Computing Isogenies of Power-Smooth Degrees Between PPAVs

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

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

Abstract. The wave of attacks by Castryck and Decru (Eurocrypt, 2023), Maino, Martindale, Panny, Pope and Wesolowski (Eurocrypt, 2023) and Robert (Eurocrypt, 2023), highlight the destructive facet of calculating power-smooth degree isogenies between higher-dimensional abelian varieties in isogeny-based cryptography. Despite those recent attacks, there is still interest in using isogenies but for building protocols on top of higher-dimensional abelian varieties. Examples of such protocols are Public-Key Encryption, Key Encapsulation Mechanism, Verifiable Delay Function, Verifiable Random Function, and Digital Signatures.
This work abstracts and proposes a generalization of the strategy technique by Jao, De Feo and Plût (Journal of Mathematical Cryptology, 2014) to give an efficient generic algorithm for computing isogenies between higher-dimensional abelian varieties with kernels being maximal isotropic of power-smooth degree.
To illustrate the impact of using such strategy technique, we draft our experiments on the computation of isogenies over two-dimensional abelian varieties determined by a maximal isotropic subgroup of torsion with a power of two or three. Our experiments illustrate a speed-up of 1.25x faster than the state-of-the-art (about 20% of savings).

Keywords: Higher-Dimensional Abelian Varieties · Isogenies · Maximal Isotropic Subgroups · Strategies

1 Introduction

The devastating attacks on SIDH, started by Castryck and Decru [7] and subsequently improved by Maino, Martindale, Panny, Pope and Wesolowski [24] and Robert [30], have as the most demanding calculations the isogenies of power-smooth degree between higher-dimensional abelian varieties. The key ingredient of those attacks is the Kani's theorem, which connects isogenies between supersingular curves and isogenies between product of curves (passing through Jacobian of genus-two curves). In fact, Kani's theorem plays an interesting role for buinding protocols on top of higher-dimensional abelian varieties.

Decru and Kunzweiler [15] described a genus-two hash function based on the Charles-Goren-Lauter hash function by employing kernels generators of torsion 3^n. Dartois, Leroux, Robert and Wesolowski [13] proposed a higher-dimensional SQISign construction, namely SQISignHD, to reduce sizes. Basso, Maino and Pope [3] presented an efficient isogeny-based Public Key Encryption, called FESTA, based on a trapdoor function that uses some improved techniques analyzed in the SIDH attacks. Subsequently, Nakagawa and Onuki [28] described QFESTA as a Quaternion variant of FESTA with one-third of key and ciphertext sizes than the original FESTA proposal. Decru, Maino and Sanso [16] detailed a weak Verifiable Delay Function with delay-based computation on large-degree isogeny between elliptic curves and verification on the computation of isogenies between products of elliptic curves. Leroux [21] suggested a Verifiable Random Function that requires isogenies over higher-dimensional varieties. Moriya [26] recently proposed a Key Encapsulation Mechanism, called IS-CUBE, that requires isogenies between the product of elliptic curves.

It is worth highlighting that all the above constructions over higher-dimensional abelian varieties require kernel generators either of torsion 2^n or 3^n.

Our contributions. We wholly center on the task of computing separable $(\ell^n, \ldots, \ell^n)$-isogenies from $(\ell^n, \ldots, \ell^n)$-subgroups. In particular, we focus on the scenario where ϕ splits as the composition of n $(\ell, \ldots, \ell)$-isogenies. Moreover, we extend and formalize the strategies for calculating isogenies of power-smooth degree between supersingular elliptic curves to the higher-dimensional PPAVs context [4]. In a nutshell, we give a polynomial-time algorithm for performing the two below tasks efficiently:

- Calculate the codomain of $(\ell^n, \ldots, \ell^n)$-isogenies.
- Push points through $(\ell^n, \ldots, \ell^n)$-isogenies.

We additionally provide two proof-of-concept implementations using the Magma Computer Algebra System and the SageMath library. Our experiments land on $(2^n, 2^n)$-isogenies and $(3^n, 3^n)$-isogenies between PPAVs of dimension two. Our local experiments illustrate a speed-up of about 1.25x compared with state-of-the-art techniques [5]. Additionally, our SageMath code is currently used in the implementation of FESTA [3] and implicitly integrated into QFESTA [28] and IS-CUBE [26].

2 Preliminaries

This section gives an overview description concerning principal polarized abelian varieties of dimension $g \geq 1$ and isogenies between them. For a deeper understanding, we suggest reading [12, 25, 27].

[4] PPAVs stands for principally polarized abelian varieties

[5] Our code is freely available at this GitHub repository

An abelian variety is a smooth projective algebraic variety which is an algebraic group. The dual abelian variety of an abelian variety $\mathcal{A}$ is denoted by $\widehat{\mathcal{A}}$ and is isomorphic to the group $\mathrm{Pic}^0(\mathcal{A})$ of divisor classes of degree zero on $\mathcal{A}$.

We use the additive group law notation for clarity when operating points on the abelian varieties. We denote the neutral element in $\mathcal{A}$ by $\mathbf{0}_{\mathcal{A}}$, and the ℓ-torsion subgroup $\{P \in \mathcal{A} \mid [\ell]P = \mathbf{0}_{\mathcal{A}}\}$ of $\mathcal{A}$ by $\mathcal{A}[\ell]$ where

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

An isogeny between abelian varieties is a surjective morphism $\phi\colon \mathcal{A} \to \mathcal{A}'$ with a finite kernel such that $\phi(\mathbf{0}_{\mathcal{A}}) = \mathbf{0}_{\mathcal{A}'}$. An ample divisor of $\mathcal{A}$ defines an isogeny $\lambda\colon \mathcal{A} \to \widehat{\mathcal{A}}$, which is called a polarization of $\mathcal{A}$. A polarization is principal if it is an isomorphism. We call $\mathcal{A}$ a principally polarized abelian variety (PPAV) if $\mathcal{A}$ is endowed with a principal polarization λ. See [25, Page 53] for more details.

