up of 1.25x faster than the state-of-the-art (about 20% of savings). Our results should be viewed as proof-of-concept implementation and considered for optimized C-language implementations.

Keywords: Elliptic Curves · Isogenies · Quotient Groups · Strategies · superspecial PPAS

1 Introduction

Isogenies of elliptic curves have played an important role in post-quantum cryptography. Informally speaking, isogenies are peculiar objects being surjective morphism (over the algebraic closure) between abelian varieties that fix the point at infinity (the neutral element). Mainly, isogenies act as group homomorphism.

The most known isogeny-based protocols based on Diffie-Hellman-like constructions are

- The Supersingular Isogeny Diffie-Hellman [17],
- The Supersingular Isogeny Key Encapsulation SIKE [2],
- The Commutative SIDH (CSIDH) suggests working over prime fields [8],
- The B-SIDH proposal from [12] suggests using twist-quadratic curves, and
- The genus-two variant of SIDH (G2SIDH) from [23] suggests using isogenies over higher-dimensional abelian varieties.

The SIKE protocol was the only proposal participating in the Post-Quantum NIST competition based on isogenies. Nevertheless, the recent attacks in [6,28,30] break SIDH/SIKE (and theoretically B-SIDH) in minutes. The most demanded calculations in these attacks are the computation of isogenies over higher-dimensional abelian varieties. That is primarily because the attacks translate the initial isogeny problem into higher-dimensional abelian varieties.

However, beyond the key-exchange protocols proposed in [17,8], isogenies have become a powerful tool in building practical post-quantum secure [Signature/PoK] schemes proposed in [18,16,19,3,9,25,15]. Aside from those works, there is interest in using such isogenies for building protocols on top of Jacobian of genus-two curves and the product of elliptic curves [9,15]. In particular, all the constructions from [17,16,9,3,25] can be extended to the Jacobian of genus-two curves and the product of elliptic curves, allowing them to work with smaller fields [14].

In the elliptic curve scenario, the most efficient approach to computing isogenies of smooth large-degree [4] is splitting the computation into small-degree isogenies. Such an approach was initially described in [17], where the authors introduced the concept of strategies and followed up in [26,10,1,21]. The strategy approach reduces the computational cost of computing a ℓ^n-isogeny from $O(n^2)$ to $O(n \log n)$ scalar points multiplications by ℓ and ℓ-isogeny evaluations, where ℓ is a prime number.

Considering that obtaining the codomain curve of an isogeny is computing (as a group) the quotient between the domain curve and the isogeny kernel, we can generalize the strategies concept to a more general setting. In the present work, we generalize this construction, focusing on the mathematical abstraction of computing the quotient group G/H for a pair of smooth-order groups G and $H \leq G$. We apply this abstraction to obtain optimized computation of (ℓ^n, ℓ^n)-isogenies over higher-dimensional varieties.

[4] The term smooth-order kernel refers to a kernel of order that equals $N = \ell_1 \cdots \ell_n$ for some small prime primes ℓ_i's. No restriction on the ℓ_i's, and there can be repeated ℓ_i's in the factors of N. We assume N has hundreds of bits. Similarly, we refer to a large-smooth integer as an integer like $N = \ell_1 \cdots \ell_n$.

1.1 Contributions

For simplicity, in this section, we use a multiplicative group notation. That is, G denotes the group $(G, *)$ and gh means $g * h$ for any pair of elements $g, h \in G$.

As the main contribution, we abstract and generalize the strategy technique used to compute isogenies (between elliptic curves) with a given smooth-order [4] kernel [17,26,10,1,21], this time to calculate the quotient group. We define two new objects required to decompose the quotient group computation into a chain of small prime-order quotient subgroups.

Definition 1.1 (Informal definition of a QAB). *Given a prime number ℓ and an integer $k \geq 1$. A QAB_k^ℓ for a group G and order-ℓ^k subgroup $H \leq G$ is a 3-tuple composed by three algorithms $\mathsf{Multiply}_\ell \colon g \in G \mapsto g^\ell \in G$, $\mathsf{Evaluate}_\ell \colon (g, H) \mapsto gH \in G/H$ and $\mathsf{Construct} \colon H \mapsto G/H$.*

Definition 1.2 (Informal definition of a CAB). *Given a composite integer $L = \ell_1 \cdots \ell_n$ with small primes factors ℓ_i's and an integer $k \geq 1$. A CAB_k^L for a group G and an order-L^k subgroup $H \leq G$ is a collection of all the $\mathsf{QAB}_k^{\ell_i}$'s along with an extra algorithm $\mathsf{Compose}_L \colon H \mapsto G/H$.*

One can compute G/H using $\mathsf{Construct}$ with the large integer L instead of a small prime ℓ. Nevertheless, the procedure $\mathsf{Construct}$ usually runs in polynomial time concerning the input order. Therefore, computing G/H may not be feasible for large values of $L \approx 2^\lambda$. For smooth-order subgroups H, one can calculate G/H in polynomial time concerning each small prime factor of L, considerably improving the naive approach. We informally describe the strategy technique on group quotients in the following definition.

Definition 1.3 (Informal definition of a strategy for a CAB). *Given a composite integer $L = \ell_1 \cdots \ell_n$ with small primes factors ℓ_i's and an integer $k \geq 1$, and a CAB_k^L for an order-L^k subgroup H. An strategy St_n^L for CAB_k^L is a technique to perform $\mathsf{Compose}_L$ in polynomial time concerning the variable $(\ell_1, \ldots, \ell_n)$.*

Remark 1.1 (Informal codification of the strategies). Any strategy for a CAB_k^L can be expressed as a list of $(n-1)$ positive integers. We provide an efficient algorithm for computing and evaluating any optimal strategy.

Our generalization is not limited to L-isogenies between elliptic curves but also applies to compute (L, L)-isogenies between superspecial principally polarized abelian surface (superspecial PPAS). In the following two informal lemmas, we characterize the isogeny computations over elliptic curves and superspecial PPAS as CABs.

Lemma 1.1 (Informal first lemma). *Given a supersingular curve E and a kernel subgroup H of order L on E. The task of computing the L-isogeny $\phi \colon E \to E/H$ with kernel H is a special case of a CAB_1^L.*

Lemma 1.2 (Informal second lemma). *Given a superspecial PPAS $\mathcal{A}$ and a kernel subgroup H of order L^2, generated by two elements, on A. The task of computing the (L, L)-isogeny $\phi\colon \mathcal{A} \to \mathcal{A}/H$ with kernel H is a special case of a* CAB_2^L.

Consequently, our main new results center on an efficient technique to construct (L, L)-isogenies over PPAS, particularly for computing $(2^n, 2^n)$-isogenies and $(3^n, 3^n)$-isogenies. Our strategy technique theoretically improves the (L, L)-isogeny constructions required in the attacks presented in [6,28,30]. We additionally provide a proof-of-concept implementation using the Magma Computer Algebra System and the SageMath library. Our local experiments draw a decent speed up of about 1.25x compared with state-of-the-art techniques [5].

1.2 Organization of the Paper

The paper organizes as follows. Section 2 lists the required group theory, elliptic curve, and superspecial PPAS background.

Section 3 presents the generic framework to compute quotient groups efficiently. In particular, Section 3.1 discusses the (optimal) strategies when the group sizes are large-smooth integers (not necessarily a power of a prime number). In contrast, Section 3.2 describes the particular case of a power of a small prime integer (i.e., the straightforward extension of strategy by Jao, De Feo, and Plut). Both sections provide algorithms to compute and evaluate the optimal strategies. By "evaluate a strategy", we mean a procedure that calculates quotient groups efficiently by employing such a strategy.

Section 4 lands the results from Section 3 to the case of superspecial PPAS when the kernel has a size that equals a power of a small prime. Moreover, Section 4.1 gives the state-of-the-art concerning $(3, 3)$-isogenies over superspecial PPAS (the $\mathsf{CAB}_3^{3^n}$'s built-in functions) and compares against the state-of-the-art for computing $(3^n, 3^n)$-isogenies. Analousgly, Section 4.2 illustrates the state-of-the-art concerning $(2, 2)$-isogenies (the $\mathsf{CAB}_2^{2^n}$'s built-in functions) and compares against the state-the-art for computing $(2^n, 2^n)$-isogenies.

Finally, Section 5 summarizes the concluding remarks and discusses the potential applications of our results.

2 Preliminaries

This section introduces the basics of group theory needed to describe the extension of strategies to quotient groups. Additionally, it presents the required definitions to land our results into the case of supersingular elliptic curves and principally polarized abelian surfaces.

Let n be a positive integer. We use the notation $[\![n]\!]$ to refer the list (in increasing order) $[1, \ldots, n-1, n]$, while $]\!]n[\![$ denotes the list (in decreasing order)

[5] Our code is freely available in this GitHub repository.

$[n, n-1, \ldots, 1]$. The list of n ones is denoted as $[\![1]\!]^n$. We represent vectors by bold letters (e.g., $\mathbf{v}$) and sub-indexes label each entry of such vectors (e.g., $\mathbf{v}_i$).

An abelian group $(G, *)$ is a 2-tuple composed of a non-empty set G and a closed binary operation $*\colon G \times G \to G$ such that the following four group axioms hold:

– **Identity element:** There is a unique element $e \in G$ such that for any $g \in G$ we have $e * g = g = g * e$.
– **Inverse element:** For any element $g \in G$ there exists $\overline{g} \in G$ such that $g * \overline{g} = e = \overline{g} * g$.
– **Associativity:** For any $g, h, i \in G$, we have $(g * h) * i = g * (h * i)$.
– **Commutativity:** For any $g, h \in G$, we have $g * h = h * g$.

