Radical 2-isogenies and cryptographic hash functions in dimensions $1, 2$ and 3

Sabrina Kunzweiler[1], Luciano Maino[2], Tomoki Moriya[4], Christophe Petit[7], Giacomo Pope[2,3], Damien Robert[1], Miha Stopar[7,8], and Yan Bo Ti[5,6]

[1] Inria Bordeaux, Institut de Mathématiques de Bordeaux, France
sabrina.kunzweiler@math.u-bordeaux.fr, damien.robert@inria.fr,
[2] University of Bristol, Bristol, United Kingdom
luciano.maino@bristol.ac.uk
[3] NCC Group, Cheltenham, United Kingdom
giacomo.pope@nccgroup.com
[4] University of Birmingham, United Kingdom
t.moriya@bham.ac.uk
[5] DSO National Laboratories, Singapore
[6] National University of Singapore, Singapore
yanbo.ti@gmail.com
[7] Université libre de Bruxelles, Belgium
christophe.petit@ulb.be
[8] Ethereum Foundation
stopar.miha@gmail.com

Abstract. We provide explicit descriptions for radical 2-isogenies in dimensions one, two and three using theta coordinates. These formulas allow us to efficiently navigate in the corresponding isogeny graphs.
As an application of this, we implement different versions of the CGL hash function. Notably, the three-dimensional version is fastest, which demonstrates yet another potential of using higher dimensional isogeny graphs in cryptography.

1 Introduction

One of the first isogeny-based cryptographic protocols is the Charles–Goren–Lauter (CGL) hash function [16]. This hash function utilises the input bits to generate a random walk on the supersingular elliptic curve 2-isogeny graph, and outputs the j-invariant of the final vertex. The hard problem that underpins the security of the hash function is the difficulty of finding isogenies between two given supersingular elliptic curves.

Methods to compute isogenies in various cryptographic schemes have included using modular polynomials, Vélu's formulas, Vélu-sqrt [5] and radical isogenies. These methods work best for low degree isogenies, which can then be chained together to produce (smooth) large degree isogenies. The concept of radical isogenies between elliptic curves was introduced in [14]. A radical N-isogeny formula inputs a pair (E, P) consisting of an elliptic curve E and an N-torsion point $P \in E$, and outputs a pair (E', P') such that

- $P' \in E' = E/\langle P \rangle$ is an N-torsion point on the codomain of the isogeny $\phi : E \to E'$ with kernel $\langle P \rangle$,

- the isogeny $\phi' : E' \to E'/\langle P' \rangle$ is a *good* extension of ϕ, i.e. $\phi' \circ \phi$ is an N^2-isogeny.

These formulas are algebraic expressions in the coefficients of E, the coordinates of P and a radical $\sqrt[N]{\mathfrak{r}}$, where $\mathfrak{r}$ itself is an algebraic expression in the input. Furthermore it is required that varying the chosen N-th root changes the isogeny ϕ' induced by the output and thereby allows to obtain all N different good extensions of ϕ.

Elliptic curves are principally polarized (p.p.) abelian varieties of dimension $g = 1$. There have been various suggestions to generalize isogeny-based protocols to higher dimensions [31,62,13]. In this work, we will primarily be interested in p.p. abelian varieties of dimensions $g = 2$ and $g = 3$ which are called *p.p. abelian surfaces* and *p.p. abelian threefolds*, respectively.

There is a natural generalization of radical isogenies to higher dimensions as conjectured in [10] and proved in [58]. In dimension g, the kernel of an N-isogeny is isomorphic to the group $(\mathbb{Z}/N\mathbb{Z})^g$. Consequently, a radical isogeny formula takes as input a tuple $(A, P_1, \ldots, P_g)$ where A is the p.p. abelian variety and $P_1, \ldots, P_g$ generate the kernel of an N-isogeny. The output consists of the tuple $(A', P_1', \ldots, P_g')$, where similar as above

- $P_1', \ldots, P_g' \in A' = A/\langle P_1, \ldots, P_g \rangle$ generate the kernel of an N-isogeny $\phi' : A' \to A'/\langle P_1', \ldots, P_g' \rangle$
- the isogeny ϕ' is a good extension of ϕ, i.e. $\phi' \circ \phi$ is an N^2-isogeny.

Here, the radical formula will involve $g(g+1)/2$ N-th roots $\sqrt[N]{\mathfrak{r}_1}, \ldots, \sqrt[N]{\mathfrak{r}_{g(g+1)/2}}$. Different choices of these roots are in one-to-one correspondence with different good extensions of the isogeny ϕ.[9]

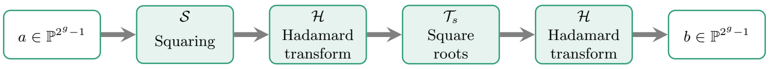
1.1 Our contributions

Throughout the paper, we consider p.p. abelian varieties A of dimension g, equipped with a level-2 theta structure θ^A. Broadly speaking, this allows us to identify each A with a point $a = (a_{0\ldots0} : \cdots : a_{1\ldots1}) \in \mathbb{P}^{2^g-1}$ in the projective space (indices counted in base two). This is known as the *level-*2 *theta null point* of A w.r.t. θ^A. More concretely, for dimensions $g = 1, 2, 3$:

- an elliptic curve is identified with a point $a = (a_0 : a_1) \in \mathbb{P}^1$,
- a p.p. abelian surface is identified with $a = (a_{00} : a_{01} : a_{10} : a_{11}) \in \mathbb{P}^3$,
- a p.p. abelian threefold is identified with $a = (a_{000} : \cdots : a_{111}) \in \mathbb{P}^7$.

We note that the level-2 theta null point further contains information on the entire 2-torsion of A.

Radical 2-isogenies in dimensions $g = 1, 2, 3$ Let $\phi : A \to B$ be a 2-isogeny. A radical formula for ϕ in this setting, transforms a theta null point $a \in \mathbb{P}^{2^g-1}$ to a theta null point $b \in \mathbb{P}^{2^g-1}$. We show that any such isogeny can be computed as the composition of four simple operations:



[9] We note that this correspondence will be more subtle in our setting (Subsection 3.5).

The definition of the maps $\mathcal{S}$ (coordinatewise squaring) and $\mathcal{H}$ (Hadamard transformation) is completely generic. Essentially, the map $\mathcal{T}_s$ consists in taking the square-roots of the coordinates. Here, the subtlety is that only $g(g+1)/2$ of the $2^g - 1$ square-roots can be chosen arbitrarily, while the remaining ones are uniquely determined by the others. This requires a dimension specific treatment, and we find an explicit description of $\mathcal{T}_s$ in dimensions $g = 1$, $g = 2$ (Corollary 5) and $g = 3$ (Theorem 8).

By definition, composing n radical 2-isogenies, results in a 2^n-isogeny. In the case $g = 1$, this task is completely straightforward (one only needs to avoid backtracking). In higher dimensions, the situation gets more complicated (see Subsection 3.4) and failure to do this composition correctly has already led to security gaps in cryptographic constructions. For example, the first instantiation of the CGL hash function in dimension 2 ([62]) was found to be insecure due to this problem, as was shown in [31] and fixed in [13]. We show that our formulas naturally result in the composition with *good* extensions.

In addition, we also derive new formulas for radical 2^k-isogenies for small values of k. While our simple framework for 2-isogenies lends itself to this task, it still gets more complicated with increasing dimension. Concretely, we present new radical formulas for 4- and 8-isogenies in dimension $g = 1$ (Propositions 12, 13), as well as an (almost) radical[10] 4-isogeny in dimension $g = 2$ (Proposition 14).

Our cryptographic hash function: Theta-CGL As an application of our new formulas, we propose the cryptographic hash function $\mathsf{Theta\text{-}CGL}_{g,\ell}$ with different variants defined by choosing

$$(g, \ell) \in \{(1,2), (1,4), (1,8), (2,2), (2,4), (3,2)\}. \tag{1}$$

This can be viewed as a generalization of the well known CGL hash function, [16]. Our hash function $\mathsf{Theta\text{-}CGL}_{g,\ell}$ works in the ℓ-isogeny graph of superspecial p.p. abelian varieties of dimension g. This means that for $g = 1$, we use the same setting as the original hash function. However, the higher dimensional isogeny graphs provide a richer structure than those of dimension one. As a consequence, one can significantly decrease the size of the prime p without compromising the security of the hash function. At the same time, the computation that has to be done per bit, remains essentially the same. This leads to considerable speed-ups of the hash function when setting $g > 1$.

Notably, the 3-dimensional version $\mathsf{Theta\text{-}CGL}_{g=3,\ell=2}$ is the fastest among all the variants. We highlight that this is in stark contrast with the setting of other isogeny-based cryptographic primitives for which the generalization to higher dimensions seems to result in significant slow-downs (SIDH, see G2SIDH in [31]) or seems infeasible with the state-of-the-art methods to compute isogenies (for example CSIDH [15] or SQISign [22]).

Implementation of $\mathsf{Theta\text{-}CGL}_{g,\ell}$ Our paper is accompanied by an optimized implementation of $\mathsf{Theta\text{-}CGL}_{g,\ell}$ in the programming language Rust. To

[10] We say that our formulas are *almost* radical, since they require the computation of two square roots and two fourth roots, while one would expect a formula requiring the computation of three fourth roots.

test our hash function, we ran Theta-CGL$_{g,\ell}$ for all pairs (g, ℓ) as in Eq. 1 with parameters suggested to obtain $\lambda = 128$ bit security. Concretely, this means that we work with a prime of bit length 256, 128 or 64, when $g = 1$, $g = 2$ or $g = 3$, respectively.
The observed timings, recorded on an Intel Core i7-9750H CPU with a clock-speed of 2.6 GHz with turbo-boost disabled, are summarized in the table below where each entry is the time in microseconds (μs) to hash one bit of a message (computed as an average of hashing one 324 bit block).

Theta-CGL$_{g,\ell}$	$\ell = 2$ (μs)	$\ell = 4$ (μs)	$\ell = 8$ (μs)
$g = 1$	9.73	6.29	5.36
$g = 2$	3.05	2.23	—
$g = 3$	1.33	—	—

These timings backup our expectation that the three-dimensional version of Theta-CGL outperforms the lower dimensional versions.
In addition to the Rust implementation, we provide an implementation in the computer algebra system SageMath which is meant as an educational resource to the community. Source code for both implementations is available online: `https://github.com/GiacomoPope/ThetaCGL`.

1.2 Comparison to the literature

Using different models of abelian varieties, there already exist radical isogeny formulas. We discuss how our new formulas compare to the existing ones from the literature. Moreover, we compare Theta-CGL$_{g,\ell}$ to previous proposals.

Comparison of radical isogeny formulas A radical N-isogeny formula requires the computation of $g(g+1)/2$ Nth roots. Efficient formulas aim at minimizing the remaining field operations. A thorough comparison of the different formulas with our new formulas is provided in Appendix D, here we give a brief overview.

Dimension $g = 1$ In the case of elliptic curves, there have been several recent works focusing on the development of radical isogeny formulas [14,12,50,23]. An advantage of our formulas is that inversions are naturally avoided by working with projective coordinates. This makes our 4- and 8-isogeny formulas more efficient than similar ones from the literature (see Table 3), while the 2-isogeny formulas for curves in Montgomery form are more efficient than ours (see Table 3).

Dimension $g = 2$ Prior to our work, explicit formulas for radical N-isogenies in dimension $g = 2$ have only been known for $N = 2$ and $N = 3$ using *Mumford coordinates*, see [10]. Our new formula for the case $N = 2$ in the theta model is not only more efficient, but also more generic. The reason is that Mumford coordinates are specific to working with Jacobians of hyperelliptic curves, hence previous formulas do not apply to reducible abelian surfaces. In contrast to that, our formulas in theta coordinates naturally include non-generic isogenies known as *gluing* and *splitting* isogenies. Furthermore our formulas for $N = 4$ are completely new, and compared to the known formulas for $N = 3$, also remarkably compact (see Table 4).

Dimension $g = 3$ The first explicit isogeny formulas for abelian threefolds appeared in a preprint by Ohashi, Onuki, Kudo, Yoshizumi and Nuida in 2024 [49]. In this concurrent work, the methods also rely on working with theta coordinates. The main difference is that we use the level-2 theta structure, while their work relies on the level-4 structure. The goal of their work is different, as they aim at walking in the entire 2-isogeny graph, while we we exclusively consider radical isogenies (i.e. the composition of two 2-isogenies yields a 4-isogeny). This property is not discussed in [49], but we believe that after potential modifications, their formulas would fit well in the framework of radical isogenies.
An advantage of our approach is that elements can be represented by merely 8 (projective) coordinates, while their approach requires handling 36 coordinates.[11] This results in more compact formulas when using the level-2 theta structure.

Comparison of Theta-CGL with other versions of the CGL hash function Essentially, there exist three different types of variants of the CGL hash functions based on: varying the degree ℓ of the radical isogenies, varying the dimension g, and using chains of 2^n-isogenies with $n \approx \log(p)$. The new variants presented in our paper fall into the first two categories. In contrast to previous works, we not only compare different variants from a theoretical perspective, but we also provide implementations in Rust that provide more insight on the actual benefits of different variants.

Varying the degree of the radical isogenies In the original paper by Charles, Goren and Lauter [16], the cryptographic hash function was introduced in a general way relying on walks in an ℓ-isogeny graph in dimension $g = 1$ for some integer ℓ. The case of $\ell = 2$ seems particularly interesting from an implementation perspective, since exactly one square-root has to be computed per isogeny which consumes exactly one message bit. More generally, an ℓ-isogeny requires the computation of an ℓ-th root, and consumes $\log_2(\ell)$ message bits. This offers potential for speed-ups which has been studied in [35] for $\ell = 4$. Our variants $\mathsf{Theta\text{-}CGL}_{g=1,\ell=4}$ and $\mathsf{Theta\text{-}CGL}_{g=1,\ell=8}$ follow the same spirit as this previous work and extend it by also considering a degree-8 variant. Our timings obtained in a cryptographically relevant setting clearly indicate the advantage of using higher degree isogenies.

Varying the dimension g The original CGL hash function works in the supersingular isogeny graph of elliptic curves. Another potential for speeding up the hash function is offered by generalizing this to higher dimensions which allows to reduce the size of the ground field. There exist two proposals in the literature

[11] Technically, there are 64 coordinates in level 4. However, the theta coordinates associated with odd indices of theta null points are all equal to zero.

working in dimension $g = 2$. These are a variant using radical 2-isogenies [62,13], and a variant using radical 3-isogenies [10]. Similar to this, we propose the variants $\mathsf{Theta\text{-}CGL}_{g=2,\ell=2}$ and $\mathsf{Theta\text{-}CGL}_{g=2,\ell=4}$ working with 2- and 4-isogenies, respectively. Finally, our version of $\mathsf{Theta\text{-}CGL}_{g=3,\ell=2}$ is the first variant of the CGL hash function that works in dimension 3. Our experiments convincingly backup the claims made in previous works that going to higher dimensions is computationally advantageous.

Using chains of 2^k-isogenies with $k \approx \log(p)$ The idea underlying this variant is to completely avoid radical isogeny computations and instead sample torsion subgroups of size $\approx \log(p)$ in a deterministic way. Assume $g = 1$ and we are given a message $m = (m_1, \ldots, m_n)$ of length $n \gg k \approx \log(p)$. Then this message would be divided into chunks of length k, i.e. $m = (c_1, \ldots, c_{n/k})$. Essentially, processing one chunk c_i requires sampling a 2^k-torsion point (deterministically, depending on c_i), and the computation of the 2^k-isogeny with this point as kernel generator. While the sampling of a 2^k-torsion point represents an additional cost necessary after each k bits in the chain, it is in general cheaper to compute an isogeny with a given kernel generator than using radical formulas. This idea was brought forward in [27], where it is shown that in dimension 1, this results in a significant speed-up.

This variant does not immediately fit in our framework of Theta-CGL, since we work with radical isogenies. However, a potential direction for future research could be to use the recent algorithm for computing 2^k-isogenies in dimension 2 in [21] which also works with level-2 theta coordinates. We note, that working in dimension 2, it is required to construct a symplectic 2^k-torsion basis after each 2^k-isogeny computation. It would be interesting to see how the trade-off compares to dimension 1. We note that another obstacle with this method would be finding a constant time method to deterministically construct both the torsion basis and symplectic basis. As these points are created from codomain data with rejection sampling, it is not obvious how to efficiently compute all necessary data without leaking any information about the internal state.

1.3 Future directions and applications

Our results show that the superspecial isogeny graph of abelian threefolds is better suited for instantiating an isogeny-based hash function than the lower dimensional graphs. This raises several natural questions.

New constructions in dimension 3 This is the first time, 3-dimensional isogeny graphs appear in a cryptographic construction. It would be interesting to see if there are other isogeny-based protocols that can benefit from working in higher dimensions. This idea is in some sense independent of the current trend for constructing *HD protocols* in the isogeny community such as SQISignHD[20]. In these constructions, higher dimensional isogenies are used to represent isogenies of dimension 1. In contrast to our hash function, these protocols do not make use of the full structure of higher dimensional isogeny graphs.