Theorem 2.1. *([27, Page 72, Theorem 4]) Let $\mathcal{A}$ be an abelian variety. There is a 1-1 correspondence between*

1. *finite subgroups $\mathcal{G}$ of $\mathcal{A}$ and*
2. *separable isogenies $\phi\colon \mathcal{A} \to Y$.*

Two isogenies $\phi_1\colon \mathcal{A} \to Y_1$ and $\phi_2\colon \mathcal{A} \to Y_2$ with $\ker\phi_1 = \ker\phi_2 = \mathcal{G}$ are equal if there is an isomorphism $\iota\colon Y_1 \to Y_2$ such that $\phi_2 = \iota \circ \phi_1$. In other words, separable isogenies between PPAVs are uniquely determined by their kernels.

Definition 2.1. *Let p be a prime integer. Let $\mathcal{A}/\overline{\mathbb{F}}_p$ be a PPAV and ℓ be an integer relatively prime to p. The m-Weil pairing is a nondegenerate, skew-symmetric, bilinear, and alternating form*

$$e_m : \mathcal{A}[m](\overline{\mathbb{F}}_p) \times \mathcal{A}[m](\overline{\mathbb{F}}_p) \to \mu_m,$$

where μ_m is the group of m-th roots of unity.

Definition 2.2. *Let p be a prime integer. Let $\mathcal{A}/\overline{\mathbb{F}}_p$ be a PPAV and ℓ be an integer relatively prime to p. A proper subgroup $\mathcal{G}$ of $\mathcal{A}[\ell]$ is a maximal ℓ-isotropic subgroup if*

1. *the ℓ-Weil pairing on $\mathcal{A}[\ell]$ restricts trivially to $\mathcal{G}$; and*
2. *$\mathcal{G}$ is a maximal subgroup concerning the first property.*

Definition 2.3. *Let p be a prime integer and ℓ be an integer relatively prime to p. Let $\mathcal{A}/\overline{\mathbb{F}}_p$ be a g-dimensional PPAV. A proper subgroup $\mathcal{G}$ of $\mathcal{A}[\ell]$ is an $(\ell, \ldots, \ell)$-subgroup if $\mathcal{G}$ is a maximal ℓ-isotropic subgroup of $\mathcal{A}[\ell]$ such that $\mathcal{A}[n] \not\subseteq \mathcal{G}$ for any $1 < n \leq \ell$.*

For any prime number ℓ relatively prime to p, and a positive integer n, we have $\mathcal{A}[\ell^n] \cong (\mathbb{Z}_\ell)^{2g}$. Any $(\ell, \ldots, \ell)$-subgroup $\mathcal{G} \subset \mathcal{A}[\ell]$ is isomorphic to $(\mathbb{Z}_\ell)^g$ while for $(\ell^n, \ldots, \ell^n)$-subgroups $\mathcal{G} \subset \mathcal{A}[\ell^n]$ we have

$$\mathcal{G} \cong \mathbb{Z}_{\ell^{n_1}} \times \cdots \times \mathbb{Z}_{\ell^{n_g}} \text{ for some } n_1 \geq \ldots \geq n_g \text{ with } \sum_{i=1}^{n} n_i = gn.$$

Definition 2.4. *An $(\ell, \ldots, \ell)$-isogeny $\phi \colon \mathcal{A} \to \mathcal{A}'$ is an isogeny with kernel* $\ker \phi \subset \mathcal{A}[\ell]$ *being an $(\ell, \ldots, \ell)$-subgroup.*

Remark 2.1. It is worth highlighting that $(\ell, \ldots, \ell)$-isogenies preserve polarizations. Also, $(\ell^n, \ldots, \ell^n)$-isogenies can be decomposed as n $(\ell, \ldots, \ell)$-isogenies [13, Lemma 5.5.1].

Notation. Let n be a positive integer. We use the notation $[\![n]\!]$ to refer the list (in decreasing order) $[n, n-1, \ldots, 1]$. We denote the list of one repeated n times by $[\![1]\!]^n$. We represent vectors by bold letters (e.g., $\mathbf{v}$) and lists by sans serif letters (e.g., L). Sub-indexes label each entry of vectors and lists (e.g., $\mathbf{v}_k$ and L_k).

3 Strategies framework over PPAVs

This section proposes a strategy-based technique for solving the following problem.

Problem 3.1. Let p be a prime integer and ℓ be an integer relatively prime to p. Let $n \in \mathbb{Z}$ be a positive integer. Given a g-dimensional PPAV $\mathcal{A}/\overline{\mathbb{F}}_p$, a list H of points on $\mathcal{A}$, and an $(\ell^n, \ldots, \ell^n)$-subgroup $\mathcal{G} \subset \mathcal{A}[\ell^n]$: calculate the codomain of the $(\ell^n, \ldots, \ell^n)$-isogeny $\phi \colon \mathcal{A} \to \mathcal{A}'$ with kernel $\ker \phi = \mathcal{G}$ along with the list $[\phi(\mathsf{h}) \mid \mathsf{h} \in \mathsf{H}]$ of points on $\mathcal{A}'$.

Consider a g-dimensional PPAV $\mathcal{A}/\overline{\mathbb{F}}_p$, and ϕ the $(\ell^n, \ldots, \ell^n)$-isogeny with domain $\mathcal{A}$ and kernel $\mathcal{G} = \langle \mathbf{g}_1, \ldots, \mathbf{g}_g \rangle \cong (\mathbb{Z}_{\ell^n})^g$. Let $i \in [\![n-1]\!]$ and let Δ_n be a discrete rectangular triangular labeled as DRT and illustrated in Figure 1, with