The order of a group $(G, *)$ is the size of G, denoted by $\#G$. The order of an element $g \in G$ is the smallest integer $n \geq 1$ such that

$$\underbrace{g * \cdots * g}_{n \text{ times}} = e.$$

A subgroup $(H, *)$ of a group $(G, *)$, denoted by $H \leq G$, is a group such that $H \subseteq G$. A group homomorphism $f\colon G \to H$ is a function between two groups, $(G, *)$ and $(H, \star)$, such that for all $g_1, g_2 \in G$ it holds that

$$f(g_1 * g_2) = f(g_1) \star f(g_2).$$

In particular, we have that $f(e_G) = e_H$ and $f(\overline{g}) = \overline{f(g)}$ for all $g \in G$. The kenerl of f is defined as $\ker f := \{g \in G\colon f(g) = e_H\}$, while its image is $\{f(g)\colon g \in G\}$. Given a subgroup $H \leq G$ of the abelian group G, the quotient group G/H correspond with the set $\{g*H\colon g \in G\}$ where $g*H := \{g*h\colon h \in H\}$, and the closed binary operation is $\star\colon (g_1 * H, g_2 * H) \mapsto (g_1 * g_2) * H$. This time the identity element is H, and the inverse of $g * H \in G/H$ is $\overline{g} * H$.

Theorem 2.1 (The First Isomorphism Theorem). *If $f\colon G \to G'$ is a group homomorphism between two groups G and G', then $G/\ker f$ is isomorphic to the image of f.*

Direct products. The direct product $(G \times H, \bullet)$ of two groups $(G, *)$ and $(H, \star)$ is defined as follows:

– The underlying set is the cartesian product $G \times H$.
– The closed binary operation on $G \times H$ is defined component-wise:

$$\bullet\colon (\mathbf{x}, \mathbf{y}) \mapsto (\mathbf{x}_1 * \mathbf{y}_1, \mathbf{x}_2 \star \mathbf{y}_2).$$

The identity element in $G \times H$ is $\mathbf{e} := (e_G, e_H)$, and we say that $\mathbf{v} \in G \times H$ is an order-(n, m) element if $\mathbf{v}_1$ and $\mathbf{v}_2$ have order n and m. More generally, one can define the direct product concerning the cartesian product of k groups G_i's, and the above definition and properties easily extend to k-dimensional vector elements. In particular, we define that an element $\mathbf{g} = (\mathbf{g}_1, \ldots, \mathbf{g}_i)$ in $\mathbf{g} \in \prod_{i=1}^{k} G_i$ has order-$(n, \ldots, n)$ if each $\mathbf{g}_i$ has order n. If $G_i = G$, with $i = 1, \ldots, k$ for some fixed group $(G, *)$, we write its direct product as $(G^k, \bullet)$.

Remark 2.1. By group law, we mean the closed binary operation that satisfies the four above properties. For short, we write G instead of $(G, *)$ when there is no ambiguity in the group law.

Supersingular Elliptic Curves. Let p be a prime number such that $p \equiv 3 \mod 4$. We denote by $\mathbb{F}_p$ the prime field with p elements, and its quadratic extension as $\mathbb{F}_{p^2} := \mathbb{F}_p[i]/(i^2 + 1)$. Let E be a supersingular Montgomery curves given by Equation 1

$$E \colon y^2 = x^3 + Ax^2 + x, \quad A \in \mathbb{F}_{p^2} \setminus \{\pm 2\}. \tag{1}$$

Every supersingular elliptic curve can be defined over $\mathbb{F}_{p^2}$ and they have exactly $\#E(\mathbb{F}_{p^2}) = (p \pm 1)^2$ points. We restrict our analysis to the case $\#E(\mathbb{F}_{p^2}) = (p + 1)^2$. We use $\mathcal{O}$ to denote the point at infinity of E (the neutral element). For any field extension $\mathbb{F}_q$ of $\mathbb{F}_p$, the set of $\mathbb{F}_q$-rational points $E(\mathbb{F}_q) := \{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q \colon y^2 = x^3 + Ax^2 + x\}$ forms an additive group under the chord-and-tangent rule. We write $E[\ell]$ to refer the ℓ-torsion subgroup $\{P \in E\left(\overline{\mathbb{F}}_{p^2}\right) \mid [\ell]P = \mathcal{O}\}$ of E, where

$$[\ell]P = \underbrace{P + \cdots + P}_{\ell \text{ times}}.$$

An isogeny $\phi \colon E \to E'$ is a morphism satisfying, $\phi(\mathcal{O}_E) = \mathcal{O}_{E'}$. Isogenies are rational functions and if their coefficient belongs to a field $\mathbb{F}_q$, we say that the isogeny is defined over $\mathbb{F}_q$. If such isogeny exists, we say E and E' are isogenous over $\mathbb{F}_q$ which happens if and only if $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$. In particular, ϕ is a homomorphism of groups. We say that a (separable) isogeny ϕ is an ℓ-isogeny when $\# \ker \phi = \ell$ holds.

Principally Polarized Abelian Surface (PPAS). Let C be a genus two hyperelliptic curves given by Equation 2. There is no restriction on the leading coefficient of $f(x)$, so must of the time we assume $f(x)$ is not monic.

$$C \colon y^2 = f(x), \quad f(x) \in \mathbb{F}_{p^2}[x] \text{ with } \deg f = 6. \tag{2}$$

The Jacobian group J_C of C is an abelian variety of dimension 2 (i.e., abelian surface). We next summarize what the Jacobian elements look like (for a deeper understanding of Jacobian groups, we recommend reading [24]). There are two points at infinity ∞_+ and ∞_- on C. The negative of a point $P = (x, y)$ on C is $-P = (x, -y)$. A divisor is a formal sum of points on C. Every nonzero rational function on C has an associated divisor labeled as principal. Two divisors are equivalent if their difference is a principal divisor. The canonical equivalence class of divisors of the form $\mathcal{O} = (P) + (-P)$ determines the identity element in J_C. Any other divisor in J_C is of form $(P)+(Q)+(\mathcal{O})$, where P and Q are points on C (including the points at infinity ∞_+ and ∞_-). For simplicity, we denote $(P) + (Q) + (\mathcal{O})$ by $[P + Q]$. Additionally, the group law on J_C can be efficiently computed using Cantor's algorithm under the Mumford representation (pair of polynomials) [11, Chapter 14].

A principally polarized abelian surface (PPAS) $\mathcal{A}$ is either the product of two elliptic curves $E \times E'$ or the Jacobian J_C of a genus two curve C. Analogously to elliptic curves, an isogeny $\phi\colon \mathcal{A} \to \mathcal{A}'$ over $\mathbb{F}_q$ is a morphism satisfying, $\phi(\mathcal{O}_{\mathcal{A}}) = \mathcal{O}_{\mathcal{A}'}$. Moreover, ϕ is also a homomorphism of groups. A (ℓ, ℓ)-isogeny is an isogeny $\phi\colon \mathcal{A} \to \mathcal{A}'$ between two PPAS $\mathcal{A}$ and $\mathcal{A}'$ whose kernel is a maximal ℓ-Weil isotropic subgroup of the ℓ-torsion subgroup $\mathcal{A}[\ell]$. The kernel of a (ℓ, ℓ)-isogeny will called a (ℓ, ℓ)-subgroup. In particular, $\ker \phi \cong \mathbb{Z}_\ell \times \mathbb{Z}_\ell$ holds [6]. An abelian surface $\mathcal{A}$ over $\mathbb{F}_q$ is a principally polarized superspecial abelian surface if $\mathcal{A}$ is a PPAS isomorphic over $\overline{\mathbb{F}}_q$ (as an unpolarized abelian variety) to a product of supersingular elliptic curves. In particular, every principally polarized superspecial abelian surface can be defined over $\mathbb{F}_{p^2}$. If we consider $\mathcal{A}$ being (m, m)-isogenous, for some $m \nmid p$, over $\mathbb{F}_{p^2}$ to a product of supersingular elliptic curves with $(p + 1)^2$ points, then we can assume that $\#\mathcal{A}(\mathbb{F}_{p^2}) = (p + 1)^4$. We focus on the principally polarized superspecial abelian surfaces, but our results easily extend to any PPAS. In the rest of the work, we write superspecial PPAS to refer to principally polarized superspecial abelian surfaces.

3 Computing Quotient Groups

This section describes the main results and extends the definition of "strategies" in the context of computing group quotients. It generalizes the situation of computing smooth degree isogenies as in [26,10,1] when given the kernel generator. Additionally, it presents the extension to the case of a power of a small prime as in [17,2,21].

Definition 3.1 (Quotient Atomic Block: QAB_k^ℓ). *Let $N = \ell_1 \cdots \ell_n$ be an integer with n small primes ℓ_i's, and let $(G, *)$ be a finite abelian group of smooth order N^m for some positive integer m. Let us fix a positive integer $k \in [\![m]\!]$, and let $(G^k, \bullet)$ be the direct product of k copies of G with the binary operation*

$$\bullet\colon (\mathbf{g}, \mathbf{g}') \mapsto (\mathbf{g}_1 * \mathbf{g}'_1, \ldots, \mathbf{g}_k * \mathbf{g}'_k).$$

[6] We recommend reading [22,27,20] for the cases $\ell = 2, 3$.