Increasing the dimension Why did we stop in dimension 3? A simple answer is that instantiating Theta-CGL in dimension 4, requires the availability of radical 2-isogeny formulas which is still an open problem in dimension 4. We believe that it should be possible to find such formulas by generalizing the techniques that we used to derive the formulas in dimension 3. However, the resulting formula for the step $\mathcal{T}_s$ will be significantly more complicated than in dimension 3. This comes from the fact that in dimension 4, out of the $2^g - 1 = 15$ square roots that need to be computed, only $g(g+1)/2 = 10$ square-roots may be chosen arbitrarily, and the remaining 5 need to be determined by explicit formulas (Lemma 3). It would be interesting to see, how a 4-dimensional version of Theta-CGL compares to our version in dimension 3.

Going even further, one can ask for the optimal dimension in the context of the CGL hash function. While deriving formulas for radical isogenies in dimension 5 seems even more difficult than in dimension 4, this might still be the more promising case to look at. The reason being that one could work with a 32-bit prime in order to achieve 128-bit security (see Subsection 5.2). This looks very promising from a computational perspective.

Other applications of our radical isogeny formulas For cryptanalytic purposes, it is important to have efficient formulas for navigating in isogeny graphs, for instance, in order to efficiently implement path finding algorithms. With our new formulas, the navigation in dimension 2 and 3 can be implemented more efficiently. While there have not been any cryptographic proposals in dimension $g = 3$ prior to our work, it is still important to navigate in this graph in order to implement efficient attacks on a 4-dimensional protocol. This is because the fastest attacks in higher dimensional isogeny graphs are so-called splitting attacks which rely on iteratively reducing the problem to lower dimensions.

Last but not least, the attacks on SIDH [11,44,57] have shown that it is essential to understand higher dimensional isogeny graphs in order to analyse the security of isogeny-based protocols - even if these rely on isogenies in dimension 1. However, there are still many open questions in the study of these graphs. Our new methods can be utilised to obtain numerical data on isogeny graphs, and thereby help to get insight on open problems in dimensions 2 and 3.

1.4 Notation

The following functions are used in the description of our algorithms.

- CondSelect(a, b, s) for $a, b \in \mathbb{F}_q$ and $s \in \{1, -1\}$ returns a if $s = 1$ and b otherwise in constant time
- CondNeg(a, s) for $a \in \mathbb{F}_q$ and $s \in \{1, -1\}$ sets $a = s \cdot a$ in constant time.
- Normalise(x) for $x \in \mathbb{P}^n$ returns $(1 : x_1/x_0 : \cdots : x_n/x_0)$
- MessageToChunks(m, g, k) given a message $m \in \{-1, 1\}^*$ returns n bitstrings $(m_1, \ldots, m_n)$ where $m_i \in \{-1, 1\}^{g(g+1)/2}$. The message length (in bits) is assumed to be a multiple of $kg(g+1)/2$.

For finite field operations, we use the following notation.

$$\begin{aligned} &\textbf{Sqrt} = \text{square root}, \quad \textbf{Crt} = \text{cube root}, \quad \textbf{Quart} = \text{fourth root}, \\ &\textbf{Eirt} = \text{eighth root}, \quad \textbf{M} = \text{multiplication}, \quad \textbf{S} = \text{squaring}, \\ &\textbf{M}_c = \text{scalar multiplication}, \quad \textbf{a} = \text{addition}, \quad \textbf{I} = \text{inversion}. \end{aligned}$$

Furthermore, we assume that we are given canonical functions $\sqrt[2^k]{\ } : \mathbb{F}_q \to \bar{\mathbb{F}}_q$ to compute 2^k-th roots of elements in the field of definition throughout the paper.

2 Preliminaries on 2-isogenies in the theta model

In this section, we provide some background on the computation of 2-isogenies of principally polarized (p.p.) abelian varieties in the theta model of level 2. This model is particularly well suited for this type of computations, for instance it is used in [18,21]. For more details on the theory, we refer to the survey on the computation of 2^n-isogenies by Robert [54].

2.1 Principally polarized abelian varieties and level-2 theta structures

In this manuscript, we uniquely focus on p.p. abelian varieties, i.e. abelian varieties endowed with a polarization describing an isomorphism with their duals. The easiest example of p.p. abelian varieties are elliptic curves.

Let A be a p.p. abelian variety of dimension g, then there exists a double covering $\theta^A\colon A \to \mathbb{P}^{2^g-1}$. Such a double covering is given by a *level-*2 *symmetric theta structure for* A. Let 0_A denote the identity point of A, then the value $\theta^A(0_A)$ is called *level-*2 *theta null point*. The double covering θ^A is generically determined by $\theta^A(0_A)$ via a descent to level 2 of the so-called (level-4) *Riemann relations*. This motivates the definition of the map

$$Th : (A, \theta^A) \mapsto \theta^A(0_A) \tag{2}$$

which assigns to a p.p. abelian variety (equipped with a level-2 theta structure) its theta null point. By [32,63], Th defines a local embedding of the moduli space $\mathcal{A}_g(2,4)$ into $\mathbb{P}^{2^g-1}$ at points corresponding to geometrically irreducible abelian varieties. For $g \leq 3$, it even defines a global embedding of the Satake compactification of this moduli space [46].

The coordinates of $\theta^A(0_A)$ are ordered as $(a_{0\ldots0} : \ldots : a_{1\ldots1})$, where the indices represent integers expressed in their binary form. In what follows, we consider these indices as elements $i_g \ldots i_1$ in $(\mathbb{Z}/2\mathbb{Z})^g$, hence defining a group law on the indices. To a level-2 theta structure, we associate the 2^g *theta functions* $\theta_{i_g\ldots i_1} : A \to \mathbb{A}_1$, which are the coordinate functions[12] of θ^A. Given a level-2 theta structure $\theta^A\colon A \to \mathbb{P}^{2^g-1}$, we say that $\theta^A(P)$ are the *theta coordinates* of a point $P \in A$.

The theta null point $Th(A, \theta^A) = \theta^A(0_A)$ is *not* an isomorphism invariant of A. Different level-2 theta structures for the same p.p. abelian variety are related by symplectic transformations. One such transformation is given by the Hadamard transform $\mathcal{H}$ which consists in multiplication by a Hadamard matrix

$$\mathcal{H} : \mathbb{P}^{2^g-1} \to \mathbb{P}^{2^g-1}, \quad \mathcal{H} : x \mapsto H_g \cdot x, \tag{3}$$

where H_g is the $2^g \times 2^g$ Hadamard matrix obtained from the inductive description

$$H_0 = (1), \quad H_{i+1} = \begin{pmatrix} H_i & H_i \\ H_i & -H_i \end{pmatrix} \text{ for } i > 0. \tag{4}$$

We say that $\widetilde{\theta}^A := \mathcal{H} \circ \theta^A$ is the *dual theta structure* of θ^A, and for a point $P \in A$, we say that $\widetilde{\theta}^A(P) = \mathcal{H}(\theta^A(P))$ are the *dual theta coordinates* of P. In

[12] Note that these are only well-defined up to multiplication by a constant which needs to be chosen compatibly.

particular $\widetilde{\theta}^A(0_A) = (\widetilde{a}_{0\dots0} : \dots : \widetilde{a}_{1\dots1})$ denotes the *dual theta null point*. The remaining symplectic transformations consist in scaling certain theta functions by a 4-th root of unity, or specific permutations of the indices, see Lemma 17.

2.2 Level-4 theta structures

For the statements of our results, it is enough to work with a level-2 theta structure. The level-4 structure will only be needed in the proofs. The latter defines a map $\theta^A : A \to \mathbb{P}^{2^{2g}-1}$, and we say that

$$\theta^A(0_A) = \left(a_{\left(\begin{smallmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \end{smallmatrix}\right)} : \dots : a_{\left(\begin{smallmatrix} 1 & \dots & 1 \\ 1 & \dots & 1 \end{smallmatrix}\right)} \right) \in \mathbb{P}^{2^{2g}-1},$$

is the *level*-4 *theta null point*. Similar as before, coordinates are indexed by elements $\left(\begin{smallmatrix} i_g & \dots & i_1 \\ j_g & \dots & j_1 \end{smallmatrix}\right) \in (\mathbb{Z}/2\mathbb{Z})^{2g}$. For indices $i = i_g \dots i_1$, $j = j_g \dots j_1$, we denote $\langle i, j \rangle := \sum_{k=1}^{g} i_k j_k$. We say that an index $\left(\begin{smallmatrix} i_g & \dots & i_1 \\ j_g & \dots & j_1 \end{smallmatrix}\right)$ is *even* if $\langle i, j \rangle = 0$, otherwise we say it is *odd*. Consequently, there are $2^{g-1}(2^g + 1)$ even and $2^{g-1}(2^g - 1)$ odd indices in total. Furthermore, the squares of the level-4 coordinates can be expressed in terms of the level-2 coordinates (see Lemma 18).
Generically, the even coordinates of a level-4 theta null point do not vanish and they describe geometric properties of the underlying abelian variety. More details for the cases $g = 1, 2, 3$ are provided in Appendix A.2.

2.3 The 2-torsion

For a positive integer N, the N-torsion of a p.p. abelian variety A is denoted by $A[N]$. This is a free $\mathbb{Z}/N\mathbb{Z}$-module of rank $2g$, equipped with the *Weil pairing*. For a symplectic basis $\mathcal{B} = (P_1, \dots, P_g, Q_1, \dots, Q_g)$ of $A[N]$, we denote

$$K_1 = \langle P_1, \dots, P_g \rangle, \quad K_2 = \langle Q_1, \dots, Q_g \rangle,$$

and call $A[N] = K_1 + K_2$ the *symplectic decomposition* of $A[N]$ defined by $\mathcal{B}$.
Given a level-2 theta structure, it defines a symplectic basis of $A[2]$ as described below (see also Example 19).

Lemma 1. *Let A be a p.p. abelian variety and $\theta^A : A \to \mathbb{P}^{2^g-1}$ a level-2 theta structure on A with $\theta^A(0_A) = (a_{0\dots0} : \dots : a_{1\dots1})$. Then in theta coordinates, $\mathcal{B} = (P_1, \dots, P_g, Q_1, \dots, Q_g)$ is a symplectic basis for $A[2]$, where*

$$P_i = (b_{0\dots0} : \dots : b_{1\dots1}), \quad \textit{with } b_{i_g\dots i_1} = a_{j_g\dots j_1} \textit{ and } j_k = \begin{cases} i_k & \textit{if } k \neq i \\ 1 - i_k & \textit{if } k = i \end{cases}$$

and

$$Q_i = (b_{0\dots0} : \dots : b_{1\dots1}), \quad \textit{with } b_{j_g\dots j_1} = (-1)^{j_i} a_{j_g\dots j_1}.$$

Proof. This is an explicit version of the 2-torsion basis defined in [54, §3]. □

We refer to the symplectic basis from Lemma 1 as the *canonical symplectic basis* associated to a level-2 theta null point. Moreover, we refer to the induced symplectic decomposition $A[2] = K_1 + K_2$ as the *canonical decomposition*. The choice of symplectic basis is natural [54, §3].

2.4 Isogenies of principally polarized abelian varieties

An *isogeny* between abelian varieties $\phi : A \to B$ is a surjective morphism with finite kernel. An isogeny $\phi : A \to B$ of p.p. abelian varieties, also needs to preserve the principal polarization. In this case, the kernel K of the isogeny ϕ is a maximal isotropic subgroup $K \subset A[N]$ for some N.

Definition 2. *An isogeny $\phi : A \to B$ of p.p. abelian varieties is called a good N-isogeny if the kernel K satisfies $K \simeq (\mathbb{Z}/N\mathbb{Z})^g$.*

In a symplectic decomposition $A[N] = K_1 + K_2$, the subgroups $K_1, K_2 \subset A[N]$ are both maximal isotropic and of rank g, hence they both define different N-isogenies with domain A.
Not every N-isogeny of p.p. abelian varieties is a good N-isogeny. But there always exist primes $\ell_1, \dots, \ell_n$ and good ℓ_i-isogenies ϕ_i (possibly over an extension) so that $\phi = \phi_n \circ \cdots \circ \phi_1$. This implies that a composition of good ℓ_i-isogenies does not necessarily results in a good N-isogeny. In Subsection 3.4, this issue is discussed in detail for the case of $\ell = 2$ and $N = 2^n$. Since we are only interested in good N-isogenies, from now on we assume all our N-isogenies are implicitly good.

2.5 Computing 2-isogenies

From now on, let A be a p.p. abelian variety equipped with a level-2 theta structure θ^A. We let $A[2] = K_1 + K_2$ be the canonical decomposition induced by $Th(A, \theta^A)$. Finally, let $\phi : A \to B$ be the 2-isogeny with kernel K_2, and θ^B the induced level-2 theta structure on B. In the following, we often denote $\phi : (A, \theta^A) \to (B, \theta^B)$. This isogeny can be described in terms of three simple operations

- $\mathcal{H}$: the Hadamard transform which consists in multiplication by a Hadamard matrix (see Subsection 2.1),
- $\mathcal{S}$: the coordinate-wise squaring map,
- $\mathcal{C}_c$: the scaling map for a vector c, i.e. $\mathcal{C}_c(x) = (x_{0\dots0}/c_{0\dots0} : \dots : x_{1\dots1}/c_{1\dots1})$.

In the isogeny, the scaling vector will be given by the dual theta null point of B (which has to be computed by other means). Concretely, the 2-isogeny $\phi : A \to B$ can be evaluated as $\mathcal{H} \circ \mathcal{C}_{\tilde{b}} \circ \mathcal{H} \circ \mathcal{S}$, that is

$$\theta^B(\phi(P)) = \mathcal{H} \circ \mathcal{C}_{\tilde{b}} \circ \mathcal{H} \circ \mathcal{S}(\theta^A(P)). \tag{5}$$

This description can be deduced from the duplication formulas (see Remark 20) and has already been used in different other works to compute 2-isogenies [18,54,21]. We remark that the maps $\mathcal{H}, \mathcal{S}, \mathcal{C}_c$ are all defined for projective coordinates. By slight abuse of notation, we use the same notation for the corresponding maps with affine input.

3 Radical 2-isogenies

Given $Th(A, \theta^A)$, the theta null point of a p.p. abelian variety with level-2 structure, our goal is to find $Th(B, \theta^B)$, so that there exists a 2-isogeny $\phi : A \to B$. The result will be a description of the isogeny as the composition of maps $\mathcal{H} \circ \mathcal{T}_s \circ \mathcal{H} \circ \mathcal{S}$. Note that this is similar to the description of 2-isogenies in the

previous section with the scaling map $\mathcal{C}_{\widetilde{b}}$ replaced by $\mathcal{T}_s$. The latter essentially consists in computing (compatible) square roots of the coordinates.
The first part of this Section is concerned with the abstract definition of $\mathcal{T}_s$. Then we provide explicit descriptions of $\mathcal{T}_s$ in dimensions $g = 1, 2, 3$. While the cases $g = 1, 2$ are immediately clear from the construction,[13] the case $g = 3$ requires more work. Finally, we explain the composition of radical 2-isogenies in order to obtain (good) 2^n-isogenies.

3.1 Definition of $\mathcal{T}_s$

Let $\phi : (A, \theta^A) \to (B, \theta^B)$ be a 2-isogeny with kernel K_2. Then

$$(\widetilde{b}^2_{0\dots0} : \dots : \widetilde{b}^2_{1\dots1}) = \mathcal{H} \circ \mathcal{S}((a_{0\dots0} : \dots : a_{1\dots1})),$$

are the squares of the (dual) theta null point of B with respect to the induced theta structure. The map $\mathcal{T}_s$ consists in going from $(\widetilde{b}^2_{0\dots0} : \dots : \widetilde{b}^2_{1\dots1})$ to $(\widetilde{b}_{0\dots0} : \dots : \widetilde{b}_{1\dots1})$. The next lemma characterizes the possible choices for such a map.

Lemma 3. *Let (A, θ^A) be a p.p. abelian variety with level-2 theta structure, $A[2] = K_1 + K_2$ the canonical decomposition (see Subsection 2.3), and let $\phi : (A, \theta^A) \to (B, \theta^B)$ be the isogeny with kernel K_2. Denote*

$$x = (x_{0\dots0} : \dots : x_{1\dots1}) = \mathcal{H} \circ \mathcal{S}(Th(A, \theta^A)) \in \mathbb{P}^{2^g-1}.$$

Further, let $I := \{i = i_g \dots i_1 \in (\mathbb{Z}/2\mathbb{Z})^g \mid \#\{i_j = 1\} = 0, 1, 2\}$ be the set of $g(g+1)/2+1$ indices $i = i_g \dots i_1$ with at most 2 non zero bits i_j. Assume $x_i \neq 0$ for $i \in I$. Let $y_{0\dots0} = x_{0\dots0}$ and

$$y_i, \quad \textit{with } y_i^2 = x_{0\dots0} x_i \textit{ for } i \in I \setminus \{0\}$$

be an arbitrary choice of square roots. Then there exist unique values y_i for $i \notin I$ so that $Th(B, \theta^{B'}) = (y_{0\dots0} : \dots : y_{1\dots1})$ for some (compatible) theta structure $\theta^{B'}$ on B.

Proof. First note that there necessarily exists a choice of square roots so that

$$(x_{0\dots0} : \sqrt{x_{0\dots0}x_{0\dots01}} : \dots : \sqrt{x_{0\dots0}x_{1\dots1}}) = \theta(0_B) \qquad (6)$$

for some level-2 theta structure θ on B. This follows from the description of the 2-isogeny $\phi : A \to B$ established in Subsection 2.5. It remains to show the following.