- Point $\mathsf{pt}_{0,0} = (\mathbf{g}_1, \ldots, \mathbf{g}_g)$ at the right angle.
- Points $\mathsf{pt}_{0,i} = ([\ell^i]\mathbf{g}_1, \ldots, [\ell^i]\mathbf{g}_g)$ at the left cathetus.
- Points $\mathsf{pt}_{0,n-1}$ and

  $$\mathsf{pt}_{i,n-1-i} = (\phi_{i-1} \circ \cdots \circ \phi_1(\mathbf{g}'_1), \ldots, \phi_{i-1} \circ \cdots \circ \phi_1(\mathbf{g}'_g))$$

  at the hypotenuse, where $\mathbf{g}' \coloneqq (\mathbf{g}'_1, \ldots, \mathbf{g}'_g) = \mathsf{pt}_{i,n-2-i}$, $\phi_{i-1} \colon \mathcal{A}_{i-1} \to \mathcal{A}_i$ is the $(\ell, \ldots, \ell)$-isogeny with kernel the $(\ell, \ldots, \ell)$-subgroup $\langle \mathsf{pt}_{i-1,n-i} \rangle$ and $\mathcal{A}_0 = \mathcal{A}$.
- Points $\mathsf{pt}_{i,0} = (\phi_i(\mathbf{g}'_1), \ldots, \phi_i(\mathbf{g}'_g))$ at the upper cathetus with $\mathbf{g}' \coloneqq (\mathbf{g}'_1, \ldots, \mathbf{g}'_g) = \mathsf{pt}_{i-1,0}$.

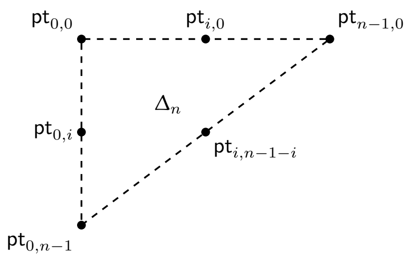

Fig. 1: Discrete rectangular triangular (DRT).

Any other point in Δ_n corresponds with scalar multiplications and evaluations of the cathetuses. Notice that the hypotenuse implicitly describes a path between $\mathcal{A}$ and the codomain $\mathcal{A}' = \mathcal{A}_n$ of the $(\ell^n, \ldots, \ell^n)$-isogeny $\phi = \phi_n \circ \cdots \phi_2 \circ \phi_1$ with kernel the $(\ell^n, \ldots, \ell^n)$-subgroup $\mathcal{G}$,

$$\mathcal{A}_0 = \mathcal{A} \xrightarrow{\phi_1} \mathcal{A}_1 \xrightarrow{\phi_2} \mathcal{A}_2 \xrightarrow{\phi_3} \cdots \xrightarrow{\phi_n} \mathcal{A}' = \mathcal{A}_n.$$

Definition 3.1. *A g-tuple* $\mathbf{g} = (\mathbf{g}_1, \ldots, \mathbf{g}_g)$ *has order* $(\ell, \ldots, \ell)$ *if each* $\mathbf{g}_i$ *has order* ℓ*.*

Definition 3.2. *Given a g-dimensional PPAV* $\mathcal{A}$*, and an* $(\ell, \ldots, \ell)$*-subgroup* $\mathcal{G} = \langle \mathbf{g}_1, \ldots, \mathbf{g}_g \rangle \cong (\mathbb{Z}_{\ell^n})^g$ *on* $\mathcal{A}$*, let* Δ_n *be the DRT as described above. A strategy is a weighted binary tree* St_n *inside* Δ_n*, with the root being the point at the right angle of* Δ_n *and the tree leaves being the points at the hypotenuse of* Δ_n*.*

One crucial remark is that any strategy, as defined in Definition 3.2, can be recursively decomposed into two binary sub-trees [14], one contained in Δ_{n-h} and another in Δ_h. 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. Moreover, one needs to compute h multiplications-by-ℓ (resp. $n-h$ $(\ell, \ldots, \ell)$-isogeny evaluations) to move into the left-side (resp. right-side) sub-tree. Figure 2 illustrates the general idea behind a strategy. Since Δ_n has $\frac{(n-1)n}{2}$ points, the maximum number of multiplications-by-ℓ and isogeny evaluations is then $\frac{(n-1)n}{2}$.

Definition 3.3. *An* $(n-1)$*-length encoded strategy is a strategy but represented as a list of* $n-1$ *positive integers smaller than* n*.*

Definition 3.4 (Multiplicative strategy). *An* $(n-1)$*-length encoded strategy* St_n *of the form* $[\![n-1]\!]$ *is called a multiplicative strategy.*

Definition 3.5 (Evaluative strategy). *An* $(n-1)$*-length encoded strategy* St_n *of the form* $[\![1]\!]^{n-1}$ *is called an evaluative strategy.*

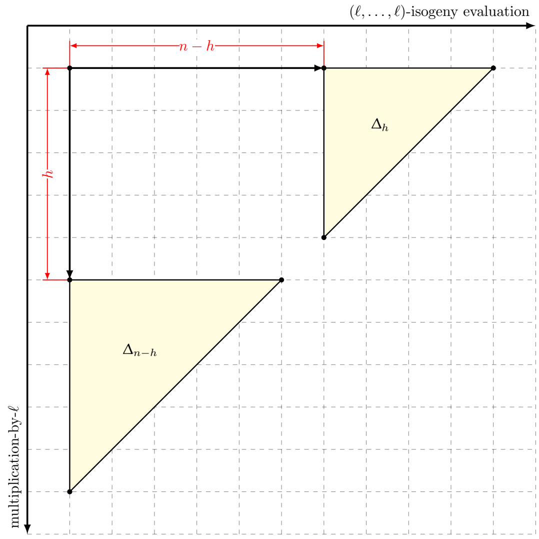

Fig. 2: The strategy technique reduces the computations from Δ_n into two binary subtrees, one contained in Δ_{n-h} and another in Δ_h.