Let ℓ be a prime integer number in $\{\ell_1, \dots, \ell_n\}$, and $\mathbf{h} := (\mathbf{h}_1, \dots, \mathbf{h}_k) \in G^k$ of order $(\ell, \dots, \ell)$ such that $\mathbf{H}_\ell := \langle \mathbf{h}_1, \dots, \mathbf{h}_k \rangle \leq G$ has order ℓ^k. A Quotient Atomic Block, QAB_k^ℓ for short, is composed of three main algorithms:

- $\mathsf{Multiply}_\ell(\mathbf{g})$*: takes as input an element $\mathbf{g} \in G^k$ and returns $\mathbf{g}' = \bullet^\ell \mathbf{g}$ where $\bullet^\ell$ denotes the scalar multiplication by ℓ, i.e.,*

$$\bullet^\ell \colon \mathbf{g} \mapsto \underbrace{\mathbf{g} \bullet \cdots \bullet \mathbf{g}}_{\ell \text{ times}}.$$

- $\mathsf{Construct}_\ell(\mathbf{h})$*: takes as input an order-$(\ell, \dots, \ell)$ element $\mathbf{h} \in G^k$ as defined above, and returns the quotient subgroup $G' = G/\mathbf{H}_\ell$.*
- $\mathsf{Evaluate}_\ell(\mathbf{g}, \mathbf{h})$*: takes as input two group elements $\mathbf{g}, \mathbf{h} \in G^k$, with $\mathbf{h}$ of order $(\ell, \dots, \ell)$ as defined above, and returns $\mathbf{g}' := \mathbf{g} \bullet \mathbf{H}_\ell$, the image of $\mathbf{g}$ under the canonical projection π of G onto $G' = G/\mathbf{H}_\ell$.*

Definition 3.2 (Chained Atomic Block (CAB_k^L)). *Let $N = \ell_1 \cdots \ell_n$ be a integer with n small primes ℓ_i's, and let $(G, *)$ be a finite abelian group of smooth order N^m for some positive integer m. Let $L = \ell_{\sigma(1)} \cdots \ell_{\sigma(n')}$ be a smooth factor of N for some integer $1 \leq n' \leq n$ and permutation $\sigma \colon [\![n]\!] \to [\![n]\!]$, and let $\mathbf{h} := (\mathbf{h}_1, \dots, \mathbf{h}_k) \in G^k$ of order $(L, \dots, L)$ such that $\mathbf{H}_L := \langle \mathbf{h}_1, \dots, \mathbf{h}_k \rangle \leq G$ has order L^k. A Chained Atomic Block, CAB_k^L for short, is a collection of n' QABs along with an algorithm $\mathsf{Compose}_L$,*

$$\mathsf{CAB}_k^L := \left(\left\{ \mathsf{QAB}_k^{\ell_{\sigma(i)}} \right\}_{i=1}^{n'}, \mathsf{Compose}_L \right), \tag{3}$$

such that

- $\mathsf{Compose}_L(\mathbf{h})$*: takes as input an order-$(L, \dots, L)$ element $\mathbf{h} \in G^k$ as defined above, and returns the quotient subgroup $G' = G/\mathbf{H}_L$.*

Lemma 3.1 (Supersingular Elliptic Curves). *Given a supersingular curve E defined over $\mathbb{F}_{p^2}$ and H a subgroup of order L on $E(\mathbb{F}_{p^2})$. The task of computing the L-isogeny $\phi \colon E \to E/H$ with kernel H is a special case of a CAB_1^L.*

Proof. Let $E/\mathbb{F}_{p^2}$ be a supersingular elliptic curve satisfying $\#E(\mathbb{F}_{p^2}) = (p+1)^2$. If $L \mid (p+1)^2$, then $E[L] \leq E(\mathbb{F}_{p^2})$, so every subgroup H of order L of E is defined over $\mathbb{F}_{p^2}$. There is a unique (up to isomorphism) elliptic curve E' and isogeny $\phi \colon E \to E'$ such that ker $\phi = H$. Combining the First Isomorphism Theorem with the fact that $\phi \colon E(\overline{\mathbb{F}}_{p^2}) \to E'(\overline{\mathbb{F}}_{p^2})$ is surjective, we have that

$$E(\overline{\mathbb{F}}_{p^2}) / \ker \phi \cong E'(\overline{\mathbb{F}}_{p^2}).$$

In this case, the isogeny is defined over $\mathbb{F}_{p^2}$, so the curve E' is also defined over $\mathbb{F}_{p^2}$. The algorithm $\mathsf{Multiply}_\ell(P)$ corresponds with the scalar multiplication by ℓ in elliptic curves, i.e., $\mathsf{Multiply}_\ell(P) = [\ell]P$, where $P \in E(\mathbb{F}_{p^2})$. The algorithm

$\mathsf{Construct}_\ell(H)$ returns the codomain curve E' of the isogeny ϕ with kernel H and the algorithm $\mathsf{Evaluate}_\ell(P, H)$, returns the point $\phi(P) \in E'(\mathbb{F}_{p^2})$. Notice that although the group G should be $E(\overline{\mathbb{F}}_{p^2})$, in practice, we only consider curves and points over $\mathbb{F}_{p^2}$; therefore, the curves are also defined over $\mathbb{F}_{p^2}$. □

Lemma 3.2 (Superspecial PPAS). *Given a superspecial PPAS $\mathcal{A}$ defined over $\mathbb{F}_{p^2}$ and a (L, L)-subgroup H of order L^2, generated by two order-L elements, on $\mathcal{A}$. The task of computing the (L, L)-isogeny $\phi\colon \mathcal{A} \to \mathcal{A}/H$ with kernel H is a special case of a CAB_2^L.*

Proof. Let C be a hyperelliptic genus 2 curve over $\mathbb{F}_{p^2}$ such that J_C is a superspecial PPAS defined over $\mathbb{F}_{p^2}$ with $\#\mathsf{J}_C(\mathbb{F}_{p^2}) = (p+1)^4$. Then if $L \mid (p+1)$, then $\mathsf{J}_C[L] \leq \mathsf{J}_C(\mathbb{F}_{p^2})$ and every subgroup H of order L^2 of J_C is defined over $\mathbb{F}_{p^2}$. Let H be a (L, L)-subgroup H of order L^2. Then, there is a unique superspecial PPAS $\mathsf{J}_{C'}$ and (L, L)-isogeny $\phi : \mathsf{J}_C \to \mathsf{J}_{C'}$ such that $\ker \phi = H$. Analogously to the elliptic curve's case, we have that

$$\mathsf{J}_C(\overline{\mathbb{F}}_{p^2})/\ker\phi \cong \mathsf{J}_{C'}(\overline{\mathbb{F}}_{p^2}),$$

and the three algorithms $\mathsf{Multiply}_\ell(P)$, $\mathsf{Construct}_\ell(H)$ and $\mathsf{Evaluate}_\ell(P, H)$, are considered in the superspecial PPAS's setting. More precisely, the algorithm $\mathsf{Multiply}_\ell(P)$ corresponds with the scalar multiplication by ℓ in the Jacobian of the curve C, i.e., $\mathsf{Multiply}_\ell(P) = [\ell]P$, where $P \in \mathsf{J}_C(\mathbb{F}_{p^2})$. The algorithm $\mathsf{Construct}_\ell(H)$ returns the codomain curve C' of the isogeny ϕ with kernel H and the algorithm $\mathsf{Evaluate}_\ell(P, H)$, returns the point $\phi(Q) \in J_{C'}(\mathbb{F}_{p^2})$. Analogously to the elliptic-curve case, all the points and curves are defined over $\mathbb{F}_{p^2}$. □

Definition 3.3 (Strategies for a two order-L elements CAB_k^L). *Let $N = \ell_1 \cdots \ell_n$ be an integer with n small odd primes ℓ_i's, and let $(G, *)$ be a finite abelian group of smooth order N^m for some positive integer m. Let $L = \ell_{\sigma(1)} \cdots \ell_{\sigma(n')}$ be a smooth factor of N for some integer $1 \leq n' \leq n$ and permutation $\sigma\colon [\![n]\!] \to [\![n]\!]$. Let CAB_k^L be a chained atomic block for a finite abelian group G as defined in Definition 3.2 and given by Equation (3). Let $\mathbf{h} \coloneqq (\mathbf{h}_1, \ldots, \mathbf{h}_k) \in G^k$ of order $(L, \ldots, L)$ such that $\mathbf{H}_L \coloneqq \langle \mathbf{h}_1, \ldots, \mathbf{h}_k\rangle \leq G$ has order L^k. Let $\Delta_{n'}^L$ be a discrete rectangular triangular with the point at the right angle $\mathbf{h}_0 = \mathbf{h}$, opposite cathetus be composed by*

$$\begin{aligned}
\mathbf{h}_0 &= \mathbf{h},\\
\mathbf{h}_1 &= \mathsf{Multiply}_{\ell_{\sigma(1)}}\mathbf{h}_0,\\
\mathbf{h}_2 &= \mathsf{Multiply}_{\ell_{\sigma(2)}}\mathbf{h}_1,\\
&\vdots\\
\mathbf{h}_{n'-1} &= \mathsf{Multiply}_{\ell_{\sigma(n'-1)}}\mathbf{h}_{n'-2},
\end{aligned}$$

the hypotenuse determined by following order-$(\ell_{\sigma(i)}, \ldots, \ell_{\sigma(i)})$ elements

$$\begin{aligned}
\mathbf{h}^{(1)} &= \mathbf{h}_{n'-1},\\
\mathbf{h}^{(2)} &= \mathsf{Evaluate}_{\ell_{\sigma(n')}}\left(\mathbf{h}_{n'-2}, \mathbf{h}^{(1)}\right),\\
\mathbf{h}^{(3)} &= \mathsf{Evaluate}_{\ell_{\sigma(n'-1)}}\left(\mathsf{Evaluate}_{\ell_{\sigma(n')}}\left(\mathbf{h}_{n'-3}, \mathbf{h}^{(1)}\right), \mathbf{h}^{(2)}\right),\\
&\vdots\\
\mathbf{h}^{(n')} &= \mathsf{Evaluate}_{\ell_{\sigma(2)}}\left(\ldots \mathsf{Evaluate}_{\ell_{\sigma(n')}}\left(\mathbf{h}_0, \mathbf{h}^{(1)}\right)\ldots, \mathbf{h}^{(n'-1)}\right).
\end{aligned}$$

and the adjacent cathetus given by

$$\begin{aligned}
\mathbf{h}'_0 &= \mathbf{h}_0,\\
\mathbf{h}'_1 &= \mathsf{Evaluate}_{\ell_{\sigma(n')}}\left(\mathbf{h}'_0, \mathbf{h}^{(1)}\right),\\
\mathbf{h}'_2 &= \mathsf{Evaluate}_{\ell_{\sigma(n'-1)}}\left(\mathsf{Evaluate}_{\ell_{\sigma(n')}}\left(\mathbf{h}'_2, \mathbf{h}^{(1)}\right), \mathbf{h}^{(2)}\right),\\
&\vdots\\
\mathbf{h}'_{n'-1} &= \mathsf{Evaluate}_{\ell_{\sigma(2)}}\left(\ldots \mathsf{Evaluate}_{\ell_{\sigma(n')}}\left(\mathbf{h}'_0, \mathbf{h}^{(1)}\right)\ldots, \mathbf{h}^{(n'-1)}\right).
\end{aligned}$$