1. We may choose square roots for the $g(g+1)/2$ elements with indices in $I \setminus \{0\}$ arbitrarily.
2. The choice of the square roots in (1.) determines the signs of the remaining square roots, for a choice of level structure on B compatible with the one on A.

[13] Indeed, the computation of radical 2-isogenies in the theta model in dimensions $g = 1, 2$ is already outlined in [54, Appendix C].

We will repeatedly use Lemma 17 which characterizes equivalent level-2 theta structures for a given p.p. abelian variety. We note that only the second type of transformation, scaling by a 4-th roots of unity, is admissible here. This is clear, when all coordinates are distinct. In this case these are the only transformations that leave the squares of all theta functions invariant (if in addition the exponents are even). To understand the general case, one needs to consider the induced action on the symplectic decompositions $A[2] = K_{A,1} + K_{A,2}$ and $B[2] = K_{B,1} + K_{B,2}$. The compatibility condition implies that $K_{B,1} = \phi(K_{A,1})$, which is only preserved by using the second type of transformations.[14]
Any such transformation can be written as the composition of the following two types of transformations.

$$M_j : \theta_{i_g,\dots,i_1} \mapsto \begin{cases} \theta_{i_g,\dots,i_1} & \text{if } i_j = 0, \\ -\theta_{i_g,\dots,i_1} & \text{if } i_j = 1, \end{cases} \qquad \text{for } j \in \{1, \dots, g\},$$

$$M_{j,k} : \theta_{i_g\dots i_1} \mapsto \begin{cases} \theta_{i_g\dots i_1} & \text{if } i_j = 0 \text{ or } i_k = 0, \\ -\theta_{i_g\dots i_1} & \text{if } i_j = i_k = 1, \end{cases} \qquad \text{for } j \neq k \in \{1, \dots, g\}.$$

To prove (1.), assume that we are given a level-2 theta structure θ for B as in Equation 6, and let $\theta_{i_g\dots i_1}$ be the associated theta functions. In what follows, we demonstrate that the sign of any of the functions $\theta_{i_g\dots i_1}$ with index $i = i_g \dots i_1 \in I \setminus \{0\}$ may be changed without affecting the signs of the remaining functions with indices in $I \setminus \{i\}$. First, let $i = i_1 \dots i_g$ in I with exactly one nonzero index $i_j = 1$. The action of $M_j \cdot \prod_{k \neq j} M_{j,k}$ changes the sign of the function θ_i, but does not affect the signs of the theta function $\theta_{i'}$ with index $i' \in I \setminus \{i\}$. Now, let $i = i_1 \dots i_g$ in I with exactly two nonzero indices $i_j = i_k = 1$. The action of $M_{j,k}$ changes the sign of θ_i but does not affect the signs of the theta function $\theta_{i'}$ with indices $i' \in I \setminus \{i\}$.
We now prove (2.), the uniqueness of the remaining signs of θ, once a choice for the signs of the functions with indices in I has been fixed. Suppose that θ is an admissible choice of square-roots. And let C be a symmetric matrix acting as

$$\theta'_{i_g\dots i_1} = \zeta_4^{(i_g\dots i_1)C(i_g\dots i_1)^t} \cdot \theta_{i_g\dots i_1}.$$

which fixes the theta functions with indices in I. We denote $C = (c_{j,k})_{j,k}$. Since C does not affect $\theta_{i_g\dots i_1}$ with exactly one entry $i_j = 1$, we have that $\zeta_4^{c_{j,j}} = 1$ for all j. Furthermore, since C does not affect $\theta'_{i_g\dots i_1}$ with exactly two entries $i_j = i_k = 1$, we have

$$\zeta^{2c_{j,k}+c_{j,j}+c_{k,k}} = \zeta^{2c_{j,k}} = 1.$$

This implies that for a general index $i_g \dots i_1$, we have

$$\zeta_4^{(i_g\dots i_1)\cdot C\cdot(i_g\dots i_1)^t} = \zeta_4^{i_1^2 c_{1,1}+\dots+i_g^2 c_{g,g}+\sum_{j\neq k} 2c_{j,k} i_j i_k} = 1,$$

hence the corresponding theta function is unchanged. This completes the proof of Lemma 3. □

We can now formally define the operation $\mathcal{T}_s$.

[14] We refer to [55, Remarque 6.4.3] for more information on the relation between transformations and the torsion.

Notation 4. *Let $\phi : A \to B$ as in Lemma 3, and in particular $x = (x_{0\dots 0} : \cdots : x_{1\dots 1}) = \mathcal{H} \circ \mathcal{S}(Th(A, \theta^A))$ with $x_\ell \neq 0$ for $\ell \in I = \{i_g \dots i_1 \in (\mathbb{Z}/2\mathbb{Z})^g \mid \#\{i_j = 1\} = 0, 1, 2\}$. We denote*

$$\{\ell_1, \dots, \ell_{g(g+1)/2}\} := I \setminus \{0 \dots 0\}.$$

For a vector $s = (s_1, \dots, s_{g(g+1)/2}) \in \{-1, 1\}^{g(g+1)/2}$, we then denote

$$\mathcal{T}_s(x) = (y_{0\dots 0} : \cdots : y_{1\dots 1}), \quad \text{where } y_\ell = \begin{cases} x_{0\dots 0} & \text{if } \ell = 0 \dots 0, \\ s_{\ell_j} \sqrt{x_{0\dots 0} x_{\ell_j}} & \text{if } \ell = \ell_j \in I, \end{cases}$$

and the remaining $2^g - 1 - g(g+1)/2$ values y_ℓ with $\ell \notin I$ are uniquely determined as per Lemma 3.

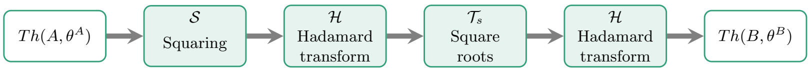


Fig. 1. A radical 2-isogeny

The definition of $\mathcal{T}_s$ allows us to describe radical 2-isogenies as the composition $\mathcal{H} \circ \mathcal{T}_s \circ \mathcal{H} \circ \mathcal{S}$. This is sketched in Figure 1. This definition can also be extended to the case where $x_\ell = 0$ for some $\ell \in I$, see Subsection 3.3.

In the cases $g = 1$ and $g = 2$, it holds that $2^g - 1 = g(g+1)/2$, hence we immediately obtain the following simple description for radical 2-isogenies.

Corollary 5 (Description of $\mathcal{T}_s$ for $g = 1, 2$). *Let A be a p.p. abelian variety of dimension g, and $\phi : A \to B$ an isogeny as in Notation 4.*

1. *For $g = 1$, we have*

$$\mathcal{T}_s : (x_0 : x_1) \mapsto (x_0 : s_1 \sqrt{x_0 x_1}).$$

2. *For $g = 2$, we have*

$$\mathcal{T}_s : (x_{00} : x_{01} : x_{10} : x_{11}) \mapsto (x_{00} : s_1 \sqrt{x_{00} x_{01}} : s_2 \sqrt{x_{00} x_{10}} : s_3 \sqrt{x_{00} x_{11}}).$$

Corollary 6. *Using Algorithm 1, the computation of a radical 2-isogeny in dimension $g = 1$ costs $1\mathbf{Sqrt} + 1\mathbf{M} + 2\mathbf{S} + 4\mathbf{a}$.*

Using Algorithm 2, the computation of a radical 2-isogeny in dimension $g = 2$ costs $3\mathbf{Sqrt} + 3\mathbf{M} + 4\mathbf{S} + 16\mathbf{a}$.

Algorithm 1 Radical 2-isogeny for $g = 1$

```
Input: A level-2 theta null point $Th(A, \theta^A) = (a_0 : a_1)$ and a choice bit $s \in \{-1, 1\}$
Output: A level-2 theta null point $Th(B, \theta^B) = (b_0 : b_1)$
$x_0, x_1 \leftarrow \mathcal{H} \circ \mathcal{S}(a_0, a_1)$                    ▷ $2\mathbf{S} + 2\mathbf{a}$
$x_1' \leftarrow x_0 \cdot x_1$                                                 ▷ $1\mathbf{M}$
$y_1 \leftarrow \sqrt{x_1'}$                                                    ▷ $1\mathbf{Sqrt}$
$y_0 \leftarrow x_0$
$y_1 \leftarrow \mathsf{CondNeg}(y_1, s)$
$b_0, b_1 \leftarrow \mathcal{H}(y_0, y_1)$                                     ▷ $2\mathbf{a}$
return $(b_0, b_1)$                  ▷ Total Cost: $1\mathbf{Sqrt} + 1\mathbf{M} + 2\mathbf{S} + 4\mathbf{a}$
```

Algorithm 2 Radical 2-isogeny for $g = 2$

```
Input: A level-2 theta null point $Th(A, \theta^A) = (a_0 : a_1 : a_2 : a_3)$ of an irreducible abelian
    surface $A$ and choice bits $s_i \in \{-1, 1\}^3$
Output: A level-2 theta null point $Th(B, \theta^B) = (b_0 : b_1 : b_2 : b_3)$
$x_0, x_1, x_2, x_3 \leftarrow \mathcal{H} \circ \mathcal{S}(a_0, a_1, a_2, a_3)$          ▷ $4\mathbf{S} + 8\mathbf{a}$
$x_1', x_2', x_3' \leftarrow x_0 \cdot x_1, x_0 \cdot x_2, x_0 \cdot x_3$                   ▷ $3\mathbf{M}$
$y_1, y_2, y_3 \leftarrow \sqrt{x_1'}, \sqrt{x_2'}, \sqrt{x_3'}$                            ▷ $3\mathbf{Sqrt}$
$y_0 \leftarrow x_0$
$y_1, y_2, y_3 \leftarrow \mathsf{CondNeg}(y_1, s_0), \mathsf{CondNeg}(y_2, s_1), \mathsf{CondNeg}(y_3, s_2)$
$b_0, b_1, b_2, b_3 \leftarrow \mathcal{H}(y_0, y_1, y_2, y_3)$                             ▷ $8\mathbf{a}$
return $(b_0, b_1, b_2, b_3)$        ▷ Total Cost: $3\mathbf{Sqrt} + 3\mathbf{M} + 4\mathbf{S} + 16\mathbf{a}$
```

3.2 Explicit description of $\mathcal{T}_s$ for $g = 3$

For abelian varieties of dimension $g > 2$, it holds that $2^g - 1 > g(g+1)/2$ and the description of $\mathcal{T}_s$ gets more complicated. In the following, we make this explicit for the case $g = 3$, where $2^g - 1 = 7 > 6 = g(g+1)/2$. Essentially, this means that we may choose 6 arbitrary square-roots, and the 7-th square-root is uniquely determined by this choice. An important ingredient for determining the last square-root is the following proposition which provides a relation among the coordinates of a level-2 theta null point.

Proposition 7. *Let $(a_{000}, \ldots, a_{111}) \in k^8$. We define*

$$(\beta_{00}, \ldots, \beta_{11}, \delta_{00}, \ldots, \delta_{11}) = \mathcal{H} \circ \mathcal{S}(a_{000}, \ldots, a_{111})$$

as well as $\gamma_i = 2 \cdot \sum_{j \in (\mathbb{Z}/2\mathbb{Z})^2} (-1)^{\langle i,j \rangle} a_{0j} a_{1j}$ for $i = i_2 i_1 \in (\mathbb{Z}/2\mathbb{Z})^2$, and set

$$R_1 = \prod_i \beta_i, \quad R_2 = \prod_i \gamma_i, \quad R_3 = \prod_i \delta_i.$$

Then $(a_{000} : \cdots : a_{111})$ is a level-2 theta null point of a p.p. abelian variety if and only if

$$R_1^2 + R_2^2 + R_3^2 - 2(R_1 R_2 + R_1 R_3 + R_2 R_3) = 0. \tag{7}$$

Proof. The statement essentially follows from the formula in [63, Example 1.4]. Here, we give some more details on this formula and deduce an explicit description in terms of the coordinates $(a_{000} : \cdots : a_{111})$.

In dimension $g = 3$, the map $Th : \mathcal{A}_g(2,4) \to \mathbb{P}^{2^g-1}$ as in Equation 2 defines an embedding of the moduli space of p.p. abelian threefolds into $\mathbb{P}^7$. The latter has dimension $g(g+1)/2 = 6 < 7$. Therefore the image $\mathrm{Im}(Th) \subset \mathbb{P}^7$ is a hypersurface. In the following, we show that this hypersurface is defined by Equation 7.

We note that the Riemann relations [47, §3] provide relations among the coordinates of level-4 theta null points. One such relation is the following.

$$\begin{aligned}\underbrace{a_{\binom{000}{000}} a_{\binom{000}{001}} a_{\binom{000}{010}} a_{\binom{000}{011}}}_{r_1} = {} & \underbrace{a_{\binom{100}{000}} a_{\binom{100}{001}} a_{\binom{100}{010}} a_{\binom{100}{011}}}_{r_2} \\ & + \underbrace{a_{\binom{000}{100}} a_{\binom{000}{101}} a_{\binom{000}{110}} a_{\binom{000}{111}}}_{r_3} .\end{aligned} \tag{8}$$

This implies

$$\prod_{0 \le i,j \le 1} (r_1 + (-1)^i r_2 + (-1)^j r_3) = R_1^2 + R_2^2 + R_3^2 - 2(R_1R_2 + R_2R_3 + R_1R_3) = 0, \tag{9}$$

with $R_i = r_i^2$ for $i = 1, 2, 3$.

The R_i, being products of squares of level-4 coordinates, may be expressed in terms of level-2 coordinates as per Lemma 18. In particular, for any $i = i_2 i_1 \in (\mathbb{Z}/2\mathbb{Z})^2$, we obtain

$$a_{\left(\begin{smallmatrix}0&0&0\\0&i_2&i_1\end{smallmatrix}\right)}^{\;2} = \beta_i, \quad a_{\left(\begin{smallmatrix}1&0&0\\0&i_2&i_1\end{smallmatrix}\right)}^{\;2} = \gamma_i, \quad a_{\left(\begin{smallmatrix}0&0&0\\1&i_2&i_1\end{smallmatrix}\right)}^{\;2} = \delta_i$$

with $\beta_i, \gamma_i, \delta_i$ as in the statement of the proposition. Substituting these expressions in Equation 9 yields the equation for the hypersurface. □

The relation from Proposition 7 allows us to explicitly describe $\mathcal{T}_s$.

Theorem 8. *Let A be a p.p. abelian variety of dimension $g = 3$, and $\phi : A \to B$ an isogeny as in Notation 4. In particular* [15]

$$(a_{000}, \dots, a_{111}) = Th(A, \theta^A), \quad (x_{000}, \dots, x_{111}) = \mathcal{H} \circ \mathcal{S}(Th(A, \theta^A))$$

and $I = \{\ell_1, \dots, \ell_6\} = (\mathbb{Z}/2\mathbb{Z})^3 \setminus \{000, 111\}$. We set

$$\begin{aligned} R_1 &= (16 \cdot a_{000}a_{001}a_{010}a_{011})^2, \quad R_3 = (16 \cdot a_{100}a_{101}a_{110}a_{111})^2, \\ T &= (x_{000}x_{100} - x_{001}x_{101} + x_{010}x_{110} - x_{011}x_{111})^2 \\ &\quad - 4\,(x_{000}x_{010}x_{100}x_{110} + x_{001}x_{011}x_{101}x_{111}). \end{aligned}$$

Then

$$\mathcal{T}_s : (x_{000} : \dots : x_{111}) \mapsto (x_{000} : s_1\sqrt{x_{000}x_{\ell_1}} : \dots : s_6\sqrt{x_{000}x_{\ell_6}} : y)$$

with y defined as follows.

[15] Note that all vectors are necessarily viewed as affine vectors in this theorem. In particular, we assume $x = \mathcal{H} \circ \mathcal{S}(Th(A, \theta^A))$ with $a = (a_{000}, \dots, a_{111})$.

(i) If $R_1 + R_3 - T \neq 0$, then

$$y = x_{000}^3 \cdot \frac{(R_1 + R_3 - T)^2 + 64\prod_{i \in (\mathbb{Z}/2\mathbb{Z})^3} x_i - 4R_1R_3}{16(R_1 + R_3 - T) \cdot \prod_{j=1}^6 s_j\sqrt{x_{000}x_{\ell_j}}}.$$

(ii) If $R_1 + R_3 - T = 0$, then

$$y = -x_{000}^3 \cdot \frac{2^6\prod_{i \in (\mathbb{Z}/2\mathbb{Z})^3} a_i}{\prod_{j=1}^6 s_j\sqrt{x_{000}x_{\ell_j}}}.$$

Proof. We outline the computations to find these formulas below. Symbolic verification of the computations is provided in our GitHub repository (https://github.com/GiacomoPope/ThetaCGL).