Definition 3.6 (Balanced strategy). *An $(n-1)$-length encoded strategy St_n that recursively splits Δ_n into two sub-triangles $\Delta_{\lfloor n/2 \rfloor}$ and $\Delta_{\lceil n/2 \rceil}$ is called a balanced strategy.*

Definition 3.7. *An isogeny construction refers to computing the codomain of the isogeny itself. In contrast, an isogeny evaluation refers to pushing points through the isogeny itself.*

Remark 3.1. As pointed out above, the hypothenuse of a DRT Δ_n implicitly describes the $(\ell^n, \dots, \ell^n)$-isogeny $\phi\colon \mathcal{A} \to \mathcal{A}'$ with kernel $\mathcal{G}$ as the composition of n $(\ell, \dots, \ell)$-isogenies ϕ_i's. Therefore, a strategy outlines a procedure for passing through all those $(\ell, \dots, \ell)$-isogenies ϕ_i's with less running time than when computing the full DRT Δ_n. Consequently, a strategy allows us to solve Problem 3.1 efficiently; that is, it enables us to push a list of points H through each isogeny ϕ_i and thus to get $[\phi(\mathsf{h}) \mid \mathsf{h} \in \mathsf{H}]$ along with the codomain $\mathcal{A}'$ of ϕ.

If μ and η denote the cost concerning the multiplication-by-ℓ and $(\ell, \dots, \ell)$-isogeny evaluation, respectively. Then, the associated cost of an $(n-1)$-length enconded strategy St_n is

$$\mathsf{Cost}\,(\mathsf{St}_n) = \mathsf{Cost}\,(\mathsf{St}_{n-h}) + \mathsf{Cost}\,(\mathsf{St}_h) + h\mu + (n-h)\eta,$$

A multiplicative strategy performs n $(\ell, \dots, \ell)$-isogeny constructions, $n-1$ $(\ell, \dots, \ell)$-isogeny evaluations, and a quadratic number of multiplications-by-ℓ

$\frac{(n-1)n}{2}$. While an evaluative strategy performs n $(\ell,\ldots,\ell)$-isogeny constructions, $n-1$ multiplications-by-ℓ, and a quadratic number of $(\ell,\ldots,\ell)$-isogeny 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 3.8 (Optimal strategy). *An $(n-1)$-length encoded strategy St_n with minimal associated cost $\mathsf{Cost}\left(\mathsf{St}_n\right)$ is called an optimal strategy. In other words, any other different strategy has an associated cost greater than or equal to $\mathsf{Cost}\left(\mathsf{St}_n\right)$.*

Remark 3.2. The term of *optimal* strategy was initially proposed in [14] but in the context of elliptic curves (i.e., one-dimensional PPAVs).

If κ denotes the cost of an $(\ell,\ldots,\ell)$-isogeny construction, then the cost of computing the codomain of the $(\ell^n,\ldots,\ell^n)$-isogeny $\phi\colon \mathcal{A}\to\mathcal{A}'$ becomes $\mathsf{Cost}\left(\mathsf{St}_n\right)+n\kappa$. Furthermore, pushing an m-length list of points on $\mathcal{A}$ through ϕ adds another linear factor to the associated cost, which gives the below cost.

$$\tau = \mathsf{Cost}\left(\mathsf{St}_n\right) + n\kappa + nm.$$

Algorithm 1 describes a dynamic programming technique for finding an optimal strategy given μ and η, which has a quadratic polynomial running time in n. While Algorithm 2 presents the strategy-based procedure to calculate the codomain $\mathcal{A}'$ and push a list of points on A through ϕ.

Algorithm 1 Procedure to compute an optimal strategy for a St_n

---

Inputs: A prime integer number ℓ, a positive integer n, and the costs μ and η of the multiplication-by-ℓ and the $(\ell,\ldots,\ell)$-isogeny evaluation.
Output: Optimal strategy St_n concerning μ and η

```
Set as optimal strategy St_1 = []
for i = 2 to n do
    Solve
        s = argmin_{h ∈ [[i-1]]} {Cost(St_{i-h}) + Cost(St_h) + hμ + (i-h)η}
    Set as the optimal strategy St_i = [s] ∪ St_{i-s} ∪ St_s
end for
return St_n
```

Lemma 3.1. *Let ℓ be a small prime number. Algorithm 2 provides a method for solving Problem 3.1 in polynomial time in the variables $\ell\log_2\ell$ and n.*

Algorithm 2 Strategy technique to construct $(\ell^n, \ldots, \ell^n)$-isogenies between g-dimensional PPAVs

Inputs: A PPAV g-dimensional $\mathcal{A}$, an $(\ell^n, \ldots, \ell^n)$-subgroup $\mathcal{G} = \langle \mathbf{g}_1, \ldots, \mathbf{g}_g \rangle \cong (\mathbb{Z}_{\ell^n})^g$ on $\mathcal{A}$, a list H of points on $\mathcal{A}$, and an $(n-1)$-length strategy St.

Output: Codomain PPAV of the $(\ell^n, \ldots, \ell^n)$-isogeny $\phi\colon \mathcal{A} \to \mathcal{A}'$ with kernel the $(\ell^n, \ldots, \ell^n)$-subgroup $\mathcal{G}$, and the list $[\phi(\mathsf{h}) \mid \mathsf{h} \in \mathsf{H}]$ of points on $\mathcal{A}'$

$k \leftarrow 1$
$\mathcal{A}' \leftarrow \mathcal{A}$
$\mathbf{g}' \leftarrow (\mathbf{g}_1, \ldots, \mathbf{g}_g)$
$\mathsf{K} \leftarrow [\mathbf{g}']$
$\mathsf{H}' \leftarrow \mathsf{H}$
for $i = 1$ to $n-1$ **do**
    $\mathbf{g}' \leftarrow$ last element of K
    while $\mathbf{g}'$ does not have order $(\ell, \ldots, \ell)$ **do**
        $s_k \leftarrow k$-th element of St_n
        $\mathbf{g}' \leftarrow ([\ell^{s_k}]\mathbf{g}'_1, \ldots, [\ell^{s_k}]\mathbf{g}'_g)$
        Append $\mathbf{g}'$ to the last element of K
        $k \leftarrow k + 1$
    end while
    assert $\mathbf{g}'$ has order $(\ell, \ldots, \ell)$
    Remove the last element $\mathbf{g}'$ of K
    $\mathcal{A}' \leftarrow$ codomain of the $(\ell, \ldots, \ell)$-isogeny ϕ with kernel $\langle \mathbf{g}'_1, \ldots, \mathbf{g}'_g \rangle$
    $\mathsf{K} \leftarrow [(\phi(\mathbf{k}_1), \ldots, \phi(\mathbf{k}_g)) \mid \mathbf{k} \in \mathsf{K}]$
    $\mathsf{H}' \leftarrow [\phi(\mathsf{h}') \mid \mathsf{h}' \in \mathsf{H}']$
end for
Extract and remove the last element $\mathbf{g}'$ of K
assert $\mathbf{g}'$ has order $(\ell, \ldots, \ell)$
$\mathcal{A}' \leftarrow$ codomain of the $(\ell, \ldots, \ell)$-isogeny ϕ with kernel $\langle \mathbf{g}'_1, \ldots, \mathbf{g}'_g \rangle$
$\mathsf{H}' \leftarrow [\phi(\mathsf{h}') \mid \mathsf{h}' \in \mathsf{H}']$
return $\mathcal{A}', \mathsf{H}'$