Any other point in $\Delta^L_{n'}$ corresponds with scalar multiplications and evaluations of the cathetus. Notice that $\mathbf{h}'_{n'-1}$ and $\mathbf{h}_{n'}$ are equal, and the hypotenuse implicitly describes a path between subgroups to compute $G_{n'} = G/\mathbf{H}_L$,

$$G_0 = G \to G_1 = G_0/\mathbf{h}^{(1)} \to G_2 = G_1/\mathbf{h}^{(2)} \to \cdots \to G_{n'} \cong G_{n'-1}/\mathbf{h}^{(n')}.$$

A strategy for a CAB^L_k is an extra attribute described by a weighted binary tree $\mathsf{St}^L_{n'}$ inside $\Delta^L_{n'}$ with root $\mathbf{h}_0$ and leaves $\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(n')}$. Implicitly, a strategy $\mathsf{St}^L_{n'}$ provides an efficient technique to perform $\mathsf{Compose}_L(\mathbf{h})$.

3.1 Different kinds of strategies

One crucial remark is that any strategy, as defined in Definition 3.3, can be recursively decomposed into two binary sub-trees, one contained in $\Delta^{L_{n'-h}}_{n'-h}$ and another in $\Delta^{L_h}_h$, where $L_h = [\ell_{\sigma(1)}, \ldots, \ell_{\sigma(h)}]$ and $L_{n'-h} = [\ell_{\sigma(h+1)}, \ldots, \ell_{\sigma(n')}]$. Such decomposition permits representing any strategy as a positive integer list of $n' - 1$ elements, where each entry determines the height $n' - h$ (resp. h) of the left-side (resp. right-side) sub-tree. One computes h multiplications (resp. $n' - h$ evaluations) to move into the left-side (resp. right-side) sub-tree. Figure 1 illustrates a strategy's shape and general idea. On that basis, any strategy costs smaller than or equal to the cost of constructing the whole triangle $\Delta^L_{n'}$. Since

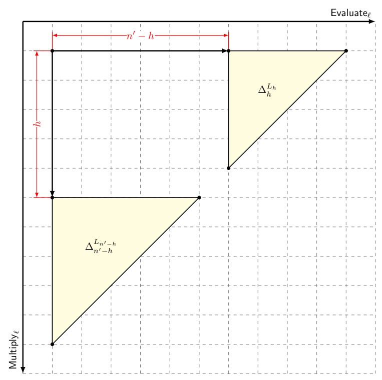

Fig. 1: Strategy technique: reduce the computations from $\Delta_{n'}^L$ into two binary sub-trees, one contained in $\Delta_{n'-h}^{L_{n'-h}}$ and another in $\Delta_h^{L_h}$, where $L_h = [\ell_{\sigma(1)}, \ldots, \ell_{\sigma(h)}]$ and $L_{n'-h} = [\ell_{\sigma(h+1)}, \ldots, \ell_{\sigma(n')}]$ and $h \in [\![n' - 1]\!]$.

$\Delta_{n'}^L$ has $\sum_{j=1}^{n'-1} j = \frac{(n'-1)n'}{2}$ points, the maximum number of multiplications and evaluations is $\frac{(n'-1)n'}{2}$.

Therefore, if the three algorithms, $\mathsf{Multiply}_\ell$, $\mathsf{Construct}_\ell$, and $\mathsf{Evaluate}_\ell$, run in polynomial time for each $\ell \in \{\ell_{\sigma(1)}, \ldots, \ell_{\sigma(n')}\}$, then a polynomial time algorithm exists concerning the variable $(\ell_{\sigma(1)}, \ldots, \ell_{\sigma(n')})$ to perform $\mathsf{Compose}_L$. We summarize the above observations in Algorithm 1. The associated cost of a strategy $\mathsf{St}_{n'}^L$ is

$$\mathsf{C}\left(\mathsf{St}_{n'}^L\right) = \mathsf{C}\left(\mathsf{St}_{n'-h}^{L_{n'-h}}\right) + \mathsf{C}\left(\mathsf{St}_h^{L_h}\right) + \sum_{j=1}^{h} \mu_{\sigma(j)} + \sum_{j=1}^{n'-h} \eta_{\sigma(h+j)} \tag{4}$$

where $\mu_{\sigma(j)}$ and $\eta_{\sigma(j)}$ denote the costs concerning $\mathsf{Multiply}_{\ell_{\sigma(j)}}$ and $\mathsf{Evaluate}_{\ell_{\sigma(j)}}$, respectively. The related cost $\kappa_{\sigma(j)}$ of $\mathsf{Construct}_{\ell_{\sigma(j)}}$ only impacts the cost for the chained computation required to get the output of $\mathsf{Compose}_L$, which gives a total cost equals to

$$\tau_L = \mathsf{C}\left(\mathsf{St}_{n'}^L\right) + \sum_{j=1}^{n'} \kappa_{\sigma(j)}.$$

Definition 3.4 (Multiplicative strategy). *A strategy* $\mathsf{St}_{n'}^L$ *of the form* $]\!]n' - 1[\![$ *is called a multiplicative strategy.*

Definition 3.5 (Evaluative strategy). *A strategy* $\mathsf{St}_{n'}^L$ *of the form* $[\![1]\!]^{n-1}$ *is called an evaluative strategy.*

Algorithm 1 Strategy for a CAB_k^L: Technique to perform $\mathsf{Compose}_L$

```
Inputs: A finite abelian group $G$ of order $N^m$ with $N = \ell_1 \cdots \ell_n$ and $m \geq 1$, an
    smooth factor $L = \ell_{\sigma(1)} \cdots \ell_{\sigma(n')}$ of $N$ for some integer $1 \leq n' \leq n$ and permutation
    $\sigma\colon [\![n]\!] \to [\![n]\!]$, an element $\mathbf{h} \coloneqq (\mathbf{h}_1, \dots, \mathbf{h}_k) \in G^k$ of order $(L, \dots, L)$ such that
    $\mathbf{H}_L \coloneqq \langle \mathbf{h}_1, \dots, \mathbf{h}_k \rangle \leq G$ has order $L^k$, a Chained Atomic Block $\mathsf{CAB}_k^L$ for $G$, and
    a strategy $\mathsf{St}_{n'}^L$ coded as a list of $n' - 1$ positive integers.
Output: Quotient subgroup $G' \coloneqq G/\mathbf{H}_L$.
 1: $k \leftarrow 1$
 2: $G' \leftarrow G$
 3: $\mathbf{h}' \leftarrow \mathbf{h}$
 4: $\mathcal{K} \leftarrow [\mathbf{h}']$
 5: $\mathsf{steps} \leftarrow [1]$
 6: for $i = 1$ to $n' - 1$ do
 7:     $\mathsf{level} \leftarrow$ sum of all elements in $\mathsf{steps}$
 8:     $\mathbf{h}' \leftarrow$ last element of $\mathcal{K}$
 9:     while $\mathbf{h}'$ does not have order $(\ell_{\sigma(n'+1-i)}, \dots, \ell_{\sigma(n'+1-i)})$ do
10:         $s_k \leftarrow k$-th element of $\mathsf{St}_{n'}^L$
11:         Append $s_k$ to the last element of $\mathsf{steps}$
12:         for $j = \mathsf{level}$ to $\mathsf{level} + s_k$ do
13:             $\mathbf{h}' \leftarrow \mathsf{Multiply}_{\ell_{\sigma(j)}}(\mathbf{h}')$
14:         end for
15:         Append $\mathbf{h}'$ to the last element of $\mathcal{K}$
16:         $\mathsf{level} \leftarrow \mathsf{level} + s_k$
17:         $k \leftarrow k + 1$
18:     end while
19:     assert $\mathsf{level} = n' - i$
20:     Remove the last element $\mathbf{h}'$ of $\mathcal{K}$
21:     Remove the last element of $\mathsf{steps}$
22:     $G' \leftarrow \mathsf{Construct}_{\ell_{\sigma(n'+1-i)}}(\mathbf{h}')$
23:     for $\mathbf{k}$ in $\mathcal{K}$ do
24:         $\mathbf{k} \leftarrow \mathsf{Evaluate}_{\ell_{\sigma(n'+1-i)}}(\mathbf{k}, \mathbf{h}')$
25:     end for
26: end for
27: Extract and remove the last element $\mathbf{h}'$ of $\mathcal{K}$
28: assert $\mathbf{h}'$ has order $(\ell_{\sigma(1)}, \dots, \ell_{\sigma(1)})$
29: $G' \leftarrow \mathsf{Construct}_{\ell_{\sigma(1)}}(\mathbf{h}')$
30: return $G'$
```

Definition 3.6 (Balanced strategy). *A strategy $\mathsf{St}_{n'}^L$ that recursively splits $\Delta_{n'}^L$ into two sub-triangles of the same size is called a balanced strategy.*

A multiplicative strategy performs a n' constructions, $n'-1$ evaluations, and a quadratic number of multiplications $\frac{(n'-1)n'}{2}$. While an evaluative strategy performs a n' constructions, $n'-1$ multiplications, and a quadratic number of evaluations $\frac{(n'-1)n'}{2}$. Conversely, a balanced strategy still performs n' constructions but $n' \log_2(n')$ multiplications and evaluations. Therefore, a balanced strategy requires fewer operations than any multiplicative (and evaluative) strategy.