Our goal is to recover the vector $\widetilde{b} = Th(B, \widetilde{\theta}) = (\widetilde{b}_{000} : \cdots : \widetilde{b}_{111})$. Being a theta null point, the coordinates of $\widetilde{b}$ necessarily satisfy the relation described in Proposition 7. With the notation of the proposition, we have

$$(\beta_{00}, \ldots, \beta_{11}, \delta_{00}, \ldots, \delta_{11}) = \mathcal{H} \circ \mathcal{S}(\widetilde{b}) = \mathcal{H}(x_0 \cdot x) = 2^3 \cdot x_0 \cdot (a_{000}^2, \ldots, a_{111}^2),$$

where we used that $\mathcal{H} \circ \mathcal{H}$ is multiplication by 2^3. This implies that R_1, R_3 as defined in the theorem correspond to the $R_1^{\text{prop}}, R_2^{\text{prop}}$ in Proposition 7 up to multiplication by $2^6 \cdot x_0^4$, more precisely

$$R_1^{\text{prop}} = 2^6x_0^4 \cdot R_1, \quad R_3^{\text{prop}} = 2^6x_0^4 \cdot R_3.$$

Note that R_2^{prop} cannot be expressed in terms of the squares of the coordinates of $\widetilde{b}$. An explicit computation shows that

$$T = R_2^{\text{prop}} - \frac{2^7 \cdot \prod_{i \in (\mathbb{Z}/2\mathbb{Z})^3} \widetilde{b}_i}{x_{000}^4}.$$

This allows us to rewrite the relation from Proposition 7 in terms of R_1, R_3, T and $\prod_{i \in (\mathbb{Z}/2\mathbb{Z})^3} \widetilde{b}_i$. We use this to solve for $\widetilde{b}_{111}$. There are two cases to consider. If $R_1 + R_3 - T \neq 0$, there is a unique solution when solving for $\widetilde{b}_{111}$ which is given by the formula in the theorem.

In the second case, when $R_1 + R_3 - T = 0$, we find

$$\widetilde{b}_{111}^2 = x_{111}x_{000} = \left(\frac{2^6x_{000}^3\prod_{i \in (\mathbb{Z}/2\mathbb{Z})^3} a_i}{\prod_{j=1}^6 \sqrt{x_{000}x_{\ell_j}}}\right)^2.$$

This provides us with a formula for $\pm\widetilde{b}_{111}$. In order to choose the correct sign, it is necessary to go back to the proof of Proposition 7. With the notation of the proof, we have that

$$r_1^2 = R_1^{\text{prop}}, \, r_3^2 = R_3^{\text{prop}}, \quad \text{and} \quad r_1r_3 = 2^{12}\prod_{i \in (\mathbb{Z}/2\mathbb{Z})^3} a_i.$$

The last equality follows from the isogeny formula [47, Theorem 4].

Since $r_1 - r_2 - r_3 = 0$, we have $(r_1 - r_3)^2 - r_2^2 = 0$. Therefore,

$$0 = R_1 + R_3 - 2^{1+12}\prod_{i \in (\mathbb{Z}/2\mathbb{Z})^3} a_i - r_2^2.$$

Using $R_1 + R_3 - T = 0$ and $R_2 = T + 2^7 \cdot \frac{\prod_{i\in(\mathbb{Z}/2\mathbb{Z})^3} \widetilde{b}_i}{x_{000}^4}$, we obtain

$$\prod_{i\in(\mathbb{Z}/2\mathbb{Z})^3} \widetilde{b}_i = -2^6 \cdot x_{000}^4 \cdot \prod_{i\in(\mathbb{Z}/2\mathbb{Z})^3} a_i.$$

□

Theorem 8 is translated into Algorithm 3. We remark that this describes a constant time implementation, and moreover does not require any inversions.

Algorithm 3 LastSqrt for radical 2-isogenies for $g = 3$

```
Input: A level-2 theta null point $Th(A, \theta^A) = (a_0, \dots, a_7)$, the vector $(x_0, \dots, x_7) = \mathcal{H} \circ \mathcal{S}(Th(A, \theta^A))$ with $x_i \neq 0$ for all $i$, and $y_0, \dots, y_6$ so that $y_i^2 = x_0 x_i$ for $i = 0, \dots, 6$.
Output: $(y_0, \dots, y_7)$ so that $(y_0 : \dots : y_7) = Th(B, \widetilde{\theta}^B)$.
$a_{0123}, a_{4567} \leftarrow 16 \cdot a_0 \cdot a_1 \cdot a_2 \cdot a_3,\ 16 \cdot a_4 \cdot a_5 \cdot a_6 \cdot a_7$    ▷ $6\mathbf{M} + 2\mathbf{M}_c$
$R_1, R_3 \leftarrow a_{0123} \cdot a_{0123},\ a_{4567} \cdot a_{4567}$    ▷ $2\mathbf{S}$
$x_{04}, x_{15}, x_{26}, x_{37} \leftarrow x_0 \cdot x_4,\ x_1 \cdot x_5,\ x_2 \cdot x_6,\ x_3 \cdot x_7$    ▷ $4\mathbf{M}$
$x_{0246}, x_{1357} \leftarrow x_{04} \cdot x_{26},\ x_{15} \cdot x_{37}$    ▷ $2\mathbf{M}$
$t_0 \leftarrow (x_{04} - x_{15} + x_{26} - x_{37})^2 - 4 \cdot (x_{0246} + x_{1357})$    ▷ $1\mathbf{S} + 1\mathbf{M}_c + 5\mathbf{a}$
$T \leftarrow R_1 + R_3 - t_0$    ▷ $2\mathbf{a}$
$y \leftarrow \prod_{i=1}^{6} y_i$    ▷ $5\mathbf{M}$
$t_1 \leftarrow T^2 + 64 \cdot x_{0246} \cdot x_{1357} - 4 \cdot R_1 \cdot R_3$    ▷ $2\mathbf{M} + 1\mathbf{S} + 2\mathbf{M}_c + 2\mathbf{a}$
$t_2 \leftarrow 16 \cdot T \cdot y$    ▷ $1\mathbf{M} + 1\mathbf{M}_c$
$t_1' \leftarrow -a_{0123} \cdot a_{4567}$    ▷ $1\mathbf{M}$
$t_2' \leftarrow 4y$    ▷ $1\mathbf{M}_c$
$t_1, t_2 \leftarrow$ CondSelect$(t_1', t_1, T = 0)$, CondSelect$(t_2', t_2, T = 0)$
$y_0, \dots, y_6 \leftarrow t_2 \cdot (y_0, \dots, y_6)$    ▷ $7\mathbf{M}$
$y_7 \leftarrow x_0^3 \cdot t_1$    ▷ $2\mathbf{M} + 1\mathbf{S}$
return $y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7$    ▷ Total Cost: $30\mathbf{M} + 5\mathbf{S} + 7\mathbf{M}_c + 9\mathbf{a}$
```

As a result, we also obtain an algorithm for computing radical isogenies in dimension 3. This is summarized in Algorithm 4.

Corollary 9. *Using Algorithm 4, a radical 2-isogeny in dimesnion $g = 3$ can be computed in* $6\mathbf{Sqrt} + 36\mathbf{M} + 13\mathbf{S} + 7\mathbf{M}_c + 57\mathbf{a}$.

3.3 Description of $\mathcal{T}_s$ for vanishing theta null values

In the definition of $\mathcal{T}_s$ (Notation 4), we assumed that $x_\ell \neq 0$ for $\ell \in I$. When one or more of the coordinates are zero, one has to be more careful.
We remark that the coordinates of $x = (x_{0\dots0} : \dots : x_{1\dots1}) = \mathcal{H} \circ \mathcal{S}((a_{0\dots0} : \dots : a_{1\dots1}))$ are the squares of 2^g of the $2^{g-1}(2^g + 1)$ even theta values of level-4 (Lemma 18). The vanishing of the level-4 theta null values is related to the geometry of the abelian variety, see Subsection 2.2. For our main application, Theta-CGL (Section 5), we are only interested in irreducible varieties. In dimensions $g = 1$ and $g = 2$, this means that none of the level-4 theta null values vanishes. In dimension $g = 3$, it can happen that at most one of the x_i vanishes

Algorithm 4 Radical 2-isogeny for $g = 3$

```
Input: A level-2 theta null point $Th(A, \theta^A) = (a_0 : \cdots : a_7)$ of an irreducible abelian threefold
    $A$ and a choice bits $s_i \in \{-1, 1\}^6$
Output: A level-2 theta null point $Th(B, \theta^B) = (b_0 : \cdots : b_7)$
$x_0, \dots, x_7 \leftarrow \mathcal{H} \circ \mathcal{S}(a_0, \dots, a_7)$    ▷ $8\mathbf{S} + 24\mathbf{a}$
$i^* \leftarrow 7$    ▷ Set $i^*$ to the value $i$ such that $x_i = 0$ or 7 otherwise
for $i = 0, \dots, 6$ do
    $i^* \leftarrow$ CondSelect$(i, i^*, x_i = 0)$
end for
$x_{i^*}, x_7 \leftarrow x_7, x_{i^*}$    ▷ Ensure that if present, a zero coordinate is swapped to $x_7$
$y_0 \leftarrow x_0$
$x'_1, \dots, x'_6 \leftarrow y_0 \cdot (x_1, \dots, x_6)$    ▷ $6\mathbf{M}$
$y_1, \dots, y_6 \leftarrow \sqrt{x'_1}, \dots, \sqrt{x'_6}$    ▷ $6\mathbf{Sqrt}$
for $i = 1, \dots, 6$ do
    $y_i \leftarrow$ CondNeg$(y_i, s_{i-1})$
end for
$y_0, \dots, y_7 \leftarrow$ LastSqrt$(a_0 \dots, a_7, x_0, \dots, x_7, y_0, \dots, y_6)$    ▷ Cost: $30\mathbf{M} + 5\mathbf{S} + 7\mathbf{M}_c + 9\mathbf{a}$
$y_{i^*}, y_7 \leftarrow y_7, y_{i^*}$    ▷ Swap in the case that one of $x_i$ were zero
$b_0, \dots, b_7 \leftarrow \mathcal{H}(y_0, \dots, y_7)$    ▷ $24\mathbf{a}$
return $(b_0 : \cdots : b_7)$    ▷ Total Cost: $6\mathbf{Sqrt} + 36\mathbf{M} + 13\mathbf{S} + 7\mathbf{M}_c + 57\mathbf{a}$
```

even when the underlying p.p. abelian variety is irreducible. In this case it is the Jacobian of a hyperelliptic curve.

When A is the Jacobian of a hyperelliptic genus-3 curve, then exactly one of the 36 even theta coordinates vanishes. If this is *not* one of the coordinates corresponding to $(x_{000} : \cdots : x_{111}) = \mathcal{H} \circ \mathcal{S}(Th(A, \theta^A))$, then we can still apply Theorem 8. In the other case, let $\ell_{-1} \in (\mathbb{Z}/2\mathbb{Z})^3$ be the index of the vanishing coordinate and $\{i_0, \dots, i_6\} = (\mathbb{Z}/2\mathbb{Z})^3 \setminus \{i_{-1}\}$. We define

$$\mathcal{T}_s : (x_{000} : \cdots : x_{111}) \mapsto (y_{000} : \cdots : y_{111}),$$

where

$$y_k = \begin{cases} 0 & \text{if } k = \ell_{-1}, \\ x_k & \text{if } k = \ell_0, \\ s_i\sqrt{x_{\ell_{-1}} x_{\ell_i}} & \text{if } k = \ell_i, \ \text{with } i \in \{1, \dots, 6\}. \end{cases}$$

Note that this case still requires the computation of $6 = g(g+1)/2$ square roots, and none of the bits of $s = (s_1, \dots, s_6)$ is redundant. To see that our definition of $\mathcal{T}_s$ still results in an isogeny given by $\mathcal{H} \circ \mathcal{T}_s \circ \mathcal{H} \circ \mathcal{S}$, one can use an argument very similar to that of Lemma 3 with the indices of nonzero coordinates permuted appropriately.

Finally, we remark that it is possible to define $\mathcal{T}_s$ for the reducible cases as well. More details are given in Appendix B.

3.4 Composition of 2-isogenies

This part concerns the composition of 2-isogenies to obtain 2^n-isogenies. While this is a trivial task in dimension $g = 1$ (it only means avoiding dual isogenies at any step), the situation is more complicated in dimension $g > 1$. However, it turns out that the *good* composition of 2-isogenies is very natural in the theta model. In particular, we show that the isogeny computation sketched in Figure 2 is a (good) 2^n-isogeny.

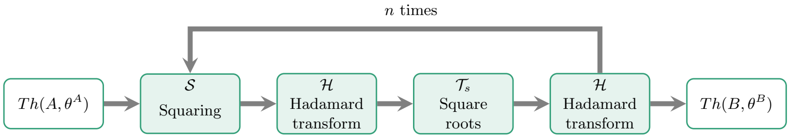


Fig. 2. A 2^n-isogeny

To make the potential obstructions in dimension $g > 1$ explicit, let $\phi : A \to B$ and $\psi : B \to C$ be 2-isogenies of p.p. varieties of dimension g. Then the kernel $K = \ker(\psi \circ \phi)$ is a 4-isotropic subgroup of A, and in particular

$$K = \ker(\psi \circ \phi) \simeq (\mathbb{Z}/4\mathbb{Z})^m \times (\mathbb{Z}/2\mathbb{Z})^{2k}, \quad \text{with } m + k = g.$$

In accordance with [13, Definition 2], we make the following definition, depending on the values m and k

Definition 10. *Let ϕ, ψ and m, k as above.*

- *We say that ψ is the* dual extension *of ϕ if $k = g, m = 0$.*
- *We say that ψ is a* bad extension *of ϕ if $0 < k, m < g$.*
- *We say that ψ is a* good extension *of ϕ if $k = 0, m = g$.*

As the naming suggests, we are only interested in *good* extensions of isogenies. Note that the composition $\psi \circ \phi$ is a good 4-isogeny if and only if ψ is a good extension of ϕ. Any given 2-isogeny $\phi : A \to B$, has exactly $2^{g(g+1)/2}$ different good extensions [10, Section 2.2].

In our framework, one automatically works with good extensions. And in particular, composing n radical 2-isogenies results in a good 2^n-isogeny. More formally, given an isogeny $\phi = \phi_n \circ \cdots \circ \phi_1$ described as the composition of radical 2-isogenies, where each $\phi_i : A_i \to A_{i+1}$, is described by $\mathcal{H} \circ \mathcal{T}_{s_i} \circ \mathcal{H} \circ \mathcal{S}$ for some $s_i \in \{-1, 1\}^{g(g+1)/2}$. Then for all $i = 1, \ldots, n-1$, the isogeny ϕ_{i+1} is a good extension of ϕ_i, and in particular, ϕ is a good 2^n-isogeny.

3.5 Relationship between sign choices and isogeny kernels

At each step i in the isogeny chain, one can choose $g(g+1)/2$ signs, which results in $2^{g(g+1)/2}$ different outcomes at each step. In the context of radical isogenies, one might expect that these corresponds to the different good extensions of the isogeny. However, the relation is more subtle.

First of all, we recall that the sign choice at step i has no influence on the kernel of the computed isogeny. By contrast, the choice of signs determines the next theta null point $Th(A_{i+1}, \theta^{A_{i+1}})$ which determines the kernel of the isogeny $\phi_{i+1} : A_{i+1} \to A_{i+2}$. In order to see the full effect of the sign choices on the kernel, one needs to go even one step further.

Lemma 11. *Let (A, θ^A) be a p.p. abelian variety equipped with a theta structure θ^A, and consider the 8-isogenies $\phi = \phi_3 \circ \phi_2 \circ \phi_1 : A \to B$ and $\phi' = \phi'_3 \circ \phi'_2 \circ \phi'_1 : A \to B'$, each computed as the composition of three 2-isogenies $\phi_i = \mathcal{H} \circ \mathcal{T}_{s_i} \circ \mathcal{H} \circ \mathcal{S}$ and $\phi'_i = \mathcal{H} \circ \mathcal{T}_{s'_i} \circ \mathcal{H} \circ \mathcal{S}$ respectively.*
Then if $\ker(\phi) = \ker(\phi')$, then $s_1 = s'_1$ (and conversely if $s_1 = s'_1, s_2 = s'_2$ then $\ker(\phi) = \ker(\phi')$). .

Proof. The theta structure θ^A implicitly fixes a symplectic decomposition $A[2] = K_1^A \oplus K_2^A$. In particular, by construction, we have that $\ker(\phi_1) = \ker(\phi'_1) = K_2^A$. Even if the isogenies $\phi_1 : A \to A_1$ and $\phi'_1 : A \to A_1$ coincide, the theta-null points obtained as output of $\mathcal{H} \circ \mathcal{T}_{s_1} \circ \mathcal{H} \circ \mathcal{S}$ and $\mathcal{H} \circ \mathcal{T}_{s'_1} \circ \mathcal{H} \circ \mathcal{S}$ are distinct since $s_1 \neq s'_1$; let θ^{A_1} and $\theta^{',A_1}$ denote the theta structures resulting from s_1 and s'_1, respectively.
By construction, the symplectic decompositions of the two torsion $A_1[2] = K_1^{A_1} \oplus K_2^{A_1}$ induced by $\theta^{A_1}$and $\theta^{',A_1}$ coincide. Now, let us denote by $T_1^{A_1} \oplus T_2^{A_1}$ the symplectic decomposition of the four-torsion inducing θ^{A_1} and by $T_1^{',A_1} \oplus T_2^{',A_1}$ the one inducing $\theta^{',A_1}$.
Let $r_1, \dots, r_g$ be the generators of $T_1^{A_1}$ corresponding to the theta structure θ^{A_1}, $t_1, \dots, t_g$ be the generators of $T_2^{A_1}$, and similarly $r'_1, \dots, r'_g$ and $t'_1, \dots, t'_g$ for $T_1^{',A_1}$ and $T_2^{',A_1}$. In particular $r_i = r'_i$.
Since $\theta^{A_1} \neq \theta^{',A_1}$, there exists $\widetilde{i} \in \{1, \dots, g\}$ such that $t'_{\widetilde{i}} = t_{\widetilde{i}} + 2r_{\widetilde{i}}$. This proves that the kernel of ϕ_3 and ϕ'_3 must be distinct. □

We provide a more thorough analysis of the situation in Appendix F.3. However, the statement of Lemma 11 is precise enough for the applications in this paper.