Proof. Notice that a multiplication-by-ℓ over g-dimensional PPAVs runs in time $\log_2 \ell$ (e.g., using Right-to-left algorithm, Montgomery Ladders, wNAF-based algorithms, etc.). On the other hand, Lubicz and Robert provide in [22, 23] algorithms for computing $(\ell, \ldots, \ell)$-isogenies between higher-dimensional abelian varieties with $(\ell, \ldots, \ell)$-subgroups as kernels in polynomial time in $\ell \log_2 \ell$. On that basis, our Algorithm 2 gives a method to compute (and push points through) $(\ell^n, \ldots, \ell^n)$-isogenies in polynomial time in the variables $\ell \log_2 \ell$ and n.

Remark 3.3 (one-dimensional PPAVs). The case of one-dimensional PPAVs lands in the well-known elliptic curve case. Our Algorithm 2 coincides with the technique from [14]. For small primes $\ell \leq 89$, the traditional Vélu formulas give a polynomial time complexity of ℓ operations for computing ℓ-isogenies, which implies that Algorithm 2 runs in polynomial time in the variables ℓ and n. For larger primes $\ell > 89$, the square-root Vélu formulas from [4] reduce the running time of computing ℓ-isogenies to $\tilde{O}(\sqrt{\ell})$ operations [6], which implies that Algorithm 2 runs in polynomial time in the variables $\sqrt{\ell}\log_2 \ell$ and n when $\ell \geq 89$.

Remark 3.4 (two-dimensional PPAVs). Cosset and Robert give in [10] a method to compute (ℓ,ℓ)-isogenies in polynomial time in ℓ on Jacobians of genus-two curves. Consequently, our Algorithm 2 gives a method to compute (and push points through) (ℓ^n,ℓ^n)-isogenies in polynomial time in the variables ℓ and n.

4 Experiments on two-dimensional PPAVs

For a deeper definition of Jacobians of genus-two curves, we recommend reading [10, 17, 19, 31]. Let $\mathcal{C}$ be a genus-two hyperelliptic curves given by Equation 1,

$$\mathcal{C}\colon y^2 = f(x), \quad f(x) \in \mathbb{F}_{p^2}[x] \text{ with } \deg f = 6. \tag{1}$$

The Jacobian $\mathcal{J}$ of $\mathcal{C}$ is a two-dimensional abelian variety. Elements in $\mathcal{J}$ are represented as pair of polynomials (u,v) where u is monic degree-two polynomial, and $v^2 - f \bmod u \equiv 0$, namely Mumford representation [9, Chapter 14]. The roots of $u(x)$ determine two points P and Q on the curve $\mathcal{C}$ over $\overline{\mathbb{F}}_{p^2}$. When the points P and Q are known, we write the element $(u,v) \in \mathcal{J}$ as $[P+Q]$.

If $\mathcal{A}$ is a two-dimensional PPAV over $\overline{\mathbb{F}}_p$, then $\mathcal{A}$ is isomorphinc to the product of two elliptic curves $\mathcal{E} \times \mathcal{F}$ or the Jacobian $\mathcal{J}$ of a genus-two curve $\mathcal{C}$.

4.1 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, 8, 20] for a better understanding.

Consider a genus-two curve $\mathcal{C}$ determined Equation (1). 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 $i \coloneqq 1,2,3$, such that

$$\mathcal{G} = \langle (F_1(x),0),(F_2(x),0)\rangle = \{\mathbf{0}_{\mathcal{J}},(F_1(x),0),(F_2(x),0),(F_3(x),0)\}$$

is a $(2,2)$-subgroup. 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 of the $(2,2)$-isogeny $\phi\colon \mathcal{J} \to \mathcal{J}'$ with $\ker\phi = \mathcal{G}$ is isomorphic to

[6] For cryptographic sizes of ℓ, the square-roof Vélu formulas are tailored to a Karatsuba-like polynomial multiplication [1], slightly "increasing" the complexity from $\tilde{O}(\sqrt{\ell})$ to $O(\ell^{\log_2 3})$

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

where

$$H_i(x) = \delta^{-1}\left(\frac{dF_j(x)}{dx}F_k(x) - \frac{dF_k(x)}{dx}F_j(x)\right)$$

with (ijk) a cyclic permutation of 1,2,3. On the other hand, pushing an element $D \in \mathcal{J}$ through ϕ summarizes as follows.

1. Decompose $D \in \mathcal{J}$ as $D = [P + Q]$ where $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are points on the curve $\mathcal{C}$.
2. Find four points P', Q', P'', Q'' on $\mathcal{C}$ such that $\phi(D) = [P' + P''] + [Q' + Q'']$ as follows.
   - Calculate the abscissa of P' (resp. P'') by solving the following quadratic equation in x_2:

     $$F_1(x_P)H_1(x_2) + F_2(x_P)G_2(x_2) = 0.$$

   - Calculate the ordinate of P' (resp. P'') by solving the following equation in y_2:

     $$y_p y_2 = F_1(x_P)H_1(x_{P'})(x_P - x_{P'}).$$

   - Repeat the same as above but for Q' (resp. Q'').
3. Compute $\phi(D) = [P' + P''] + [Q' + Q'']$.