Definition 3.7 (Optimal strategy). *A strategy* $\mathsf{St}_{n'}^{L}$ *with minimal associated cost* $\mathsf{C}\left(\mathsf{St}_{n'}^{L}\right)$ *is called an optimal strategy. That is, any other different strategy has an associated cost greater than* $\mathsf{C}\left(\mathsf{St}_{n'}^{L}\right)$*.*

As initially pointed out in [17] and extended in [26,10], the recursive nature of the strategies allows applying well-known dynamic-programming algorithms for computing optimal strategies. Similar to in [26,10], different orderings on the list L impact the cost $\mathsf{C}\left(\mathsf{St}_{n'}^{L}\right)$, implying a different optimal strategy per permutation. By assumption, we have that $\mu_{\sigma(i)} \geq \mu_{\sigma(j)}$ and $\eta_{\sigma(i)} > \eta_{\sigma(j)}$ whenever $\ell_{\sigma(i)} > \ell_{\sigma(j)}$ since the costs are polynomials concerning $\ell_{\sigma(i)}$ and $\ell_{\sigma(j)}$ (respectively). Consequently, the associated cost $\mathsf{C}\left(\mathsf{St}_{n'}^{L}\right)$ to the optimal ordering should be close to when $\ell_{\sigma(1)} > \ell_{\sigma(2)} > \cdots > \ell_{\sigma(n')}$ (i.e. when the strategy processes from the smallest to the largest $\ell_{\sigma(i)}$). The authors in [10] provided an algorithm that finds an optimal strategy in time $O(n'^3)$, which essentially extends to our case study. In a nutshell, the algorithm computes an optimal strategy $\mathsf{St}_{n'}^{L}$ by iteratively solving

$$\underset{h\in[\![i-1]\!]}{\arg\min} \left\{ \mathsf{C}\left(\mathsf{St}_{i-h}^{\mathcal{L}_{i-h}}\right) + \mathsf{C}\left(\mathsf{St}_{h}^{\mathcal{L}_{h}}\right) + \sum_{j=1}^{h} \mu_{\sigma(j)} + \sum_{j=1}^{i-h} \eta_{\sigma(h+j)} \right\}$$

for each $i \coloneqq 1, \ldots, n'$ and $k \coloneqq 1, \ldots (n'+1-i)$, where $\mathcal{L} = \ell_{\sigma(k)}\ell_{\sigma(k+1)}\cdots\ell_{\sigma(k+i)}$. We formalize and land the strategy-search procedure from [10] in Algorithm 2.

3.2 Optimal strategies: when L is a power of a small prime number

This section centers on the case when the subgroup order $L = \ell^{n'}$ is a power of a small prime ℓ; this case is much simpler than the general case from above. For instance, a $\mathsf{CAB}_{n'}^{\ell^{n'}}$ corresponds with the following tuple

$$\mathsf{CAB}_{k}^{\ell^{n'}} \coloneqq \left(\mathsf{QAB}_{k}^{\ell}, \mathsf{Compose}_{L}\right).$$

The search for optimal strategies relies on the simplified Algorithm 3, and the associated cost of a strategy $\mathsf{St}_{n'}^{\ell^{n'}}$ reduces to

$$\mathsf{C}\left(\mathsf{St}_{n'}^{\ell^{n'}}\right) = \mathsf{C}\left(\mathsf{St}_{n'-h}^{\ell^{n'-h}}\right) + \mathsf{C}\left(\mathsf{St}_{h}^{\ell^{h}}\right) + h\mu + (n'-h)\eta,$$

where μ and η denote the cost concerning $\mathsf{Multiply}_{\ell}$ and $\mathsf{Evaluate}_{\ell}$, respectively. Algorithm 4 present the strategy-based procedure to perform $\mathsf{Compose}_{\ell^{n'}}$.

Algorithm 2 Procedure to compute an optimal strategy for a CAB_k^L

```
Inputs: An integer $N = \ell_1 \cdots \ell_n$ and $m \geq 1$, an smooth factor $L = \ell_{\sigma(1)} \cdots \ell_{\sigma(n')}$ of
    $N$ for some integer $1 \leq n' \leq n$ and permutation $\sigma\colon [\![n]\!] \to [\![n]\!]$, and the associated
    costs $\mu_{\sigma(i)}$ and $\eta_{\sigma(i)}$ of a $\mathsf{CAB}_k^L$ for each $i \coloneqq 1, \ldots, n'$.
Output: Optimal strategy $\mathsf{St}_{n'}^L$.
 1: for $j = 1$ to $n'$ do
 2:     Set as optimal strategy $\mathsf{St}_1^{\mathcal{L}^{(j,1)}} = []$ for $\mathcal{L}^{(j,1)} = \ell_{\sigma(j)}$
 3: end for
 4: for $i = 2$ to $n'$ do
 5:     for $j = 1$ to $n' + 1 - i$ do
 6:         Set $\mathcal{L}^{(j,i)} \leftarrow \ell_{\sigma(j)} \cdots \ell_{\sigma(j+i)}$
 7:         Solve
              $$s = \arg\min_{h \in [\![i-1]\!]} \left\{ \mathsf{C}\left(\mathsf{St}_{i-h}^{\mathcal{L}^{(j+h,i-h)}}\right) + \mathsf{C}\left(\mathsf{St}_h^{\mathcal{L}^{(j,h)}}\right) + \sum_{l=j}^{j+h} \mu_{\sigma(l)} + \sum_{l=j}^{j+i-h} \eta_{\sigma(h+l)} \right\}$$
 8:         Compute as optimal strategy $\mathsf{St}_i^{\mathcal{L}^{(j,i)}} = [s] \cup \mathsf{St}_{i-s}^{\mathcal{L}^{(j+s,i-s)}} \cup \mathsf{St}_s^{\mathcal{L}^{(j,s)}}$
 9:     end for
10: end for
11: return $\mathsf{St}_{n'}^{\mathcal{L}^{(1,n')}}$
```

Algorithm 3 Procedure to compute an optimal strategy for a $\mathsf{CAB}_k^{\ell^{n'}}$

```
Inputs: An integer $N = \ell^z \cdot \ell_1 \cdots \ell_n$ and $m \geq 1$, an integer $1 \leq n' \leq z$, and the
    associated costs $\mu$ and $\eta$ of a $\mathsf{CAB}_k^L$.
Output: Optimal strategy $\mathsf{St}_{n'}^{\ell^{n'}}$.
1: Set as optimal strategy $\mathsf{St}_1^{\ell} = []$
2: for $i = 2$ to $n'$ do
3:     Solve
          $$s = \arg\min_{h \in [\![i-1]\!]} \left\{ \mathsf{C}\left(\mathsf{St}_{i-h}^{\ell^{(i-h)}}\right) + \mathsf{C}\left(\mathsf{St}_h^{\ell^h}\right) + h\mu + (i-h)\eta \right\}$$
4:     Compute as optimal strategy $\mathsf{St}_i^{\ell^i} = [s] \cup \mathsf{St}_{i-s}^{\mathcal{L}^{(j+s,i-s)}} \cup \mathsf{St}_s^{\mathcal{L}^{(j,s)}}$
5: end for
6: return $\mathsf{St}_{n'}^{\ell^{n'}}$
```

4 Applications

This section simplifies the strategy technique through Algorithm 5 for computing (ℓ^n, ℓ^n)-isogenies over superspecial PPAS. Corollary 4.1 describes the case study in terms of a CAB. In particular, to illustrate the practical implications, we next give an overview of the state-of-the-art for computing $(2,2)$-isogenies and $(3,3)$-isogenies over superspecial PPAS (the built-in functions concerning a

Algorithm 4 Strategy for a $\mathsf{CAB}_k^{\ell^{n'}}$: Technique to perform $\mathsf{Compose}_{\ell^{n'}}$

```
Inputs: A finite abelian group $G$ of order $N^m$ with $N = \ell^z \cdot \ell_1 \cdots \ell_n$ and $m \geq 1$, an
    integer $1 \leq n' \leq z$, an element $\mathbf{h} \coloneqq (\mathbf{h}_1, \ldots, \mathbf{h}_k) \in G^k$ of order $(\ell^{n'}, \ldots, \ell^{n'})$ such
    that $\mathbf{H}_{\ell^{n'}} \coloneqq \langle \mathbf{h}_1, \ldots, \mathbf{h}_k \rangle \leq G$ has order $\ell^{n'}$, a Chained Atomic Block $\mathsf{CAB}_k^{\ell^{n'}}$ for
    $G$, and a strategy $\mathsf{St}_{n'}^{\ell^{n'}}$ coded as a list of $n' - 1$ positive integers.
Output: Quotient subgroup $G' \coloneqq G/\mathbf{H}_{\ell^{n'}}$.
 1: $k \leftarrow 1$
 2: $G' \leftarrow G$
 3: $\mathbf{h}' \leftarrow \mathbf{h}$
 4: $\mathcal{K} \leftarrow [\mathbf{h}']$
 5: for $i = 1$ to $n' - 1$ do
 6:    $\mathsf{level} \leftarrow$ sum of all elements in $\mathsf{steps}$
 7:    $\mathbf{h}' \leftarrow$ last element of $\mathcal{K}$
 8:    while $\mathbf{h}'$ does not have order $(\ell, \ldots, \ell)$ do
 9:       $s_k \leftarrow k$-th element of $\mathsf{St}_{n'}^{L}$
10:       $\mathbf{h}' \leftarrow \underbrace{\mathsf{Multiply}_\ell(\ldots(\mathbf{h}')\ldots)}_{s_k \text{ multiplications}}$
11:       Append $\mathbf{h}'$ to the last element of $\mathcal{K}$
12:       $k \leftarrow k + 1$
13:    end while
14:    assert $\mathbf{h}'$ has order $(\ell, \ldots, \ell)$
15:    Remove the last element $\mathbf{h}'$ of $\mathcal{K}$
16:    $G' \leftarrow \mathsf{Construct}_\ell(\mathbf{h}')$
17:    for $\mathbf{k}$ in $\mathcal{K}$ do
18:       $\mathbf{k} \leftarrow \mathsf{Evaluate}_\ell(\mathbf{k}, \mathbf{h}')$
19:    end for
20: end for
21: Extract and remove the last element $\mathbf{h}'$ of $\mathcal{K}$
22: assert $\mathbf{h}'$ has order $(\ell, \ldots, \ell)$
23: $G' \leftarrow \mathsf{Construct}_\ell(\mathbf{h}')$
24: return $G'$
```

$\mathsf{CAB}_2^{\ell^n}$) and present experiments according to our [SageMath/Magma]-language implementation of Algorithm 5.