4 Radical 2^k-isogenies for small k

As we have seen in the last section, the formulas for radical 2-isogenies in the theta model are strikingly compact, especially in dimensions one and two. Naturally, this raises the question if this is also true for radical 2^k-isogeny formulas for some small $k > 1$. Here, we derive such formulas for radical 4- and 8-isogenies in dimension 1, as well as (almost) radical 4-isogenies in dimension 2. The proofs of our new formulas are elementary, and essentially only rely on applying the radical 2-isogeny formulas several times. Here, we provide pseudocode for the efficient valuations of our new formulas. The symbolic expressions, as well as detailed proofs can be found in Appendix C.
While the proofs are elementary, the general strategy to derive such formulas requires more advanced techniques. It relies on cubical arithmetic as presented in [59]. We outline the general strategy in Subsection 4.3.

4.1 Explicit formulas in dimension $g = 1$

In dimension $g = 1$, we found formulas for radical radical 4- and 8-isogenies (Algorithms 5 and 6).

Proposition 12. *On input a level-2 theta null point, $(a_0 : a_1) \in \mathbb{P}^1$ of an elliptic curve A, Algorithm 5 outputs the level-4 theta null point of an elliptic curve B, so that there exists a 4-isogeny $\phi : A \to B$ with kernel $K = \langle P \rangle$ and $\theta^A(P) = (1 : 0)$. The computational cost is $1\boldsymbol{Quart} + 1\boldsymbol{M} + 2\boldsymbol{S} + 1\boldsymbol{M}_c + 4\boldsymbol{a}$.*

Proof. See Appendix C.1 □

Algorithm 5 Radical 4-isogeny for $g = 1$

```
Input: A level-2 theta null point $Th(A, \theta^A) = (a_0 : a_1)$ and choice bits $s_1, s_2 \in \{-1, 1\}$
Output: A level-2 theta null point $Th(B, \theta^B) = (b_0 : b_1)$
$x_0, x_1 \leftarrow \mathcal{H} \circ \mathcal{S}(a_0, a_1)$    ▷ $2\mathbf{S} + 2\mathbf{a}$
$x_1' \leftarrow x_0 \cdot x_1$    ▷ $1\mathbf{M}$
$\lambda \leftarrow \sqrt[4]{x_1'}$    ▷ $1\mathbf{Quart}$
$\lambda \leftarrow \mathsf{CondNeg}(\lambda, s_1)$
$\lambda \leftarrow \mathsf{CondSelect}(\lambda, \zeta_4 \cdot \lambda, s_2)$    ▷ $1\mathbf{M}_c$
$b_0, b_1 \leftarrow \mathcal{H}(a_0, \lambda)$    ▷ $2\mathbf{a}$
return $(b_0, b_1)$    ▷ Total Cost: $1\mathbf{Quart} + 1\mathbf{M} + 2\mathbf{S} + 1\mathbf{M}_c + 4\mathbf{a}$
```

Proposition 13. *On input a level-2 theta null point, $(a_0 : a_1) \in \mathbb{P}^1$ of an elliptic curve A and the coordinates $\theta^A(P) = (u_0 : u_1)$ of an 8-torsion point satisfying $\theta^A(2 \cdot P) = (1 : 0)$, Algorithm 6 outputs the level-2 theta null point of an elliptic curve B, so that there exists an 8-isogeny $\phi : A \to B$ with kernel $K = \langle P \rangle$, as well as coordinates $\theta^B(Q) = (v_0 : v_1)$ pf an 8-torsion point Q with $\theta^B(2 \cdot Q) = (1 : 0)$.*
The computational cost is $1\boldsymbol{Eirt} + 7\boldsymbol{M} + 11\boldsymbol{S} + 4\boldsymbol{M}_c + 8\boldsymbol{a}$.

Proof. See Appendix C.1. □

Algorithm 6 Radical 8-isogeny for $g = 1$

```
Input: A level-2 theta null point $Th(A, \theta^A) = (a_0 : a_1)$, and $(u_0 : u_1)$ so that $(u_0 : u_1) = \theta^A(P)$ for some $P$ with $\theta^A(2 \cdot P) = (1 : 0)$; and choice bits $s_1, s_2, s_3 \in \{-1, 1\}$
Output: A level-2 theta null point $Th(B, \theta^B) = (b_0 : b_1)$ and coordinates of an 8-torsion point $(v_0 : v_1)$
$a_{00}, a_{01}, a_{11} \leftarrow a_0 \cdot a_0,\ a_0 \cdot a_1,\ a_1 \cdot a_1$    ▷ $1\mathsf{M} + 2\mathsf{S}$
$u_{00}, u_{01}, u_{11} \leftarrow u_0 \cdot u_0, u_0 \cdot u_1, u_1 \cdot u_1$    ▷ $1\mathsf{M} + 2\mathsf{S}$
$\lambda = \sqrt[8]{(u_{00}^2 - u_{11}^2)(u_{00}^2 + u_{11}^2)}$    ▷ $1\mathbf{Eirt} + 2\mathbf{S} + 1\mathbf{M} + 2\mathbf{a}$
$\lambda \leftarrow \mathsf{CondNeg}(\lambda, s_1)$
$\lambda \leftarrow \mathsf{CondSelect}(\lambda, \zeta_4 \cdot \lambda, s_2)$    ▷ $1\mathbf{M}_c$
$\lambda \leftarrow \mathsf{CondSelect}(\lambda, \zeta_8 \cdot \lambda, s_3)$    ▷ $1\mathbf{M}_c$
$\lambda_2, \lambda_4 \leftarrow \lambda^2, \lambda^4$    ▷ $2\mathsf{S}$
$b_0, b_1 \leftarrow \mathcal{H}(u_{00}, \lambda_2)$    ▷ $2\mathbf{a}$
$t \leftarrow u_{01} \cdot a_{01}$    ▷ $1\mathbf{M}_c$
$v_0 \leftarrow (t + t) \cdot b_1$    ▷ $1\mathbf{M} + 1\mathbf{a}$
$v_1 \leftarrow (a_{00} + a_{00}) \cdot u_{01}^2 + \lambda_4 \cdot a_{11} - \sqrt{2} \cdot \lambda \cdot t \cdot u_0$    ▷ $4\mathbf{M} + 1\mathbf{S} + 1\mathbf{M}_c + 3\mathbf{a}$
return $(b_0, b_1), (v_0, v_1)$    ▷ Total Cost: $1\mathbf{Eirt} + 8\mathbf{M} + 9\mathbf{S} + 4\mathbf{M}_c + 8\mathbf{a}$
```

4.2 Explicit formulas in dimension $g = 2$

In dimension $g = 2$, we provide a description of (almost) radical 4-isogenies (Proposition 14) along with pseudocode for the efficient evaluation of the formula (Algorithm 7). Here, with *almost radical*, we mean that the formula requires the computation of two fourth roots as well as two square roots. Optimally, one would expect a formula that relies on the computation of three fourth roots. The intuition behind our findings is explained in Subsection 4.3.

Proposition 14. *On input a level-2 theta null point $(a_{00} : a_{01} : a_{10} : a_{11}) \in \mathbb{P}^3$ of a p.p. abelian surface A, Algorithm 7 outputs the theta null point of a 4-isogenous p.p. abelian surface B. The 4-isogeny $\phi : A \to B$ is described by $\mathcal{H} \circ \mathcal{T}_{s_2} \circ \mathcal{H} \circ \mathcal{S} \circ \mathcal{H} \circ \mathcal{T}_{s_1} \circ \mathcal{H} \circ \mathcal{S}$ for some $s_1, s_2 \in \{\pm 1\}^3$.*
The computational cost is $2\boldsymbol{Quart} + 2\boldsymbol{Sqrt} + 14\boldsymbol{M} + 4\boldsymbol{S} + 5\boldsymbol{M}_c + 27\boldsymbol{a}$.

Proof. See Appendix C.2 □

Algorithm 7 (Almost) radical 4-isogeny for $g = 2$

Input: A level-2 theta null point $Th(A, \theta^A) = (a_0 : a_1 : a_2 : a_3)$ of an irreducible abelian surface A and bits $s_i \in \{-1, 1\}^6$

Output: A level-2 theta null point $Th(B, \theta^B) = (b_0 : b_1 : b_2 : b_3)$

```
$x_0, x_1, x_2, x_3 \leftarrow \mathcal{H} \circ \mathcal{S}(a_0, a_1, a_2, a_3)$    ▷ $4\mathbf{S} + 8\mathbf{a}$
$x_{01}, x_{02}, x_{13}, x_{23} \leftarrow x_0 \cdot x_1, x_0 \cdot x_2, x_1 \cdot x_3, x_2 \cdot x_3$    ▷ $4\mathbf{M}$
$y \leftarrow \sqrt{x_{01} \cdot x_{23}}$    ▷ $1\mathbf{Sqrt} + 1\mathbf{M}$
$y \leftarrow \mathsf{CondNeg}(y, s_0)$
$\alpha'_1, \alpha'_2 \leftarrow 4 \cdot (y + y + x_{01} + x_{23}), 4 \cdot (y + y + x_{02} + x_{13})$    ▷ $2\mathbf{M}_c + 5\mathbf{a}$
$\alpha_1, \alpha_2 \rightarrow \sqrt[4]{\alpha'_1}, \sqrt[4]{\alpha'_2}$    ▷ $2\mathbf{Quart}$
$\alpha'_3 \leftarrow 8 \cdot (x_{23} + y) \cdot ((x_{02} + y) \cdot x_{23} \cdot x_3 + (x_{13} + y) \cdot x_{23} \cdot x_2)$    ▷ $5\mathbf{M} + 1\mathbf{M}_c + 4\mathbf{a}$
$\alpha_3 \leftarrow \sqrt[2]{\alpha'_3}$    ▷ $1\mathbf{Sqrt}$
$\alpha_1, \alpha_2, \alpha_3 \leftarrow \mathsf{CondNeg}(\alpha_1, s_1), \mathsf{CondNeg}(\alpha_2, s_3), \mathsf{CondNeg}(\alpha_3, s_5)$
$\alpha_1, \alpha_2 \leftarrow \mathsf{CondSelect}(\alpha_1, \zeta_4 \cdot \alpha_1, s_2), \mathsf{CondSelect}(\alpha_2, \zeta_4 \cdot \alpha_2, s_4)$    ▷ $2\mathbf{M}_c$
$t \leftarrow x_{23} \cdot \alpha_1 \cdot \alpha_2$    ▷ $2\mathbf{M}$
$y_0 \leftarrow t \cdot (a_0 + a_0)$    ▷ $1\mathbf{M} + 1\mathbf{a}$
$y_1, y_2 \leftarrow t \cdot \alpha_1, t \cdot \alpha_2$    ▷ $2\mathbf{M}$
$y_3 \leftarrow \alpha_3$
$b_0, b_1, b_2, b_3 \leftarrow \mathcal{H}(y_0, y_1, y_2, y_3)$    ▷ $8\mathbf{a}$
return $(b_0, b_1, b_2, b_3)$    ▷ Total Cost: $2\mathbf{Quart} + 2\mathbf{Sqrt} + 14\mathbf{M} + 4\mathbf{S} + 5\mathbf{M}_c + 26\mathbf{a}$
```

4.3 General strategy

The correct framework to find ℓ-radical isogeny formulas is given by the cubical arithmetic [59]. Let (A, Θ_A) be a p.p. abelian variety, and $K = \langle P_1, \ldots, P_g \rangle$ be a totally isotropic subgroup of $A[\ell]$. Let $B = A/K$, and $\phi : (A, \Theta_A) \to (B, \Theta_B)$ the corresponding ℓ-isogeny, and $\widetilde{\phi}$ be its contragredient isogeny, so that $\widetilde{\phi}^* \Theta_B \sim \ell \Theta_A$. By the general theory of multiradical isogenies [10,58], we know that a

choice of basis $Q_1, \ldots, Q_g$ in $\widetilde{K} = \ker \widetilde{\phi}$ such that $\widetilde{\phi}(Q_i) = P_i$ is given by a choice of $g(g+1)/2$ appropriate ℓ-roots of the Tate pairings $e_{T,\ell}(P_i, P_j)$.
Now, as explained in [59, § 6.2.3], we can use the cubical arithmetic on A in level n to represent part of the action of the theta group of level $n\ell$ on B, whenever ℓ is coprime to n. More precisely, a choice of basis of a symmetric level subgroup in $G(\ell\Theta_B)$ above $\widetilde{K}$ is the same as a choice of symmetric cubical points $\widetilde{P_i}$ and $\widetilde{P_i + P_j}$. Furthermore, the obstructions to finding these symmetric cubical points correspond to the étale torsors $x^\ell = \zeta_{ij}$ where the values ζ_{ij} are given by the Tate pairings as above [58].
We thus obtain the following strategy to find ℓ-radical formulas using theta coordinates of level n: we compute symmetric cubical points $\widetilde{P_i}$ and $\widetilde{P_i + P_j}$ by solving the equations $x^\ell = \zeta_{ij}$, this involve $g(g+1)/2$ choice of ℓ-th roots. Theta coordinates are well adapted to the cubical arithmetic, since the Riemann relations give cubical relations [59, § 4.8]. Using this cubical arithmetic, we then obtain a theta null point of level ℓn on $(B, \ell n \Theta_B)$ [30,55], and we use a variant of the change of level formulas from [43] to obtain the level n theta coordinates on $(B, n\Theta_B)$ of the points Q_i, which form the basis of the next kernel on B.
Unfortunately, for the purpose of this article we want to compute 2^k-radical isogeny formulas, while working with theta functions of level 2. We can only apply the above strategy partially: first, we need to start with a basis $(P_1, \ldots, P_g)$ compatible with our existing theta structure on A. Then by [56, Remark 2.10.3], we can leverage a choice of g 2^k-roots and $g(g-1)/2$ choice of 2^{k-1}-roots into a level 2^k theta null point on $B' = A/K[2^{k-1}]$. Note that while we have access to the 2^k-torsion (and partial information on the 2^{k+1}-torsion, see appendix F) on B', $A \to B'$ only gives a 2^{k-1}-isogeny. The question becomes whether we can, replacing our $g(g-1)/2$ choices of 2^{k-1} roots into 2^k-roots, determine enough 2^{k+1}-torsion information on B' such that it descends on enough 2^k-torsion information on $B = A/K$ via the 2-isogeny $B' \to B$.
The easiest case to tackle is the case $g = 1$, because the extra information on the 2^{k+1}-torsion encoded by a level 2^k-theta is easier to describe. The above strategy is how we found the radical 4- and radical 8-isogeny formulas in dimension 1. In section 4.1, we gave a self contained proof of the formulas. Still, we feel that the above description of the global framework to find radical isogeny formulas using cubical arithmetic explains why our formulas do not come out of nowhere, and could in principle apply to compute 2^k-radical isogenies in dimension 1.
It also explains why radical 4-isogeny formulas are harder to find in higher dimension. As explained, in dimension 2 we can take two 4-th roots and one square root to obtain a theta null point of level 4 on $B' = A/K[2]$. On B' we have full information on the 4-torsion and partial information on the 8-torsion. Using an extra square root we can add enough information on this 8-torsion to obtain our 4-torsion points Q_1, Q_2 on $B = A/K$ via the 2-isogeny $B' \to B$. However, this means that the 4-radical formulas from Proposition 14 use two fourth roots and two square roots rather than three fourth roots.

5 Application to cryptography: Theta-CGL

A *cryptographic hash function* is a hash function $H : \{0,1\}^* \to \{0,1\}^r$, which on input a message $m \in \{0,1\}^*$, outputs a bit string of fixed length r. The security of the hash function relies on the following two assumptions:

- *Preimage resistance:* Given an element $x \in \{0,1\}^r$, it is hard to find a message $m \in \{0,1\}^*$ with $H(m) = x$.
- *Collision resistance:* It is hard to find two distinct messages $m \neq m' \in \{0,1\}^*$ with $H(m) = H(m')$.

In 2006, Charles, Goren and Lauter [16] proposed a cryptographic hash function (known as the CGL hash function) based on taking a deterministic walk in the supersingular 2-isogeny graph of elliptic curves over a finite field $\mathbb{F}_{p^2}$. To make this more precise, fix some supersingular elliptic curve $E_0/\mathbb{F}_{p^2}$ (together with an incoming isogeny $\phi_{-1} : E_{-1} \to E_0$), and let $m = (m_0, \dots, m_{n-1}) \in \{0,1\}^n$ be the input message. This message determines a walk

$$E_0 \to E_1 \to \cdots \to E_n,$$

where at each step i, the isogeny $\phi_i : E_i \to E_{i+1}$ is determined by the bit m_i. Note that the 2-isogeny graph is 3-regular, and imposing the condition to construct a non-backtracking walk, there are exactly two options at each vertex. The output of the hash function is then given by the j-invariant of the last elliptic curve in the walk, that is $H(m) = j(E_n)$.