The authors from [7] propose and describe an efficient Gröbner basis approach for computing (and evaluating under) $(2,2)$-isogenies. Conversely, the authors from [20] present explicit formulas for pushing points through $(2,2)$-isogenies with a kernel of the form $\mathcal{G} = \langle (x,0), (x^2 - Ax + 1, 0)\rangle$. They also characterize the family of genus-two curves given by

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

and prove that any genus-two curves can be transformed into such a shape [7]. Consequently, any $(2,2)$-subgroup over $\mathcal{J}$ maps into a suitable $\mathcal{G}$.

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

Speedups concerning the Magma-public code from [20]. The technique from [20, Section 5.3] 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 [20] manages to reduce the running time in their approach from $O(n^2)$ to $O(n\sqrt{n})$. Indeed, the technique from [20] falls into our strategy definition and relies on a multiplicative-like nature. However, the latest code version from [20] uses a balanced strategy technique based on [14]. We compare our implementation of Algorithm 2 with the given in [20]. First, following the suggestion of [14] we use Algorithm 1 for computing the balanced strategy, and we notice such a strategy differs from the approach in [20]. Second, to identify the main difference, we include counters for the number of multiplications-by-two and $(2, 2)$-isogeny evaluations. All our experiments use the balanced strategy and the parameters with a 171-bit prime proposed in [20]. Our code implementation is about 1.3x faster than [20] (see Tables 1 and 2).

Technique	#[Multiplications by 2]	#[$(2, 2)$-isogeny evaluations]	Runtime
$(2^{87}, 2^{87})$-isogeny with 4 evaluations of extra points			
Balanced strategy from [20]	1033	874	1907
Balanced strategy	768	874	1642
$(2^{87}, 2^{87})$-isogeny (only codomain curve calculation)			
Balanced strategy from [20]	1033	526	1559
Balanced strategy	768	526	1294

Table 1: Number of multiplications-by-two and $(2, 2)$-isogeny evaluations required to compute a $(2^{87}, 2^{87})$-isogeny, the runtime column corresponds with the sum of both numbers. The field characteristic is p171 as defined in [20].

Procedure	Baseline [20]	This work	Speedup
$(2^{87}, 2^{87})$-isogeny with 4 evaluations of extra points	0.1779	0.1336	1.332x
$(2^{87}, 2^{87})$-isogeny (only codomain curve calculation)	0.1659	0.1229	1.335x

Table 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 time (in seconds) of computing 100 random $(2^{87}, 2^{87})$-isogenies. The field characteristic is p171 as defined in [20].

Speedups concerning the SageMath-public code from [29]. To illustrate the impact of our results, we point out that our results directly apply to the attacks in [7,24,30]. For example, the most demanding 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 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 2 into the public SageMath language code from [29] and present 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 [2,11]. In particular, our experiments show a speedup of 1.19x—1.26x in the Castryck-Decru attack (see Table 3).

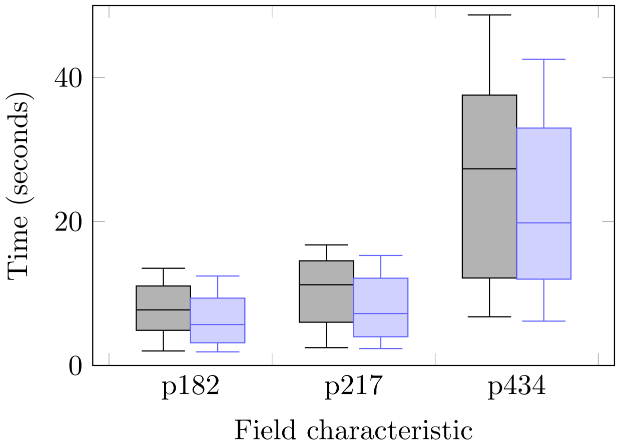

Fig. 3: Our experiments were executed on a 2.3 GHz 8-Core Intel Core i9 machine with 16GB of RAM. The measures correspond with the key-recovery timings (in seconds) of 100 random SIDH keys. The data in blue ink correspond with this work, while the gray ink is the baseline code from [29].

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 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 key-recovery average timings (in seconds) of 100 random SIDH keys.

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

This section summarizes the $(3,3)$-isogenies formulas by Bruin, Flynn and Testa [5]. Consider a $(3,3)$-subgroup $\mathcal{G} = \langle T_1, T_2 \rangle \subset \mathcal{J}[3]$ of a genus-two curve $\mathcal{D}$ given by Equation (1). In [5], the authors provide a parametrization of the genus-two curve $\mathcal{D}$ determined by the 3-tuple $(\mathcal{D}, T_1, T_2)$, namely (r,s,t)-parametrization. In particular, they show that the curve $\mathcal{D}$ is isomorphic to

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

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 [5] suggest working with the associated Kummer surface $\mathcal{K} \coloneqq \mathcal{J}/\langle -1 \rangle$ instead of the Jacobian $\mathcal{J}$. They propose mapping the divisor from $\mathcal{J}$ to K by some relation $\xi\colon D \mapsto (\xi_0\colon \xi_1, \colon \xi_2, \colon \xi_3)$. More precisely, if $f = f_6x^6 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$, and $D \in \mathcal{J}$ 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 $\mathcal{K}$ admits the following quartic equation model

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

where $\Psi(\xi_0,\xi_1,\xi_2)$ is a homogeneous degree-4 polynomial. The isogeny $\phi\colon \mathcal{J} \to \mathcal{J}'$ with kernel $\mathcal{G}$ induces an isogeny between the Kummer surfaces $\mathcal{K}$ and $\mathcal{K}'$. The authors from [5] give explicit formulas for computing the codomain curve and the induced map. While the authors from [15] provide better formulas for the $(3,3)$-isogenies. They simplify formulas and reduce the number of required multiplications in [5]. They propose to use a Gröbner basis approach [6, 15], 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.