Corollary 4.1 (Superspecial PPAS). *Given a superspecial PPAS $\mathcal{A}$ defined over $\mathbb{F}_{p^2}$ and a (ℓ^n, ℓ^n)-subgroup H of order ℓ^{2n}, generated by two order-ℓ^n elements, on $\mathcal{A}$. The task of computing the (ℓ^n, ℓ^n)-isogeny $\phi\colon \mathcal{A} \to \mathcal{A}/H$ with kernel H is a special case of a $\mathsf{CAB}_2^{\ell^n}$.*

4.1 Computing $(3^n, 3^n)$-isogenies

This section summarizes the $(3, 3)$-isogenies formulas by Bruin, Flynn and Testa [4]. Consider a maximal isotropic group $\langle T_1, T_2 \rangle \subset \mathsf{J}_C[3]$ of a genus-two curve C given by Equation (2). In [4], the authors provide a parametrization

Algorithm 5 Strategy for a $\mathsf{CAB}_2^{\ell^n}$: Technique to construct (ℓ^n, ℓ^n)-isogenies

```
Inputs: A superspecial PPAS $\mathcal{A}$, a kernel $\langle \mathbf{h}_1, \mathbf{h}_2 \rangle \cong \mathbb{Z}_{\ell^n} \times \mathbb{Z}_{\ell^n}$ on $\mathcal{A}$, and a strategy
    $\mathsf{St}_n^{\ell^n}$ coded as a list of $n-1$ positive integers.
Output: Codomain $\mathcal{A}' := \mathcal{A}/\langle \mathbf{h}_1, \mathbf{h}_2 \rangle$ of the $(\ell^n, \ell^n)$-isogeny with kernel $\langle \mathbf{h}_1, \mathbf{h}_2 \rangle$.
 1: $k \leftarrow 1$
 2: $\mathcal{A}' \leftarrow \mathcal{A}$
 3: $\mathbf{h}' \leftarrow (\mathbf{h}_1, \mathbf{h}_2)$
 4: $\mathcal{K} \leftarrow [\mathbf{h}']$
 5: for $i = 1$ to $n-1$ do
 6:     $\mathsf{level}$ ← sum of all elements in $\mathsf{steps}$
 7:     $\mathbf{h}'$ ← last element of $\mathcal{K}$
 8:     while $\mathbf{h}'$ does not have order $(\ell, \ell)$ do
 9:         $s_k \leftarrow k$-th element of $\mathsf{St}_{n'}^{L}$
10:         $\mathbf{h}' \leftarrow ([\ell^{s_k}]\mathbf{h}'_1, [\ell^{s_k}]\mathbf{h}'_2)$
11:         Append $\mathbf{h}'$ to the last element of $\mathcal{K}$
12:         $k \leftarrow k+1$
13:     end while
14:     assert $\mathbf{h}'$ has order $(\ell, \ell)$
15:     Remove the last element $\mathbf{h}'$ of $\mathcal{K}$
16:     $\mathcal{A}'$ ← codomain of the $(\ell, \ell)$-isogeny $\phi$ with kernel $\langle \mathbf{h}'_1, \mathbf{h}'_2 \rangle$
17:     for $\mathbf{k}$ in $\mathcal{K}$ do
18:         $\mathbf{k} \leftarrow (\phi(\mathbf{k}_1), \phi(\mathbf{k}_2))$
19:     end for
20: end for
21: Extract and remove the last element $\mathbf{h}'$ of $\mathcal{K}$
22: assert $\mathbf{h}'$ has order $(\ell, \ell)$
23: $\mathcal{A}'$ ← codomain of the $(\ell, \ell)$-isogeny $\phi$ with kernel $\langle \mathbf{h}'_1, \mathbf{h}'_2 \rangle$
24: return $\mathcal{A}'$
```

of the genus-two curve C determined by the 3-tuple (C, T_1, T_2), namely (r, s, t)-parametrization. In particular, they show that the curve C is isomorphic to

$$C_{rst} \colon y^2 = F_{rst}(x) = G_1(x)^2 + \lambda_1 H_1(x)^3 = G_2(x)^2 + \lambda_2 H_2(x)^3,$$

where

$$\begin{aligned}
H_1 &= x^2 + rx + t,\\
\lambda_1 &= 4s,\\
G_1 &= (s - st - 1)x^3 + 3s(r-t)x^2 + 3sr(r-t)x - st^2 + sr^3 + t,\\
H_2 &= x^2 + x + r,\\
\lambda_2 &= 4st, \quad \text{and}\\
G_2 &= (s - st + 1)x^3 + 3s(r-t)x^2 + 3sr(r-t)x - st^2 + sr^3 - t.
\end{aligned}$$

Additionally, the order-3 element T_i coincides with $(H_i(x), G_i(x))$ for each $i \in \{1, 2\}$. The authors in [4] suggest working with the associated Kummer surface

$K \coloneqq \mathsf{J}_C/\langle -1\rangle$ instead of the Jacobian J_C. They propose mapping divisor from J_C to K by some relation $\xi\colon D \mapsto (\xi_0 : \xi_1, : \xi_2, : \xi_3)$. More precisely, if $f = f_6x^6 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$, and $D \in \mathsf{J}_C$ is equal to $[(x_1, y_1) + (x_2, y_2)]$, then

$$\xi_0 = 1, \quad \xi_1 = x_1 + x_2, \quad \xi_2 = x_1x_2, \quad \xi_3 = \frac{\Phi(\xi_0, \xi_1, \xi_2) - 2y_1y_2}{\xi_1^2 - 4\xi_0\xi_2}$$

where

$$\Phi(\xi_0, \xi_1, \xi_2) = 2f_0\xi_0^3 + f_1\xi_0^2\xi_1 + 2f_2\xi_0^2\xi_2 + f_3\xi_0\xi_1\xi_2 + 2f_4\xi_0\xi_2^2 + f_5\xi_2^2\xi_1 + 2f_6\xi_2^3.$$

The Kummer surface K admits the following quartic equation model

$$K\colon (\xi_1^2 - 4\xi_0\xi_2)\xi_3^2 + \Phi(\xi_0, \xi_1, \xi_2)\xi_3 + \Psi(\xi_0, \xi_1, \xi_2) = 0,$$

where $\Psi(\xi_0, \xi_1, \xi_2)$ is a homogeneous degree-4 polynomial. The isogeny $\phi\colon \mathsf{J}_{C_{rst}} \to \mathsf{J}_{C_{r's't'}} \coloneqq \mathsf{J}_{C_{rst}}/\langle T_1, T_2\rangle$ induces an isogeny between the Kummer surfaces $K_{C_{rst}}$ and $K_{C_{r's't'}}$. The authors provide explicit formulas for computing the codomain curve and the induced map.

The authors in [20] recently improved formulas for $(3, 3)$-isogenies. They simplify formulas and reduce the number of required multiplications in [4]. The authors use a Gröbner basis approach [20,5], to compute the coordinate transformation to a given (r, s, t)-parametrization that allows us to apply the isogeny formulas. They also provide explicit formulas for the induced transformation on the Kummer surface.

Strategies for computing $(3^n, 3^n)$-isogenies. In the recent work [20], the authors mention the “impossibility” of reusing the optimal strategies from [17]. Additionally, [20] provides a public code and claims to use a balanced strategy technique as in [17]. We use their code and implement Algorithm 5 in the context of $(3, 3)$-isogenies. Our implementation allows us to test different kinds of strategies. In particular, we compare our strategy technique with the given in [20]. First, we compute the balanced strategy as suggested in [17], and we notice such a strategy differs from the approach in [20]. Then to identify the main difference, we include counters for the number of multiplications by three and $(3, 3)$-isogeny evaluations. Table 1 lists those operation numbers concerning different strategy techniques (balanced and optimal balanced) and compares them against the algorithm from [20]. It is worth highlighting that optimal strategies also depend on the cost μ (multiplication by 3) and η ($(3, 3)$-isogeny evaluation). Indeed, regarding asymptotical runtime, the best option is determined by μ and η as the number of field operations (commonly, the number of field multiplications and excluding additions). However, in practice and for dedicated [optimized] C-language implementations, such values as μ and η determine clock cycles or milliseconds [2]. Anyhow, our experiments compares [20] against the following two different strategies:

1. Balanced strategy just as suggested in [20] but employing Algorithm 5; and
2. Optimal balanced strategy calculated as in Algorithm 3 with $\mu = \eta$ and using Algorithm 5.

Technique	#[Multiplications by 3]	#[(3, 3)-isogeny evaluations]	Runtime
Balanced strategy from [20]	2884	2380	5264
Balanced strategy	1936	2290	4226
Optimal balanced strategy	1818	2408	4226

Table 1: Number of multiplications by three and $(3, 3)$-isogeny evaluations required to compute a $(3^{236}, 3^{236})$-isogeny, the runtime column corresponds with the sum of both numbers. The field characteristic is p751 as defined in [2]. All the experiments assume the same number of extra points to be evaluated under each $(3, 3)$-isogeny (just as required for attacking SIKEp751).