Generalizations of the CGL hash function working in isogeny graphs of two dimensional p.p. abelian varieties have been proposed in [62] and [13].

In this section, we provide a new framework for instantiating CGL-type hash functions using our description of radical isogenies in level-2 theta coordinates. This can be viewed as a simple and uniform framework for the original CGL hash function as well as its generalization to dimension 2, and it also allows us to describe a generalization to dimension 3 for the first time.

5.1 Description of our hash function

In the setup, one first chooses parameters

$$(g, \ell) \in \{(1,2), (1,4), (1,8), (2,2), (2,4), (3,2)\}.$$

Here, g stands for the dimension, and $\ell = 2^k$ for the degree of the isogenies. Then a p.p. abelian variety A of dimension g, and a level-2 theta null point $Th(A, \theta^A) = (a_{0\dots0} : \cdots : a_{1\dots1})$ are provided as public parameters. The hash function takes as input a message $m \in \{-1,1\}^{n \cdot k \cdot g(g+1)/2}$ for some arbitrary value n. [16] This message is divided into chunks $m_1, \dots, m_n$ of bit length $k \cdot g(g+1)/2$. Then at each step i, a 2^k-isogeny is computed, where the choice of 2^kth roots at each step is determined by m_i. The output consists of the (normalized) theta null point of the codomain of the final isogeny.[17] A description is provided in Algorithm 8. This uses the algorithms $\mathsf{Radical}_{g,\ell}$ from Sections 3 and 4.

[16] In the implementation, one of course takes $m \in \{0,1\}^{n \cdot g(g+1)/2}$. We intentionally use $\{-1,1\}^{n \cdot g(g+1)/2}$ in the description, since message bits will correspond to signs in our hash function.

[17] In our implementation of the hash function, we normalized the coordinates. This was done for two reasons: to make the output more compact, and as a precaution against the possibility that unnormalized coordinates may leak information on the action on differentials which might provide information on the isogeny.

Algorithm 8 Theta-CGL$_{g,\ell=2^k}$

```
Input: A level-2 theta null point $Th(A, \theta^A)$ and a message $m$
Output: A normalised level-2 theta null point $Th(A', \theta^{A'})$ with $a_{0\dots0} = 1$
(m_0, ..., m_n) ← MessageToChunks(m, g, k)        ▷ $m_i \in \{-1, 1\}^{kg(g+1)/2}$
for $i = 0$ to $n$ do
    $\theta(0_A) \leftarrow \mathsf{Radical}_{g,\ell}(\theta(0_A), m_i)$
end for
$\theta(0'_A) \leftarrow \mathsf{Normalise}(\theta(0_A))$
return $\theta(0'_A)$
```

Underlying isogeny graph To allow for an efficient implementation of the hash function, we instantiate it in the *superspecial* 2*-isogeny graph* $\mathit{\Gamma}_g(2; p)$ in dimension $g = 1, 2, 3$, respectively. The vertices of this graph are isomorphism classes of superspecial p.p. abelian varieties of dimension g defined over $\bar{\mathbb{F}}_p$, and the edges represent 2-isogenies. We denote $\mathcal{S}_{g,p} = V(\mathit{\Gamma}_g(2; p))$ for the vertex set which has size $O(p^{g(g+1)/2})$ (see Theorem 25 for a more precise statement). Furthermore, it is known that the graph $\mathit{\Gamma}_g(2; p)$ is an expander graph [2]. The output of a random walk on an expander graph with N vertices tends to the uniform distribution after $O(\log(N))$ steps, hence we require that our hash function takes at least $O(\log p)$ steps. For a discussion on the constant factor, we refer to Remark 26.

Base field As the base field for our computations, we choose a finite field $\mathbb{F}_{p^2}$ with $p \equiv 3 \pmod 4$. This guarantees that any element $A \in \mathcal{S}_{g,p}$ has an $\mathbb{F}_{p^2}$-model whose 4-torsion is fully in $\mathbb{F}_{p^2}$. It follows that all 2-isogenies are $\mathbb{F}_{p^2}$-rational, and the level-2 theta null points are $\mathbb{F}_{p^2}$-rational as well. The size of the prime depends on the dimension g and the required security level λ. We let $\log(p) \approx 2\lambda$ if $g = 1$, $\log(p) \approx \lambda$ if $g = 2$, and $\log(p) \approx \lambda/2$ if $g = 3$. A detailed security analysis justifying these values is provided in Subsections 5.2 and 5.3.

Initial vertex The public parameter $Th(A, \theta^A)$ is a level-2 theta null point of an element $A \in \mathcal{S}_{g,p}$. We require that the endomorphism ring of A is unknown, cf. Remark 15. Generating such an element is still an open problem, even in dimension $g = 1$, therefore we rely on a trusted setup to generate the initial vertex in the graph.

For the trusted setup, we suggest to start with an arbitrary vertex $A_0 \in \mathcal{S}_{g,p}$ (potentially with known endomorphism ring), perform a sufficiently long random walk in the graph $\mathcal{S}_{g,p}$ starting at A_0, and outputting the last vertex in this walk. For this random walk, one may use the radical 2-isogeny formulas developed in this work. For $g = 1$, more sophisticated methods are available to compute an elliptic curve with unknown endomorphism ring in a trusted setup [4].

Admissible messages Thus far, we have presented a function that performs a walk in the 2-isogeny graph that makes $k \cdot g(g+1)/2$ choices for each step of the walk. This results in a compression function that takes in fixed-length inputs. Hence, we need to perform a *domain extension* to handle arbitrary-length inputs. To do so, we will use a variant of the *Merkle–Damgård transform* with the

compression function that have a set of theta coordinates and $k \cdot g(g+1)/2$ bits as input and outputs a set of theta coordinates. Since the compression function is preimage resistant (c.f. Theorem 27) and collision resistant (c.f. Theorem 16), we can conclude that the hash function obtained after the Merkle–Damgård transform will also retain those properties [40, Thm. 5.4].
The padding scheme that we propose for this hash function is to append a `1` to the message before padding with `0` and finally adding the 64-bit representation of the message length. This is in line with the padding scheme used in SHA-2, for example.

Isomorphism invariants versus theta null points Strictly speaking, we do not work with isomorphism classes of p.p. abelian varieties. The reason is that we represent the elements by their theta null point which are not invariant under isomorphism.
A practical advantage for using the theta null point as input and output, is that it is a value that we naturally compute during the execution of the hash function. Moreover, the theta null point provides us with a uniform representation of p.p. abelian varieties, whereas isomorphism invariants depend on the type of object. For instance in dimension $g = 2$, one would need to distinguish between Jacobians of hyperelliptic curves which can be represented by Igusa invariants, and products of elliptic curves represented by a pair of j-invariants.
On the other hand, an advantage for using isomorphism invariants would be that it makes the output size smaller. An isomorphism class is determined by $g(g+1)/2$ invariants, while the (normalized) theta null point consists of $2^g - 1$ values. We remark that this argument only becomes interesting in dimension $g = 3$ and higher.

5.2 Preimage resistance of $\mathsf{CGL\text{-}Hash}_g$

In our setting, preimage resistance essentially boils down to the problem of finding isogenies between p.p. abelian varieties. To make this more precise, let $Th(A_1, \theta^{A_1})$ be the public theta null point used in the hash function, let $m \in \{-1, 1\}^{kg(g+1)/2}$ and $Th(A_2, \theta^{A_2}) = \mathsf{Theta\text{-}CGL}(m)$. This means that there exists a 2^k-isogeny $\phi : A_1 \to A_2$. Finding the pre-image m in the context of this hash function requires finding such a 2^k-isogeny ϕ, hence solving Problem 1.

Problem 1. *Given two superspecial p.p. abelian varieties of dimension g, A_1 and A_2 defined over $\mathbb{F}_{p^2}$, find a 2^k-isogeny $\phi : A_1 \to A_2$.*

The best known algorithm to find isogenies between two superspecial p.p. abelian varieties of dimension $g > 1$ is known as the splitting attack by Costello and Smith [19] which has classical complexity $O(p^{g-1})$. This is significantly faster than generic attacks on the path finding problem in a graph which have complexity $O(p^{g(g+1)/4})$ for all $g \geq 1$. We remark that the splitting attack might not directly yield a solution to Problem 1, since the output is most likely not a 2^k-isogeny. However, it provides a useful bound for the security of the problem. A summary of the state-of-the-art can be found in Appendix E.2.
In conclusion, to reach (classical) security level of λ bits, we need to work over a finite field $\mathbb{F}_{p^2}$ with p a prime of size λ/c_g where $c_g = \max(1/2, g-1)$ as suggested in Subsection 5.1.

5.3 Collision resistance of Theta-CGL

Collision resistance in our setting essentially corresponds to finding two different 2^k-isogenies between two p.p. abelian varieties. This is captured in Problem 2.

Problem 2. *Given a superspecial p.p. abelian variety of dimension g, A_1 defined over $\mathbb{F}_{p^2}$, find a 2^k-isogeny $\phi : A_1 \to A_2$ and a $2^{k'}$-isogeny $\phi' : A_1 \to A_2'$, such that $A_2 \cong A_2'$ and $\ker(\phi) \neq \ker(\phi')$.*

First, we note that Problem 2 can be reduced to Problem 1. The idea underlying this reduction is to compute a random 2^k-isogeny $\phi : A_1 \to A_2$ for some large k and then use an oracle to solve Problem 1 on input A_1, A_2. This returns an isogeny $\phi' : A_1 \to A_2$, and with overwhelming probability ϕ_1 and ϕ_2 are different isogenies. Generically, the best known method to solve Problem 2 is indeed to solve Problem 1.
Furthermore, we note that a solution to Problem 2, that is isogenies $\phi, \phi' : A_1 \to A_2$, provides us with the non-scalar endomorphism $\hat{\phi} \circ \phi' \in End(A_1)$. We remark that finding a non-scalar endomorphism of a random supersingular elliptic curve is a common hardness assumption in isogeny-based cryptography, since [51] shows that it can be reduced to the problem of computing the entire endomorphism ring of an elliptic curve is the basis of essentially all of isogeny-based cryptography.

Remark 15. Recall that we assume that the starting p.p. abelian variety in our hash function has unknown endomorphism ring. In the light of Problem 2, this means that a priori no information on the endomorphism ring of A_1 is available. We argue that this is necessary, since knowledge of the endomorphism ring may be used to construct collisions. Explicit methods for such a construction are known for elliptic curves, [4], but have not been considered in the higher dimensional setting $g > 1$ yet. In our implementation, we began with an abelian variety which is isomorphic to E^g, hence one is able to represent the endomorphism ring of A abstractly using the endomorphism ring of E.

Theorem 16. *The hash function $\mathsf{Theta\text{-}CGL}_{g,\ell}$ with $g \in \{1,2,3\}$ is collision resistant if Problem 2 is hard.*

Proof. Let $Th(A, \theta^A)$ be the public theta null point used in the hash function. In order to obtain a collision, the adversary has to find two distinct messages $m \neq m'$ so that $\mathsf{Theta\text{-}CGL}(m) = \mathsf{Theta\text{-}CGL}(m') = Th(B, \theta^B)$ for some $B \in \mathcal{S}_{g,p}$. By construction of the hash function, this means that a collision provides us with two isogenies $\phi, \phi' : A \to B$. If $\ker(\phi) \neq \ker(\phi')$, then this provides us with a solution to Problem 2 and we are done. From now on assume that $\ker(\phi) = \ker(\phi')$. Furthermore, we assume that $\ell = 2$. The statement for $\ell = 2^k$ follows immediately from this case. This means, that we may decompose the isogenies into chains of 2-isogenies

$$\phi = \phi_t \circ \cdots \circ \phi_1, \quad \phi' = \phi_t' \circ \cdots \circ \phi_1',$$

we necessarily have that $t = t'$ and moreover ϕ_i and ϕ_i' are equal up to isomorphism for all i.

We write $m = (m_1, \dots, m_k)$ and $m' = (m_1, \dots, m_k)$. Let $i \in \{1, \dots, k\}$ be minimal with $m_i \neq m'_i$, and denote $\phi_i, \phi'_i : A_i \to A_{i+1}$. At this step, the hash function performs the computations

$$Th(A_{i+1}, \theta^{A_{i+1}}) = \mathcal{H} \circ \mathcal{T}_{m_i} \circ \mathcal{H} \circ \mathcal{S}(Th(A_i, \theta^{A_i})),$$
$$Th(A_{i+1}, \theta^{A_{i+1}\prime}) = \mathcal{H} \circ \mathcal{T}_{m'_i} \circ \mathcal{H} \circ \mathcal{S}(Th(A_i, \theta^{A_i})),$$

respectively. We remark that the input theta null point at this step $Th(A_i, \theta^{A_i})$ coincides. There are two cases to distinguish now.

Case 1: $Th(A_{i+1}, \theta^{A_{i+1}}) = Th(A_{i+1}, \theta^{A_{i+1}\prime})$. In other words, the output at this step coincides in both cases despite the fact that $m_i \neq m'_i \in \{-1, 1\}^{g(g+1)/2}$. We write

$$m_i = (s_1, \dots, s_{g(g+1)/2}), \quad m'_i = (s'_1, \dots, s'_{g(g+1)/2}),$$

and let $s_\ell \neq s'_\ell$ for some index ℓ. Going back to the definition of $\mathcal{T}_s : x \mapsto y$, this implies $x_\ell = 0$. As explained in Subsection 3.3, this can only happen if A_i is a non-generic vertex of the isogeny graph. The prime p is chosen large enough so that reducible elements are only encountered with probability $O(2^{-\lambda})$, hence these types of non-generic objects can be excluded. In the case $g = 3$, there is one more type of non-generic objects, Jacobians of hyperelliptic genus-3 curves, where exactly one theta coordinate vanishes. However, in this case the definition of $\mathcal{T}_s$ still ensures that $6 = g(g+1)/2$ bits are consumed without any redundancies (Subsection 3.3).

Case 2: $Th(A_{i+1}, \theta^{A_{i+1}}) \neq Th(A_{i+1}, \theta^{A_{i+1}\prime})$. We will show that this contradicts the assumption that $Th(A_{k+1}, \theta^{k+1}) = Th(A_{k+1}, \theta^{A'_{k+1}})$ (which is the output of the hash function). We know from Lemma 11 that a different choice of bits at step i results in a non-isomorphic codomain in step $i + 2$. Since this cannot be the case, we necessarily have $i = k - 1$. So at step $i + 1$, we have two different theta null points sandwiched between two identical theta null points at step i and $k + 1$ respectively; but the compatibility of the theta structures at each isogeny step impose them to have the same level structure, hence the same theta constant, which is a contradiction. A more direct argument is as follows. First compute the possible preimages of the theta null point $Th(B, \theta^B) = (b_{0\dots0} : \dots : b_{1\dots1})$ under the last isogenies ϕ_k and ϕ'_k. This computation reveals that the coordinates of $Th(A_k, \theta^{A_k})$ and $Th(A_k, \theta^{A_k\prime})$ only differ by signs. On the other hand, we also know that the coordinates of the dual theta null point of A_k with respect to θ^{A_k} and $\theta^{A_k\prime}$ also only differ by signs. The latter condition is obtained by looking at the isogeny $\phi_{k-1} : A_{k-1} \to A_k$. These two conditions imply $Th(A_k, \theta^{A_k}) = Th(A_k, \theta^{A_k\prime})$ which contradicts our assumption. □

Finally, we discuss two collision attacks that did not appear in Theorem 16, since these are naturally avoided by our setup.

Collisions through trivial cycles The graph $\varGamma_g(p; 2)$ with $g > 1$ naturally has many small (trivial) cycles that can lead to collisions if the hash function is not designed carefully.

To make this more precise, let $G \subset A_1[2^k]$ be a maximal isotropic subgroup, and $\phi : A_1 \to A_2$ be the isogeny with kernel G. Then ϕ can be decomposed into a chain of 2-isogenies, $\phi = \phi_k \circ \dots \circ \phi_1$. If $G \cong (\mathbb{Z}/2^k\mathbb{Z})^g$, that is ϕ is a 2^k-isogeny, then the decomposition is unique. On the contrary, if the rank of G is greater than g, then the decomposition is not unique, hence there are different paths in the isogeny graph connecting A_1 and A_2 [31, Prop. 4].

It was noted in [31] that this made the first proposal of a CGL hash function in dimension 2 [62] insecure. In [13] this problem is fixed by restricting the paths in the CGL hash functions to 2^k-isogenies, hence only allow kernels of rank g. This is naturally the case in hash function as well (Subsection 3.4).

Trivial collisions with isomorphism invariants Finally, we would like to point out that replacing the output of Theta-CGL with isomorphism invariants will lead to trivial collisions in the hash function. The reason is that in terms of isomorphism classes, the first isogeny computation is already completely determined by the input. However, it is easy to avoid such collisions by altering the definition of Theta-CGL slightly. This is explained in Appendix F.

6 Implementation and Experimental Results

We have implemented the $\mathsf{Theta\text{-}CGL}_{g,\ell}$ hash function for

$$(g, \ell) = \{(1, 2), (1, 4), (1, 8), (2, 2), (2, 4), (3, 2)\},$$

in both the programming language Rust and computer algebra system SageMath.

The Rust implementation has been written with cryptographic use in mind and so is written to both be performant as well as “constant-time” (information about the bits consumed by the hash function are not leaked through timing side-channels). In contrast, the purpose of the SageMath implementation is to offer an educational resource for our 2^k-radical isogeny formulas and additionally contains symbolic proofs of various statements in this paper.