Speedups concerning the Magma-public code from [15]. The authors from [15] provide a Magma code implementation that uses a balanced strategy technique based on [14]. We use their code and implement Algorithm 2 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 [15]. Similarly to Section 4.1, the balanced strategy as suggested in [15] differs from the balanced strategy computed by employing Algorithm 1. So, to identify the main difference, we include counters for the number of multiplications-by-three and $(3,3)$-isogeny evaluations. Table 4 lists those operation numbers concerning different strategy techniques (balanced and optimal balanced) and compares them against the algorithm from [15]. Our experiments compare [15] against the following two different strategies:

1. Balanced strategy just as suggested in [15] but employing Algorithm 2; and
2. Optimal balanced strategy calculated as in Section 3 with $\mu = \eta$ and using Algorithm 2.

Technique	#[Multiplications by 3]	#[$(3,3)$-isogeny evaluations]	Runtime
Balanced strategy from [15]	2884	2380	5264
Balanced strategy	1936	2290	4226
Optimal balanced strategy	1818	2408	4226

Table 4: 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 4, we expect our implementation of Algorithm 2 to be 1.25x faster than [15], which is about 20% of savings.

On the other hand, we point out that our results directly apply to the attacks in [7,24,30]. For example, the most demanding computations in the Castryck-Decru attack are the $(3^i, 3^i)$-isogenies for some integer $i \in [\![n]\!]$ close to n; such a shortcut splits the computations into two parts: the $(3^i, 3^i)$-isogeny computation and some discrete logarithm computations. In any case, the isogenies still play an essential role in the Castryck-Decru attack. We additionally plug our Algorithm 2 into the public Magma language code of [15] and present our results in Figure 4. Our experiments focus on the quadratic field extensions of $\mathbb{F}_{p^2}$ with prime characteristic p751 as defined in [2].

It is worth highlighting that the nature of Algorithm 2 allows isolating the mappings of points from the Kummer Surface into the Jacobian, which are only needed when computing the codomain of the isogeny. Consequently, our implementation of Algorithm 2 isolates the calls to $\mathsf{Points}(\mathsf{J}, \mathsf{h})[1]$ into the isogeny codomain calculation (i.e., in steps 16 and 22 of Algorithm 2).

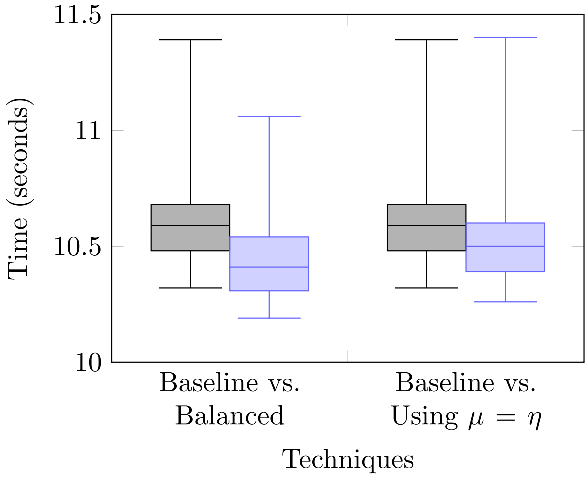

Fig. 4: Our experiments were executed on a 2.3 GHz 8-Core Intel Core i9 machine with 16GB of RAM. The measures correspond with the key-recovery timings (in seconds) of 100 random SIDH keys. The data in blue ink correspond with this work, while the gray ink is the baseline code from [15]. The field characteristic is p751 as defined in [2].

We notice from the experiments that the bottleneck in the current implementations in [15] and ours is the calculation of the codomain curve along with the data required for evaluating the $(3, 3)$-isogeny [8], which takes on average 0.04 seconds [9]. 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 4.2, 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 4 illustrate such savings.

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

8 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

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

5 Discussion on the applications of our results.

Constructive applications. The authors in [15] propose a genus-two variant of the Charles-Goren-Lauter hash function by employing isogenies over curves with torsion 3^n. In particular, [15] suggests constructing isogenies with $(3^n, 3^n)$-subgroups as kernels defined over $\mathbb{F}_{p^2}$. Now, our experiments from Section 4.2 illustrate a theoretical savings of 20% (see Table 4) when computing isogenies as required in [15]. Therefore, our results should speedup the hash function from [15] by at most 1.25x.

The presented strategy techniques also applies to the recent work [13]. That work discusses the need of strategies for computing higher-dimensional isogeny. More precisely, Algorithm 2 describes an efficient algorithm to perform the KernelToIsogeny procedure from [13]. The recent work by Leroux [21] also requires isogenies between higher-dimensional abelian varieties, and thus our results also apply to the Verifiable Random Function proposal from [21].

The most demanding computations in the Public-Key Encryption FESTA [3] (resp. QFESTA [28]) are the isogenies between products of elliptic curves (passing through Jacobian of genus-two curves). The authors from [3] include a public SageMath code that integrates our implementation of Algorithm 2. Additionally, the public SageMath implementations of QFESTA [28] and the Key Encapsulation Mechanism from [26] use the FESTA code for computing the $(2^n, 2^n)$-isogenies, which currently (and implicitly) employs our implementation of Algorithm 2.

Lastly, the weak Verifiable Delay Function proposal from [16] also requires $(2^n, 2^n)$-isogenies as the central core. Therefore, our Algorithm 2 should improve their running time by at most 1.25x faster.

Better optimal strategies. The authors from [18] suggest computing 2^{2k+1}-isogenies by calculating at first one 2-isogeny, and next k 4-isogenies (with a different weight than 2-isogenies). More precisely, [18] proposes optimal strategies by assuming that the first isogeny (which is a 2-isogeny) has a lower cost than the subsequent isogenies (which are 4-isogenies); [18] shows that such optimal strategies lead to 15% savings. When the domain of the $(2^n, 2^n)$-isogeny is the product of elliptic curves, then the optimal strategy falls into a similar case as in [18]: the first isogeny corresponds with a $(2, 2)$-isogeny going from a product of elliptic curves to the Jacobian of a genus-two curve, while the remaining $(2, 2)$-isogenies (probably except for the last one) are between Jacobian of genus-two curves. Since the point arithmetic and isogenies over products of elliptic curves cost less than over Jacobian of genus-two curves, then there is still room for improving higher-dimension isogenies with domain (and maybe also with codomain) being a product of elliptic curves. However, further analysis is required.