From Table 1, we expect our implementation of Algorithm 5 to be 1.25x faster than [20], which is about 20% of savings. We discuss and analyze the impact in seconds of our strategy technique below.

Experimental results. To illustrate the impact of our results, we point out that our results directly apply to the attacks in [6,28,30]. For example, the most demanded computations in the Castryck-Decru attack are the $(3^i, 3^i)$-isogenies for some integer $i \in [\![n]\!]$ close to n. We additionally plug our Algorithm 5 into the public Magma language code of [20] and draw our results in Figure 2. Our experiments focus on the quadratic field extensions of $\mathbb{F}_{p^2}$ with prime characteristic p751 as defined in [2]. Our implementation isolates the calls to $\mathsf{Points}(\mathsf{J}, \mathsf{h})[1]$, which corresponds with the map sending points $\mathbf{h}$ from the Kummer Surface into the Jacobian. Consequently, $\mathsf{Points}(\mathsf{J}, \mathsf{h})[1]$ only plays a role when computing the codomain of the isogeny. Therefore, such a cost is not required for computing the optimal strategy.

We notice from the experiments that the bottleneck in the current implementations in [20] and ours is the calculation of the codomain curve along with the data required for evaluating the $(3, 3)$-isogeny [7], which takes on average 0.04 seconds [8]. Both methods perform exactly 236 use of $\mathsf{Points}(\mathsf{J}, \mathsf{h})[1]$, which gives 9.44 seconds (about 89.06% of the total running time [in average] of 10.6). For instance, according to the discussion in Section 4.1, we expect a 1.25x speedup in the $(3^n, 3^n)$-isogeny computation, giving a runtime of $1.15964/1.25 = 0.927712$ seconds instead of 1.15964 seconds (the 1.15964% of 10.6). Overall, the expected running time would be $(0.927712 + 9.44) = 10.367712$ seconds on average, and our experiments from Figure 2 illustrate such savings.

[7] We highlight that the data required for evaluating the $(3, 3)$-isogenies are only computed once and, thus we can view such computations as part of the calculation of the codomain curve.

[8] We include the cost concerning $\mathsf{Points}(\mathsf{J}, \mathsf{h})[1]$.

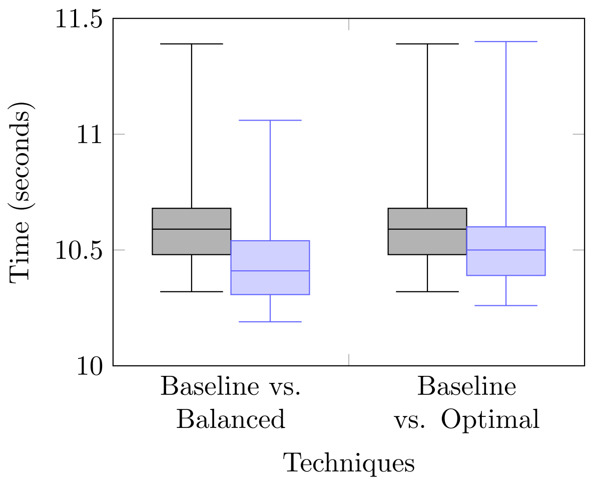

Fig. 2: Our experiments were executed on a 2.3 GHz 8-Core Intel Core i9 machine with 16GB of RAM. The measures correspond with the average of 100 random instances and determine seconds. The data in blue ink correspond with this work, while the gray ink is the baseline code from [20]. The field characteristic is p751 as defined in [2].

Consequently, any improvement in computing the codomain curve along with the calculation of the data required for evaluating the $(3,3)$-isogeny [78] should speed up the $(3^n,3^n)$-isogeny computation and make the optimal strategies the most efficient technique (about 1.25x faster).

4.2 Computing $(2^n, 2^n)$-isogenies

This section summarizes how to compute codomains of $(2,2)$-isogenies and push points through $(2,2)$-isogenies. For simplicity, we swap (when needed) between Mumford's representation and formal sums representations to land the general idea behind $(2,2)$-isogenies. We suggest reading [7,27] for a better understanding.

Consider a genus two curve C determined Equation (2), and let us assume $f(x) = F_1(x)F_2(x)F_3(x)$, where $F_t(x) = g_{t2}x^2 + g_{t1}x + g_{t0}$ for each $i := 1,2,3$, such that $G = \langle (F_1(x),0),(F_2(x),0)\rangle = \{\mathcal{O},(F_1(x),0),(F_2(x),0),(F_3(x),0)\}$ is a maximal isotropic group. Let

$$\delta := \det \begin{bmatrix} g_{10} & g_{11} & g_{12} \\ g_{20} & g_{21} & g_{22} \\ g_{30} & g_{31} & g_{32} \end{bmatrix}.$$

Then, the codomain curve C/G of the $(2,2)$-isogeny $\phi\colon \mathsf{J}_C \to \mathsf{J}_{C/G}$ is isomorphic to

$$C' \colon y^2 = H_1(x)H_2(x)H_3(x)$$

where

$$H_i(x) = \delta^{-1}\left(F_j'(x)F_k(x) - F_k'(x)F_j(x)\right)$$

with (ijk) a cyclic permutation of 1,2,3. On the other hand, evaluating element $D \in \mathsf{J}_C$ through ϕ summarize as follows.

1. Decompose $D \in \mathsf{J}_C$ as $D = [P + Q]$ where $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are points on the curve C. The goal is to find point P', Q', P'', Q'' such that $\phi(D) = [P' + P''] + [Q' + Q'']$.
2. The abscissas of P' and P'' are obtained by solving the quadratic equation in x_2:
$$F_1(x_P)H_1(x_2) + F_2(x_P)G_2(x_2) = 0,$$
and the ordinate by solving in y_2
$$y_p y_2 = F_1(x_P)H_1(x_{P'})(x_P - x_{P'}).$$
3. Repeat the same for Q', Q''.
4. Calculate $\phi(D) = [P' + P''] + [Q' + Q'']$.

The author in [27] presents explicit formulas for pushing points through $(2,2)$-isogenies with a kernel of the form $G = \langle (x, 0), (x^2 - Ax + 1, 0)\rangle$. The author characterizes the family of curves determined by

$$C \colon y^2 = Ex(x^2 - Ax + 1)(x^2 - Bx + C)$$

and proves that any genus-two curves can be transformed into such a shape [9].
and proves that any genus-two curves can be transformed into such a shape [10].
In particular, any order-$(2,2)$ subgroup over J_C maps into a suitable G.

Strategies for computing $(2^n, 2^n)$-isogenies. The technique from [27] [11] suggests splitting the isogeny computation into m isogeny chunks of $(2^{k_i}, 2^{k_i})$-isogenies ϕ_i's with $\sum_{i=1}^{m} k_i = n$. The author in [27] manages to reduce the running time in their approach from $O(n^2)$ to $O(n\sqrt{n})$. Indeed, the technique from [27] falls into our strategy definition and relies on a multiplicative-like nature. However, the latest code version from [27] includes the same balanced strategy technique as in [20]. Therefore, we compare our implementation of Algorithm 5 against [27] regarding the number of multiplications by two and $(2,2)$-isogeny evaluations, along with the running time in seconds (we add counters in both codes). All our experiments use the balanced strategy and the parameters with a 171-bit prime proposed in [27]. Our code implementation is about 1.3x faster than [27] (see Tables 2 and 3).

[9] The isomorphism cuold

[10] The isomorphism could be defined over a quartic field extension of $\mathbb{F}_p$.

[11] For more details, we recommend reading [27, Section 5.3].

Technique	#[Multiplications by 2]	#[$(2,2)$-isogeny evaluations]	Runtime
$(2^n, 2^n)$-isogeny with 4 evaluations of extra points			
Balanced strategy from [27]	1033	874	1907
Balanced strategy	768	874	1642
$(2^n, 2^n)$-isogeny (only codomain curve calculation)			
Balanced strategy from [27]	1033	526	1559
Balanced strategy	768	526	1294

Table 2: Number of multiplications by three and $(2,2)$-isogeny evaluations required to compute a $(2^{87}, 2^{87})$-isogeny, the runtime column correspond with the sum of both numbers. The field characteristic is p171 as defined in [27].

Procedure	Baseline [27]	This work	Speedup
$(2^n, 2^n)$-isogeny with 4 evaluations of extra points	0.1779	0.1336	1.332x
$(2^n, 2^n)$-isogeny (only codomain curve calculation)	0.1659	0.1229	1.335x

Table 3: Our experiments were executed on a 2.3 GHz 8-Core Intel Core i9 machine with 16GB of RAM. The measures correspond with the average of 100 random instances and determine seconds. Number of multiplications by three and $(2,2)$-isogeny evaluations required to compute a $(2^{87}, 2^{87})$-isogeny. The field characteristic is p171 as defined in [27].

Experimental results. To illustrate the impact of our results, we point out that our results directly apply to the attacks in [6,28,30]. For example, the most demanded computations in the Castryck-Decru attack are the $(2^i, 2^i)$-isogenies for each $i \in [\![n]\!]$. However, [29] shows that it is enough to compute a few $(2^i, 2^i)$-isogenies for some integer $i \in [\![n]\!]$ close to n; such a shortcut splits the computations into two parts: the $(2^i, 2^i)$-isogeny computation and some discrete logarithm computations. In any case, the isogenies still play an essential role in the Castryck-Decru attack, and at most, we expect a speedup of 1.3x when using the strategy technique.

We plug our Algorithm 5 into the public SageMath language code from [29] and draw our results in Figure 3. Our experiments focus on the quadratic field extensions of $\mathbb{F}_{p^2}$ with prime characteristic pXXX for each XXX $\in \{182, 217, 434\}$ as defined in [13,2]. In particular, our experiments show a speedup of 1.19x—1.26x in the Castryck-Decru attack (see Table 4).