In this section, we outline parameter selection for dimensions one, two and three. In Table 2 we present the performance of $\mathsf{Theta\text{-}CGL}_{g,\ell}$ for all variants by benchmarking the number of kilobits (1000 bits) hashed per second (Kbps).

The Rust and SageMath code is made available at the following GitHub link: https://github.com/GiacomoPope/ThetaCGL.

6.1 Picking explicit parameters

Picking the characteristic for our fields is motivated firstly by the security considerations presented in Section 5.1, where to reach $\lambda = 128$ bits of security we require the base field to have a characteristic of size 256, 128, 64-bits in dimensions one, two and three respectively. Secondly, our choice is made to optimize the cost of the arithmetic in $\mathbb{F}_{p^2}$. The only restriction coming from the hash function is $p \equiv 3 \mod 4$ so that the theta null point and the isogenies are all $\mathbb{F}_{p^2}$-rational. Apart from this, the focus is on picking a base field to allow for efficient arithmetic and 2^k-roots in $\mathbb{F}_{p^2}$.

Note that the restriction $p \equiv 3 \mod 4$ further allows for the extension field $\mathbb{F}_{p^2}$ to have modulus $x^2 + 1$ for faster arithmetic [42]. Additionally, if we pick $p \equiv 15 \mod 16$, then we can efficiently compute square-roots, fourth-roots and eighth-roots in $\mathbb{F}_p$ with a single exponentiation. This (not-necessary) congruence condition is the only restriction we put in place, meaning there is an abundance of possible primes for the base field to be picked from with the size required to meet our security goals.

At the level of arithmetic in the base field $\mathbb{F}_p$, we can ask for efficient modular arithmetic by looking for primes with particularly efficient modular reduction, allowing for faster addition and multiplications. For dimension two, we are particularly lucky as we are in the best-case scenario with a Mersenne prime $p = 2^{127} - 1$, where modular reduction in $\mathbb{F}_p$ is particularly efficient as $2^{127} \equiv 1 \mod p$ and we represent elements with two 64-bit words. For dimensions one and three we are less lucky, but are still able to pick primes to allow for optimized modular reduction. This is a very common problem for elliptic curve and isogeny-based cryptography and there is is a wealth of references on the various choices we could make [6,3,60]. From experimentation, we found that for dimension one the "Montgomery friendly" [3] prime $p = 5 \cdot 2^{248} - 1$[18] represented by four 64-bit words had the best performance whereas for dimension three we picked the Solinas prime $p = 2^{64} - 2^8 - 1$, which is represented by a single 64-bit word. Details on our implementation of finite field arithmetic, we refer to Appendix G. The main complexities are summarized in Table 1.

Table 1. The characteristic picked for dimensions one, two and three together with the arithmetic cost of square-roots, inversions and Legendre symbols in the base field. For the 256-bit prime, inversion and Legendre symbols are computed using the binary GCD algorithm [52] and so is cheaper in practice than the arithmetic cost reported.

Characteristic	Square-root ($\mathbb{F}_p$)	Inversion ($\mathbb{F}_p$)	Legendre Symbol ($\mathbb{F}_p$)
$5 \cdot 2^{248} - 1$	$1\mathbf{M} + 248\mathbf{S}$	$\leq 13\mathbf{M} + 249\mathbf{S}$	$\leq 13\mathbf{M} + 248\mathbf{S}$
$2^{127} - 1$	$125\mathbf{S}$	$10\mathbf{M} + 127\mathbf{S}$	$10\mathbf{M} + 126\mathbf{S}$
$2^{64} - 2^8 - 1$	$7\mathbf{M} + 61\mathbf{S}$	$9\mathbf{M} + 63\mathbf{S}$	$9\mathbf{M} + 62\mathbf{S}$

All finite field arithmetic has been written to be constant time to avoid side-channels in our hash-function. This engineering restriction is carried through to the hash-function itself with no secret dependent branching or table look ups. Bits of the message which are hashed are consumed in constant time by using conditional negation and conditional swaps.

6.2 Performance

All variants of the $\mathsf{Theta\text{-}CGL}_{g,\ell}$ hash function described in this paper have been implemented in Rust. The reported times in Table 2 were measured using an Intel Core i7-9750H CPU with a clock-speed of 2.6 GHz with turbo-boost disabled. The Rust code was compiled using version `1.80.0-nightly` with the additional flag `-C target-cpu=native` to allow the use of Intel specific opcodes, especially useful for the finite field arithmetic.[19]

[18] One could write $5 = 2^2 + 1$ and then this prime is also in Solinas form $p = 2^{250} + 2^{248} - 1$.

[19] For the interested reader, details on how to reproduce the benchmarks are included with the implementation code https://github.com/GiacomoPope/ThetaCGL.

Table 2. Performance of the $\mathsf{Theta\text{-}CGL}_{g,\ell}$ function measured by kilobits per second (Kbps) hashed in dimensions one, two and three. Times were recorded on an Intel Core i7-9750H CPU with a clock-speed of 2.6 GHz with turbo-boost disabled.

$\mathsf{Theta\text{-}CGL}_{g,\ell}$	$\ell = 2$ (Kbps)	$\ell = 4$ (Kbps)	$\ell = 8$ (Kbps)	Output (bytes)
$g = 1$	81	126	149	64
$g = 2$	261	374	—	192
$g = 3$	597	—	—	448

Experimentally, we see that there is a moderate performance benefit from working with 2^k-radical isogenies for larger k, but a more significant speed-up is found by working in higher dimension. The reason for this is two-fold. Firstly, the underlying cost of 2^k-roots in $\mathbb{F}_{p^2}$ are $k+3$ exponentiations with $O(\log(p))$ squares in $\mathbb{F}_p$, which becomes cheaper as the size of p diminishes. Secondly, as the size of p decreases, the number of limbs required to represent a single element in $\mathbb{F}_p$ is reduced. A single multiplication requires $O(n^2)$ word operations and on a 64-bit machine we represent elements of $\mathbb{F}_p$ with four, two and one word(s) respectively for dimensions one, two and three. As a result there is a significant speed-up for the base arithmetic for the smaller prime.

One could push further and consider 2-radical isogenies in dimensions four and above, but we conjecture that the resulting hash-function would be less efficient than our dimension three proposal. Firstly, the cost of the arithmetic itself in dimension four would only be marginally faster as the prime would have approximately 43 bits to match our $\lambda = 128$ bit security goal, and so would still require elements represented using a single `u64` word. More costly though is that points in dimension four require 16 coordinates and although $g(g+1)/2 = 10$ bits could be consumed for each 2-radical isogeny, we would need many more multiplications in each step after computing 10 roots to ensure the other coordinates are correct (contrast in dimension three where we need about 30 multiplications to fix only one additionally root). Additionally, as the dimension increases, one is more likely to walk through the graph onto "edge-case" nodes which would require special treatment and a constant time implementation would have to gracefully handle cases where there are zero, one or many zero-coordinates in the null point of the variety. However, as these radical formulae are not currently known, future work in this area may show that moving to higher dimension is indeed more efficient.

References

1. Gora Adj and Francisco Rodríguez-Henríquez. Square root computation over even extension fields. Cryptology ePrint Archive, Paper 2012/685, 2012.
2. Yusuke Aikawa, Ryokichi Tanaka, and Takuya Yamauchi. Isogeny graphs on superspecial abelian varieties: Eigenvalues and connection to bruhat-tits buildings, 2022.
3. Jean Claude Bajard and Sylvain Duquesne. Montgomery-friendly primes and applications to cryptography. *Journal of Cryptographic Engineering*, 11(4):399–415, Nov 2021.

4. Andrea Basso, Giulio Codogni, Deirdre Connolly, Luca De Feo, Tako Boris Fouotsa, Guido Maria Lido, Travis Morrison, Lorenz Panny, Sikhar Patranabis, and Benjamin Wesolowski. Supersingular curves you can trust. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 405–437. Springer, 2023.
5. Daniel Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. In *Algorithmic Number Theory Symposium (ANTS XIV)*, volume 4, pages 39–55. Mathematical Sciences Publishers, 2020.
6. Joppe W. Bos and Simon J. Friedberger. Arithmetic considerations for isogeny-based cryptography. *IEEE Transactions on Computers*, 68(7):979–990, 2019.
7. Bradley W. Brock. *Superspecial curves of genera two and three*. Phd thesis, Princeton University, 1993.
8. Nils Bruin, E Flynn, and Damiano Testa. Descent via (3, 3)-isogeny on jacobians of genus 2 curves. *Acta Arithmetica*, 3(165):201–223, 2014.
9. Wouter Castryck and Thomas Decru. CSIDH on the surface. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 111–129, Paris, France, April 15–17, 2020. Springer, Cham, Switzerland.
10. Wouter Castryck and Thomas Decru. Multiradical isogenies. Cryptology ePrint Archive, Paper 2021/1133, 2021.
11. Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.
12. Wouter Castryck, Thomas Decru, Marc Houben, and Frederik Vercauteren. Horizontal racewalking using radical isogenies. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part II*, volume 13792 of *Lecture Notes in Computer Science*, pages 67–96, Taipei, Taiwan, December 5–9, 2022. Springer, Cham, Switzerland.
13. Wouter Castryck, Thomas Decru, and Benjamin Smith. Hash functions from superspecial genus-2 curves using Richelot isogenies. *Journal of Mathematical Cryptology*, 14(1):268–292, 2020.
14. Wouter Castryck, Thomas Decru, and Frederik Vercauteren. Radical isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 493–519, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland.
15. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Cham, Switzerland.

16. Denis Xavier Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *Journal of Cryptology*, 22(1):93–113, January 2009.
17. Maria Corte-Real Santos, Craig Costello, and Benjamin Smith. Efficient $(3, 3)$-isogenies on fast kummer surfaces. In *Algorithmic Number Theory Symposium (ANTS XVI)*. Mathematical Sciences Publishers, 2024.
18. Craig Costello. Computing supersingular isogenies on Kummer surfaces. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 428–456, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Cham, Switzerland.
19. Craig Costello and Benjamin Smith. The supersingular isogeny problem in genus 2 and beyond. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, volume 12100 of *Lecture Notes in Computer Science*, pages 151–168. Springer, 2020.
20. Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQIsignHD: New dimensions in cryptography. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 3–32, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.
21. Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert. An algorithmic approach to (2,2)-isogenies in the theta model and applications to isogeny-based cryptography. *IACR Cryptol. ePrint Arch.*, page 1747, 2023.
22. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 64–93, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland.
23. Thomas Decru. Radical $\sqrt[N]{\text{élu}}$ isogeny formulae. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part V*, volume 14924 of *Lecture Notes in Computer Science*, pages 107–128, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.
24. Thomas Decru and Sabrina Kunzweiler. Efficient computation of $(3^n, 3^n)$-isogenies. In *AfricaCrypt*, pages 53–78. Springer, 2023.
25. Christina Delfs and Steven D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mho_{\mathrm{p}}$. *Des. Codes Cryptogr.*, 78(2):425–440, 2016.
26. Augusto Jun Devegili, Michael Scott, and Ricardo Dahab. Implementing cryptographic pairings over barreto-naehrig curves. Cryptology ePrint Archive, Paper 2007/390, 2007.
27. Javad Doliskani, Geovandro CCF Pereira, and Paulo SLM Barreto. Faster cryptographic hash function from supersingular isogeny graphs. In *International Conference on Selected Areas in Cryptography*, pages 399–415. Springer, 2022.
28. Régis Dupont. *Moyenne arithmético-géométrique, suites de Borchardt et applications.* PhD thesis, PhD thesis, École polytechnique, 2006.
29. Torsten Ekedahl. On supersingular curves and abelian varieties. *Mathematica Scandinavica*, 60:151–178, 1987.

30. Jean-Charles Faugère, David Lubicz, and Damien Robert. Computing modular correspondences for abelian varieties. *Journal of Algebra*, 343(1):248–277, 10 2011.
31. E. Victor Flynn and Yan Bo Ti. Genus two isogeny cryptography. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 286–306, Chongqing, China, May 8–10, 2019. Springer, Cham, Switzerland.
32. Eberhard Freitag. *Siegelsche modulfunktionen*, volume 254. Springer-Verlag, 1983.
33. Oded Goldreich. Randomized methods in computation (lecture notes by oded goldreich).
34. Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.
35. Yuji Hashimoto and Koji Nuida. Efficient construction of cgl hash function using legendre curves. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 106(9):1131–1140, 2023.
36. Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43(04):439–562, August 2006.
37. Tomoyoshi Ibukiyama, Toshiyuki Katsura, and Frans Oort. Supersingular curves of genus two and class numbers. *Compositio Mathematica*, 57(2):127–152, 1986.
38. Jun-ichi Igusa. Theta functions. *Grundlehren der mathematischen Wissenschaften*, 1972.
39. Bruce W Jordan, Allan G Keeton, Bjorn Poonen, Eric M Rains, Nicholas Shepherd-Barron, and John T Tate. Abelian varieties isogenous to a power of an elliptic curve. *Compositio Mathematica*, 154(5):934–959, 2018.
40. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* Chapman & Hall/CRC, 2nd edition, 2014.
41. David Kohel, Kristin E. Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion -isogeny path problem. *LMS J. Comput. Math.*, 17(Theory):418–432, 2014.
42. Patrick Longa. Efficient algorithms for large prime characteristic fields and their application to bilinear pairings. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):445–472, Jun. 2023.
43. David Lubicz and Damien Robert. Fast change of level and applications to isogenies. *Research in Number Theory (ANTS XV Conference)*, 9(1), 12 2022.
44. Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 448–471, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.
45. Nikhil S. Mande and Ronald de Wolf. Tight bounds for the randomized and quantum communication complexities of equality with small error. *Electron. Colloquium Comput. Complex.*, TR21-113, 2021.
46. Riccardo Salvati Manni. On the projective varieties associated with some subrings of the ring of thetanullwerte. *Nagoya mathematical journal*, 133:71–83, 1994.

47. David Mumford. On the Equations Defining Abelian Varieties. I. *Inventiones Mathematicae*, 1, 12 1966.
48. Ryo Ohashi and Hiroshi Onuki. Construction of a hash function using superspecial abelian 3-folds, 2024. Symposium on Cryptography and Information Security. https://www.iwsec.org/scis/2024/program.html.
49. Ryo Ohashi, Hiroshi Onuki, Momonari Kudo, Ryo Yoshizumi, and Koji Nuida. Computing richelot isogeny graph of superspecial abelian threefolds, 2024.
50. Hiroshi Onuki and Tomoki Moriya. Radical isogenies on montgomery curves. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13177 of *Lecture Notes in Computer Science*, pages 473–497, Virtual Event, March 8–11, 2022. Springer, Cham, Switzerland.
51. Aurel Page and Benjamin Wesolowski. The supersingular endomorphism ring and one endomorphism problems are equivalent. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 388–417, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.
52. Thomas Pornin. Optimized binary gcd for modular inversion. Cryptology ePrint Archive, Paper 2020/972, 2020.
53. Fried Jul Richelot. De transformatione integralium abelianorum primi ordinis commentatio. 1837.
54. Damien Robert. A note on optimising 2^n-isogenies in higher dimension.
55. Damien Robert. *Theta functions and cryptographic applications*. PhD thesis, Université Henri-Poincarré, Nancy 1, France, 7 2010.
56. Damien Robert. Efficient algorithms for abelian varieties and their moduli spaces, 6 2021.
57. Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 472–503, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.
58. Damien Robert. The geometric interpretation of the tate pairing and its applications. 2 2023.
59. Damien Robert. Fast pairings via biextensions and cubical arithmetic. *Cryptology ePrint Archive*, 2024.
60. Michael Scott. Elliptic curve cryptography for the masses: Simple and fast finite field arithmetic. Cryptology ePrint Archive, Paper 2024/779, 2024.
61. Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, Dordrecht, second edition, 2009.
62. Katsuyuki Takashima. Efficient algorithms for isogeny sequences and their cryptographic applications. *Mathematical Modelling for Next-Generation Cryptography: CREST Crypto-Math Project*, pages 97–114, 2018.
63. Bert van Geemen and Gerard van der Geer. Kummer varieties and the moduli spaces of abelian varieties. *American Journal of Mathematics*, 108(3):615–641, 1986.

A More details on 2-isogenies in the theta model

Here, we collect several known results on the level-2 theta structure of a p.p. abelian variety as well as on the computation of 2-isogenies. The section complements Section 2, where we provided a brief overview on the topic.

A.1 Symplectic transformations

Given a p.p. abelian variety A, there exist different level-2 theta structures $\theta^A : A \to \mathbb{P}^{2^g-1}$. The different theta structures are related by so-called symplectic transformations. These are described in the following lemma.