Acknowledgements. We thank Sabrina Kunzweiler, Thomas Decru, and anonymous reviewers for their comments and discussions on an early version of this manuscript. We additionally thank Giacomo Pope for his comments about the

SageMath code, particularly in the gluing step in the Castryck-Decru attack, where the domain of the isogeny is a product of elliptic curves.

A. Pizarro-Madariaga has been partially supported by Proyecto PUENTE UVA20993. E. Riquelme was supported in part by grants 2020433 IF/R (Universidad del Bío-Bío, Chile) and CyTeD ("Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo") project 522RT0131.

References

1. Adj, G., Chi-Domínguez, J.J., Rodríguez-Henríquez, F.: Karatsuba-based square-root Vélu's formulas applied to two isogeny-based protocols. Journal of Cryptographic Engineering **13**, 89–106 (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., Maino, L., Pope, G.: FESTA: Fast Encryption from Supersingular Torsion Attacks. Cryptology ePrint Archive, Report 2023/660 (2023), https://eprint.iacr.org/2023/660
4. Bernstein, D.J., Feo, L.D., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. Algorithmic Number Theory Symposium (ANTS-XIV), MSP Open Book Series **4**(1), 39–55 (2020). https://doi.org/10.2140/obs.2020.4.39
5. Bruin, N., Flynn, E.V., 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
6. Castryck, W., Decru, T.: Multiradical isogenies. Cryptology ePrint Archive, Report 2021/1133 (2021), https://eprint.iacr.org/2021/1133
7. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 423–447. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_15
8. 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
9. Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of Elliptic and Hyperelliptic Curve Cryptography, Second Edition. Chapman & Hall/CRC, 2nd edn. (2012)
10. Cosset, R., Robert, D.: Computing (ℓ, ℓ)–isogenies in polynomial time on Jacobians of genus 2 curves. Mathematics of Computation **84**(294), 1953–1975 (2015). https://doi.org/http://www.ams.org/jourcgi/jour-getitem?pii=S0025-5718-2014-02899-8/
11. Costello, C.: The case for SIKE: A decade of the supersingular isogeny problem. Cryptology ePrint Archive, Report 2021/543 (2021), https://eprint.iacr.org/2021/543
12. Costello, C., Smith, B.: The supersingular isogeny problem in genus 2 and beyond. In: Ding, J., Tillich, J.P. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020. pp. 151–168. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-44223-1_9
13. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: SQISignHD: New Dimensions in Cryptography. Cryptology ePrint Archive, Report 2023/436 (2023), https://eprint.iacr.org/2023/436

14. 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
15. Decru, T., Kunzweiler, S.: Efficient Computation of $(3^n, 3^n)$-Isogenies. In: Mrabet, N.E., Feo, L.D., Duquesne, S. (eds.) Progress in Cryptology - AFRICACRYPT 2023 - 14th International Conference on Cryptology in Africa. LNCS, vol. 14064, pp. 53–78. Springer (2023). https://doi.org/10.1007/978-3-031-37679-5_3
16. Decru, T., Maino, L., Sanso, A.: Towards a Quantum-resistant Weak Verifiable Delay Function. In: Aly, A., Tibouchi, M. (eds.) Progress in Cryptology - LATINCRYPT 2023 - 8th International Conference on Cryptology and Information Security in Latin America. LNCS, vol. 14168, pp. 149–168. Springer (2023), https://doi.org/10.1007/978-3-031-44469-2_8
17. Delfs, F.C.: Isogenies and Endomorphism Rings of Abelian Varieties of Low Dimension. Ph.D. thesis, Carl von Ossietzky Universität Oldenburg (2015), available at https://d-nb.info/1093683937/34
18. 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. LNCS, vol. 13161, pp. 49–72. Springer (2022). https://doi.org/10.1007/978-3-030-95312-6_3
19. 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
20. Kunzweiler, S.: Efficient computation of $(2^n, 2^n)$-isogenies. Cryptology ePrint Archive, Report 2022/990 (2022), https://eprint.iacr.org/2022/990
21. Leroux, A.: Verifiable random function from the Deuring correspondence and higher dimensional isogenies. Cryptology ePrint Archive, Report 2023/1251 (2023), https://eprint.iacr.org/2023/1251
22. Lubicz, D., Robert, D.: Computing isogenies between abelian varieties. Compositio Mathematica **148**(5), 1483–1515 (2012). https://doi.org/10.1112/S0010437X12000243
23. Lubicz, D., Robert, D.: Fast change of level and applications to isogenies. Research in Number Theory **9**(7), 1–28 (2023). https://doi.org/10.1007/s40993-022-00407-9
24. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 448–471. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_16
25. Milne, J.S.: Abelian Varieties (v2.00) (2008), available at www.jmilne.org/math/
26. Moriya, T.: IS-CUBE: An isogeny-based compact KEM using a boxed SIDH diagram. Cryptology ePrint Archive, Report 2023/1506 (2023), https://eprint.iacr.org/2023/1506
27. Mumford, D.: Abelian Varieties. Tata Institute of Fundamental Research, Bombay (1970), available at https://wstein.org/edu/Fall2003/252/references/mumford-abvar/
28. Nakagawa, K., Onuki, H.: QFESTA: Efficient Algorithms and Parameters for FESTA using Quaternion Algebras. Cryptology ePrint Archive, Report 2023/1468 (2023), https://eprint.iacr.org/2023/1468
29. Oudompheng, R., Pope, G.: A note on reimplementing the castryck-decru attack and lessons learned for SageMath. Cryptology ePrint Archive, Report 2022/1283 (2022), https://eprint.iacr.org/2022/1283

30. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 472–503. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_17
31. Smith, B.: Explicit endomorphisms and correspondences. Ph.D. thesis, University of Sidney (2005), available at http://hdl.handle.net/2123/1066