5 Forthcoming research

In summary, strategies give a modest speed-up of about 1.25x compared to the state-of-the-art for computing $(2^n, 2^n)$-isogenies (resp. $(3^n, 3^n)$-isogenies). Even after the wave of attacks in [6,28,30], the analyzed strategy technique can help to build "secure" Proof-of-Knowledge (PoK) protocols as in [17,16,3,9,25] efficiently but extended to isogenies over superspecial PPAS. The reason for moving to

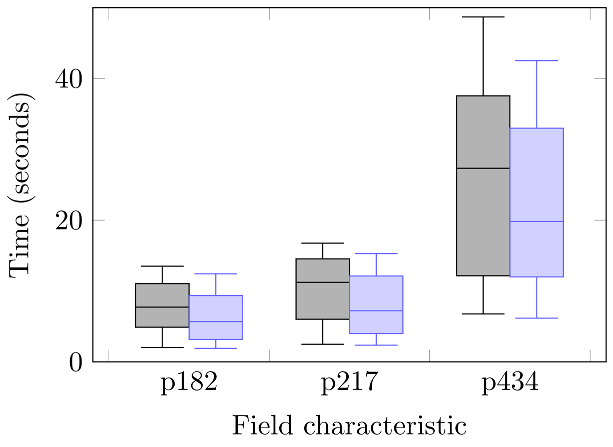

Fig. 3: Our experiments were executed on a 2.3 GHz 8-Core Intel Core i9 machine with 16GB of RAM. The measures correspond with the average of 100 random instances and determine seconds. The data in blue ink correspond with this work, while the gray ink is the baseline code from [29].

Field characteristic	Baseline [29]	This work	Speedup
p182	7.90	6.30	1.25x
p217	10.41	8.25	1.26x
p434	26.90	22.67	1.19x

Table 4: Our experiments were executed on a 2.3 GHz 8-Core Intel Core i9 machine with 16GB of RAM. The measures correspond with 100 random instances and determine seconds.

superspecial PPAS is that the best algorithm to find isogenies has a running time equal to $\widetilde{O}(p)$ [14], and therefore one can use prime fields of [128/192/256]-bits instead of [434/610/751]-bits. However, further analysis is required to decide the efficiency of superspecial PPAS-based PoK.

Another cryptanalytic application of the strategies technique is still on the attacks from [6,28,30], but when computing (L, L)-isogenies (i.e. for attacking the construction from [12]). This time, L is a product of small primes and differs from a power of a small prime number. Therefore, Algorithm 1 should perform better than the naive multiplicative-based strategy.

Lastly, the presented strategy techniques also apply to the recent work [15] that discusses strategies for computing higher dimensional isogeny. More precisely, Algorithm 4 induces an efficient algorithm to perform the KernelToIsogeny procedure from [15].

Acknowledgements. We thank Sabrina Kunzweiler and Thomas Decru for their comments about an early version of this manuscript and regarding the Magma code from [20].

References

1. Adj, G., Chi-Domínguez, J., Rodríguez-Henríquez, F.: Karatsuba-based square-root Vélu's formulas applied to two isogeny-based protocols. J. Cryptogr. Eng. (2022). https://doi.org/10.1007/s13389-022-00293-y
2. Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Jalali, A., Jao, D., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Pereira, G., Renes, J., Soukharev, V., Urbanik, D.: Supersingular Isogeny Key Encapsulation. Third Round Candidate of the NIST's post-quantum cryptography standardization process (2020), available at: https://sike.org/
3. Basso, A., Codogni, G., Connolly, D., De Feo, L., Fouotsa, T.B., Lido, G.M., Morrison, T., Panny, L., Patranabis, S., Wesolowski, B.: Supersingular Curves You Can Trust. IACR Cryptol. ePrint Arch. p. 1469 (2022), https://eprint.iacr.org/2022/1469, to Appear in EUROCRYPT 2023
4. Bruin, N., Flynn, E., Testa, D.: Descent via (3,3)-isogeny on jacobians of genus 2 curves. Acta Arithmetica **165** (01 2014). https://doi.org/10.4064/aa165-3-1
5. Castryck, W., Decru, T.: Multiradical isogenies. IACR Cryptology ePrint Archive **2021**, 1133 (2021), https://eprint.iacr.org/2021/1133
6. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. IACR Cryptol. ePrint Arch. p. 975 (2022), https://eprint.iacr.org/2022/975, to Appear in EUROCRYPT 2023
7. Castryck, W., Decru, T., Smith, B.: Hash functions from superspecial genus-2 curves using Richelot isogenies. J. Math. Cryptol. **14**(1), 268–292 (2020). https://doi.org/10.1515/jmc-2019-0021
8. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: Csidh: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018. pp. 395–427. Springer International Publishing, Cham (2018)
9. Chi-Domínguez, J.: A Note on Constructing SIDH-PoK-based Signatures after Castryck-Decru Attack. IACR Cryptol. ePrint Arch. p. 1479 (2022), https://eprint.iacr.org/2022/1479
10. Chi-Domínguez, J., Rodríguez-Henríquez, F.: Optimal strategies for CSIDH. Adv. Math. Commun. **16**(2), 383–411 (2022). https://doi.org/10.3934/amc.2020116
11. Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of Elliptic and Hyperelliptic Curve Cryptography, Second Edition. Chapman &amp; Hall/CRC, 2nd edn. (2012)
12. Costello, C.: B-SIDH: Supersingular Isogeny Diffie-Hellman Using Twisted Torsion. In: Moriai, S., Wang, H. (eds.) 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. Lecture Notes in Computer Science, vol. 12492, pp. 440–463. Springer (2020). https://doi.org/10.1007/978-3-030-64834-3_15
13. Costello, C.: The Case for SIKE: A Decade of the Supersingular Isogeny Problem. IACR Cryptol. ePrint Arch. p. 543 (2021), https://eprint.iacr.org/2021/543

14. Costello, C., Smith, B.: The Supersingular Isogeny Problem in Genus 2 and Beyond. In: Ding, J., Tillich, J. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12100, pp. 151–168. Springer (2020). https://doi.org/10.1007/978-3-030-44223-1_9
15. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: SQISignHD: New Dimensions in Cryptography. IACR Cryptol. ePrint Arch. p. 436 (2023), https://eprint.iacr.org/2023/436
16. De Feo, L., Dobson, S., Galbraith, S.D., Zobernig, L.: SIDH Proof of Knowledge. In: Agrawal, S., Lin, D. (eds.) 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. Lecture Notes in Computer Science, vol. 13792, pp. 310–339. Springer (2022). https://doi.org/10.1007/978-3-031-22966-4_11, https://doi.org/10.1007/978-3-031-22966-4_11
17. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Math. Cryptol. **8**(3), 209–247 (2014). https://doi.org/10.1515/jmc-2012-0015
18. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In: Moriai, S., Wang, H. (eds.) 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. Lecture Notes in Computer Science, vol. 12491, pp. 64–93. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_3
19. De Feo, L., Leroux, A., Longa, P., Wesolowski, B.: New algorithms for the Deuring correspondence: SQISign twice as fast. IACR Cryptol. ePrint Arch. p. 234 (2022), https://eprint.iacr.org/2022/234, to Appear in EUROCRYPT 2023
20. Decru, T., Kunzweiler, S.: Efficient Computation of $(3^n, 3^n)$-isogenies. IACR Cryptol. ePrint Arch. p. 376 (2023), https://eprint.iacr.org/2023/376
21. Elkhatib, R., Koziel, B., Azarderakhsh, R.: Faster Isogenies for Post-quantum Cryptography: SIKE. In: Galbraith, S.D. (ed.) Topics in Cryptology - CT-RSA 2022 - Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1-2, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13161, pp. 49–72. Springer (2022). https://doi.org/10.1007/978-3-030-95312-6_3
22. Florit, E., Smith, B.: An atlas of the Richelot isogeny graph. IACR Cryptol. ePrint Arch. p. 13 (2021), https://eprint.iacr.org/2021/013
23. Flynn, E.V., Ti, Y.B.: Genus two isogeny cryptography. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 286–306. Springer International Publishing, Cham (2019)
24. Flynn, E.V.: The jacobian and formal group of a curve of genus 2 over an arbitrary ground field. Mathematical Proceedings of the Cambridge Philosophical Society **107**(3), 425–441 (1990). https://doi.org/10.1017/S0305004100068729
25. Ghantous, W., Pintore, F., Veroni, M.: Efficiency of SIDH-based signatures (yes, SIDH). IACR Cryptol. ePrint Arch. p. 433 (2023), https://eprint.iacr.org/2023/433
26. Hutchinson, A., LeGrow, J.T., Koziel, B., Azarderakhsh, R.: Further Optimizations of CSIDH: A Systematic Approach to Efficient Strategies, Permutations, and Bound Vectors. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part I.

Lecture Notes in Computer Science, vol. 12146, pp. 481–501. Springer (2020). https://doi.org/10.1007/978-3-030-57808-4_24
27. Kunzweiler, S.: Efficient Computation of $(2^{\overline{n}}, 2^n)$-Isogenies. IACR Cryptol. ePrint Arch. p. 990 (2022), https://eprint.iacr.org/2022/990
28. Maino, L., Martindale, C.: An attack on SIDH with arbitrary starting curve. IACR Cryptol. ePrint Arch. p. 1026 (2022), https://eprint.iacr.org/2022/1026, to Appear in EUROCRYPT 2023
29. Oudompheng, R., Pope, G.: A Note on Reimplementing the Castryck-Decru Attack and Lessons Learned for SageMath. IACR Cryptol. ePrint Arch. p. 1283 (2022), https://eprint.iacr.org/2022/1283
30. Robert, D.: Breaking SIDH in polynomial time. IACR Cryptol. ePrint Arch. p. 1038 (2022), https://eprint.iacr.org/2022/1038, to Appear in EUROCRYPT 2023