Lemma 17. *Let (A, θ) be a p.p. abelian variety equipped with a level-2 theta structure. We denote $(\theta_{0\dots0} : \dots : \theta_{1\dots1})$ for the corresponding theta functions. A different level-2 theta structure θ' for A can be obtained by applying one or more of the following transformations.*

- ***Hadamard**: The Hadamard transform $\mathcal{H}$ is applied to $\theta = (\theta_{0\dots0} : \dots : \theta_{1\dots1})$, i.e. $\theta' = \mathcal{H}(\theta)$.*
- ***Scaling by a 4-th root of unity** ζ_4: For some symmetric matrix C with entries in $\mathbb{Z}$, a theta function $\theta_{i_g\dots i_1}$ is scaled by $\zeta_4^{(i_g\dots i_1)^t C(i_g\dots i_1)}$, i.e.*

$$\theta'_{i_g\dots i_1} = \zeta_4^{(i_g\dots i_1)C(i_g\dots i_1)^t} \cdot \theta_{i_g\dots i_1}.$$

- ***Permutation**: The theta functions are permuted, where the permutation is defined by a bijective linear map $P : (\mathbb{Z}/2\mathbb{Z})^g \to (\mathbb{Z}/2\mathbb{Z})^g$ acting on the indices, i.e.*

$$\theta'_{i_g\dots i_1} = \theta_{P(i_g\dots i_1)}.$$

Moreover, all level-2 theta structures for A are obtained in this way.

Proof. This follows from the discussion in [54, Appendix B] applied to the case $n = 2$. We remark that in this case the transformations in the statement of the lemma correspond to symplectic transformations of the underlying 4-torsion basis. □

A.2 Level-4 theta structures

We recall from Subsection 2.2 that a level-4 structure of an abelian variety defines a map $\theta^A : A \to \mathbb{P}^{2^{2g}-1}$, and we say that

$$\theta^A(0_A) = \left(a_{\binom{0\,\dots\,0}{0\,\dots\,0}} : \dots : a_{\binom{1\,\dots\,1}{1\,\dots\,1}} \right) \in \mathbb{P}^{2^{2g}-1},$$

is the *level-4 theta null point*. Coordinates are indexed by elements $\binom{i_g\,\dots\,i_1}{j_g\,\dots\,j_1} \in (\mathbb{Z}/2\mathbb{Z})^{2g}$. Further, we recall that there are $2^{g-1}(2^g+1)$ even and $2^{g-1}(2^g-1)$ odd indices.

In analogy with the level-2 case, one can define *level-4 coordinates* and *level-4 functions*. The lemma below provides us with the relation between level-2 and level-4 theta structures.

Lemma 18. *Let θ^A be a level-2 theta structure on a p.p. abelian variety A. Then there exists a level-4 theta structure θ^A such that the associated level-2 and level-4 theta functions satisfy the following relation.*[20]
For all points $P \in A$ and pairs of indices $i = i_g \dots i_1$ and $j = j_g \dots j_1$ in $(\mathbb{Z}/2\mathbb{Z})^g$, it holds that

$$\left(\theta_{\binom{i}{j}}(P)\right)^2 := \sum_{t=t_g\dots t_1} (-1)^{\langle i,t\rangle}\theta_t(P)\theta_{t+j}(P).$$

Proof. This is a consequence of the duplication formula [38, Theorem 2, p. 139], see for example [28, Proposition 5.6]. □

Note that by setting $P = 0_A$, Lemma 18 can be used to compute the squares of the level-4 theta null values in terms of a level-2 theta null point. Furthermore, one easily verifies that the squares of the odd theta null values vanish.
Generically, the even coordinates of a level-4 theta null point do not vanish and they describe geometric properties of the underlying abelian variety. More details are provided below for the cases $g = 1, 2, 3$. Here, we assume that such p.p. abelian varieties are defined over an algebraically closed field.

$g = 1$ In dimension 1, p.p. abelian varieties are necessarily irreducible. These p.p. abelian varieties are elliptic curves.

$g = 2$ In dimension 2, p.p. abelian varieties (surfaces) come in two flavors: products of two elliptic curves and Jacobians of genus-2 hyperelliptic curves [31, Theorem 1]. The reducible case, i.e. the product of two elliptic curves, satisfies the property of having exactly one zero coordinate for the level-4 theta-null point [28, Proposition 6.5]. In the irreducible case, i.e. the Jacobian of a genus-2 hyperelliptic curve, there are no zero coordinates for the level-4 theta coordinate.

$g = 3$ In dimension 3, p.p. abelian varieties can be of the following forms: (i) Jacobians of a genus-3 plane quartic, (ii) Jacobians of a genus-3 hyperelliptic curve, (iii) products of a Jacobian of a genus-2 hyperelliptic curve and an elliptic curve, and (iv) products of three elliptic curves [49, §2.3]. Let N be the number of zero coordinates for a level-4 theta-null point for a p.p. abelian variety of dimension 3. Then $N = 0$ in the case (i), $N = 1$ in the case (ii), $N = 6$ in the case (iii) and $N = 9$ in the case (iv) [49, Proposition 2.15].

A.3 The 2-torsion for $g = 1, 2$

In Subsection 2.3, we provided a description of a canonical symplectic basis for the 2-torsion of a p.p. abelian variety equipped with a level-2 theta structure. The example below illustrate this in the cases $g = 1$ and $g = 2$.

[20] By slight abuse of notation, we denote both the level-2 theta structure and the level-4 theta structure simply by θ^A. Given a theta function, the corresponding theta structure is always clear from the index set.

Example 19. Let A be a p.p. abelian variety of dimension $g = 1$ with theta structure θ^A, and $\theta^A(0_A) = (a_0 : a_1)$. Then the canonical symplectic basis is given by $\mathcal{B} = (P_1, Q_1)$ with $P_1 = (a_1 : a_0)$ and $Q_1 = (a_0 : -a_1)$.
Similarly, for a p.p. abelian variety of dimension $g = 2$ with theta structure θ^A, and $\theta^A(0_A) = (a_{00} : a_{01} : a_{10} : a_{11})$, we obtain the basis $\mathcal{B} = (P_1, P_2, Q_1, Q_2)$ with

$$\begin{aligned} P_1 &= (a_{01} : a_{00} : a_{11} : a_{10}), & P_2 &= (a_{10} : a_{11} : a_{00} : a_{01}), \\ Q_1 &= (a_{00} : -a_{01} : a_{10} : -a_{11}), & Q_2 &= (a_{00} : a_{01} : -a_{10} : -a_{11}). \end{aligned}$$

A.4 2-isogenies and the duplication formula

Here, we provide some intuition behind the 2-isogeny formula stated in Subsection 2.5. Concretely, Remark 20 explains the origins of the formula, and Remark 21 explains the computation of the dual isogeny.

Remark 20. The description of 2-isogenies is deduced from the duplication formula [47, §3]. A special case of this formula can be stated as follows

$$\theta^A(P+Q) \star \theta^A(P-Q) = \mathcal{H}(\widetilde{\theta}^B(\phi(P)) \star \widetilde{\theta}^B(\phi(Q))) \text{ for all} P, Q \in A. \tag{10}$$

Here $\phi : (A, \theta^A) \to (B, \theta^B)$ is the 2-isogeny with kernel K_2, and $\star$ denotes coordinate-wise multiplication. Then the explicit description of ϕ is obtained by evaluating this equation at $Q = 0$, see for example [54, Section 5])
Intuitively, already the first step $\mathcal{S}$ in the isogeny computation defines a 2-isogeny. And it is easy to see that all elements in K_2 get mapped to the same point under this operation. However the codomain of the isogeny defined by $\mathcal{S}$ is not in the correct form, the coordinates can be viewed as *twisted theta coordinates*. The remaining steps $\mathcal{H}, \mathcal{C}_{\widetilde{b}}, \mathcal{H}$ are only linear transformations and consist in *untwisting* the coordinates.

Remark 21. The dual of the isogeny $\phi : (A, \theta^A) \to (B, \theta^B)$ represented as $\mathcal{H} \circ \mathcal{C}_{\widetilde{b}} \circ \mathcal{H} \circ \mathcal{S}$, is given by

$$\hat{\phi} : (B, \theta^B) \to (A, \theta^A), \quad \text{with} \quad \theta^A(\hat{\phi}(P)) = \mathcal{C}_a \circ \mathcal{H} \circ \mathcal{S} \circ \mathcal{H}(\theta^B(P)),$$

where $a = \theta^A(0_A)$ as before.
This description is again a consequence of the duplication formula. For instance, this is obtained by evaluating Equation 10 at $Q = P$, see also [54, Section 5].

B Description of $\mathcal{T}_s$ in the reducible case

In order to complement Corollary 5 and Theorem 8, we will describe the operator $\mathcal{T}_s$ in the special cases where one or more of the x_i vanish. In particular, this includes the cases where we land on reducible varieties.
We remark that the coordinates of $x = (x_{0\ldots0} : \cdots : x_{1\ldots1}) = \mathcal{H} \circ \mathcal{S}((a_{0\ldots0} : \cdots : a_{1\ldots1}))$ are the squares of 2^g of the $2^{g-1}(2^g+1)$ even theta values of level-4 (Lemma 18).[21] The vanishing of the level-4 theta null values is related to the geometry of the abelian variety, see Subsection A.2. We use this description in the case distinction below.

[21] To be more precise, we can evaluate $\left(\theta_{\binom{i_g \ldots i_1}{0 \ldots 0}}(0)\right)^2$ for all $i_g \ldots i_1 \in (\mathbb{Z}/2\mathbb{Z})^g$.

$\boldsymbol{g = 1}$ In dimension one, all even theta values of level-4 are nonzero. In particular $x_0, x_1 \neq 0$ for $(x_0 : x_1) = \mathcal{H} \circ \mathcal{S}((a_0 : a_1))$.

$\boldsymbol{g = 2}$ In dimension two, at most one of the 10 even level-4 theta values vanishes. This happens if and only if the corresponding p.p.a.v. A is reducible, i.e. $A = E_1 \times E_2$ for some elliptic curves E_1, E_2. If this is *not* one of the coordinates corresponding to $(x_{00} : \cdots : x_{11}) = \mathcal{H} \circ \mathcal{S}(Th(A, \theta^A))$, then we can still apply Theorem 8. In the other case, let $\ell_{-1} \in (\mathbb{Z}/2\mathbb{Z})^2$ be the index of the vanishing coordinate, i.e. $x_{\ell_{-1}} = 0$, and set $\{\ell_0, \ell_1, \ell_2\} = (\mathbb{Z}/2\mathbb{Z})^2 \setminus \{\ell_{-1}\}$. In that case, we define

$$\mathcal{T}_s : (x_{00} : \cdots : x_{11}) \mapsto (y_{00} : \cdots : y_{11}),$$

where

$$y_k = \begin{cases} 0 & \text{if } k = \ell_{-1}, \\ x_k & \text{if } \ell = \ell_0, \\ s_i \sqrt{x_{\ell_0} x_{\ell_i}} & \text{if } k = \ell_i, \text{ with } i \in \{1, 2\}. \end{cases}$$

Note that this only requires two square root computations. In particular, the last bit in $s = (s_1, s_2, s_3)$ is redundant.

$\boldsymbol{g = 3}$ In dimension three, there are four different cases.

(i) In the generic case, i.e. when A is the Jacobian of a plane quartic, none of the coordinates vanish, and Theorem 8 can be applied to compute $\mathcal{T}_s$.

(ii) When A is the Jacobian of a hyperelliptic curve, then exactly one of the 36 even theta coordinates vanishes. If this is *not* one of the coordinates corresponding to $(x_{000} : \cdots : x_{111}) = \mathcal{H} \circ \mathcal{S}(Th(A, \theta^A))$, then we can still apply Theorem 8. In the other case, let $\ell_{-1} \in (\mathbb{Z}/2\mathbb{Z})^3$ be the index of the vanishing coordinate and $\{i_0, \ldots, i_6\} = (\mathbb{Z}/2\mathbb{Z})^3 \setminus \{i_{-1}\}$. We define

$$\mathcal{T}_s : (x_{000} : \cdots : x_{111}) \mapsto (y_{000} : \cdots : y_{111}),$$

where

$$y_k = \begin{cases} 0 & \text{if } k = \ell_{-1}, \\ x_k & \text{if } k = \ell_0, \\ s_i \sqrt{x_{\ell_{-1}} x_{\ell_i}} & \text{if } k = \ell_i, \text{ with } i \in \{1, \ldots, 6\}. \end{cases}$$

Note that this case still requires the computation of $6 = g(g+1)/2$ square roots, and none of the bits of $s = (s_1, \ldots, s_6)$ is redundant. To see that our definition of $\mathcal{T}_s$ still results in an isogeny given by $\mathcal{H} \circ \mathcal{T}_s \circ \mathcal{H} \circ \mathcal{S}$, one can use an argument very similar to that of Lemma 3 with the indices of nonzero coordinates permuted appropriately.

(iii) In the remaining two cases, A is reducible and there are either 6 or 9 nonzero even theta coordinates depending on whether A is the product of an elliptic curve and an irreducible abelian surface, or the product of three elliptic curves. This does not necessarily mean that any of the coordinates $(x_{000} : \cdots : x_{111})$ vanishes. In particular, if at most one of the coordinates vanishes, then we may use the same description of $\mathcal{T}_s$ as above.

The vanishing of more than one coordinate results in the redundancy of some of the bits of $s = (s_1, \ldots, s_6)$. An explicit description of $\mathcal{T}_s$ in that case can be obtained in a similar way as in the reducible case in dimension $g = 2$. We remark that this case can be neglected for our application to the cryptographic hash function Theta-CGL (Section 5).

C Proofs for the radical 2^k-isogeny formulas

Here, we provide the proofs for the radical isogeny formulas stated in Section 4.

C.1 Proofs in dimension $g = 1$

Proposition 22 (=Proposition 12). *Let $(a_0 : a_1) \in \mathbb{P}^1$ be the theta null point of an elliptic curve A equipped with θ^A. Then for any $s \in \{{\zeta_4}^i \mid 0 \leq i \leq 3\}$,*

$$\theta^B(0_B) = \left(a_0 - s\sqrt[4]{a_0^4 - a_1^4} : a_0 + s\sqrt[4]{a_0^4 - a_1^4}\right)$$

is the theta null point of a 4-isogeneous elliptic curve B. The 4-isogeny $\phi : A \to B$ is described by $\mathcal{H} \circ \mathcal{T}_{s_2} \circ \mathcal{H} \circ \mathcal{S} \circ \mathcal{H} \circ \mathcal{T}_{s_1} \circ \mathcal{H} \circ \mathcal{S}$ for some $s_1, s_2 \in \{\pm 1\}$ with $s = s_2\sqrt{s_1}$, and $\ker(\phi) = \langle P \rangle$ with $\theta^A(P) = (1 : 0)$. The evaluation of the formulas costs $1\boldsymbol{Quart} + 1\boldsymbol{M} + 2\boldsymbol{S} + 1\boldsymbol{M}_c + 4\boldsymbol{a}$ using Algorithm 5.

Proof. The formula is obtained by applying Algorithm 1 twice with $s = s_2\sqrt{s_1}$. Let $\phi_1 : A \to A_1$ be the isogeny described by $\mathcal{H} \circ \mathcal{T}_{s_1} \circ \mathcal{H} \circ \mathcal{S}$. It follows from the equation (**??**) that $\theta^{A_1}(\phi_1(P)) = (a_0' : -a_1')$, where $(a_0' : a_1')$ is the theta null point of A_1. Therefore, from the equation (**??**), the point $\phi_1(P)$ belongs to the kernel of the isogeny described by $\mathcal{H} \circ \mathcal{T}_{s_2} \circ \mathcal{H} \circ \mathcal{S}$, whence $P \in \ker \phi$. Note that P is of order 4 because $\theta^A(2P)$ is $(a_0 : -a_1)$, which corresponds to a point of order 2. Therefore, the kernel of ϕ is generated by P. □

Proposition 23 (=Proposition 13). *Let $(a_0 : a_1) = Th(A, \theta^A) \in \mathbb{P}^1$ for some elliptic curve A equipped with θ^A. Further let P with $\theta^A(P) = (u_0 : u_1)$ be an 8-torsion point satisfying $\theta^A(2 \cdot P) = (1 : 0)$. Then for any $s \in \{{\zeta_8}^i \mid 0 \leq i \leq 7\}$,*

$$\theta^B(0_B) = (u_0^2 + \lambda^2 : u_0^2 - \lambda^2), \quad \textit{with } \lambda = s \cdot \sqrt[8]{u_0^8 - u_1^8}$$

is the theta null point of an 8-isogeneous elliptic curve B with respect to some theta structure θ^B. The 8-isogeny $\phi : A \to B$ is described by $\mathcal{H} \circ \mathcal{T}_{s_3} \circ \mathcal{H} \circ \mathcal{S} \circ \mathcal{H} \circ \mathcal{T}_{s_2} \circ \mathcal{H} \circ \mathcal{S} \circ \mathcal{H} \circ \mathcal{T}_{s_1} \circ \mathcal{H} \circ \mathcal{S}$ for some $s_1, s_2, s_3 \in \{\pm 1\}$, and $\ker(\phi) = \langle (u_0 : u_1) \rangle$. Moreover, the point Q with

$$\theta^B(Q) = (v_0 : v_1) = \left(a_0 a_1 (u_0^2 - \lambda^2) : a_0^2 u_0 u_1 + \lambda^4 a_1^2/(2u_0 u_1) - \sqrt{2}\lambda a_0 a_1 u_0\right)$$

is an 8-torsion point on B satisfying $\theta^B(2 \cdot Q) = (1 : 0)$.
The evaluation of the formulas costs $1\boldsymbol{Eirt} + 7\boldsymbol{M} + 11\boldsymbol{S} + 4\boldsymbol{M}_c + 8\boldsymbol{a}$ using Algorithm 6

Proof. Let $P = (u_0 : u_1)$ be an 8-torsion point with $\theta^A(2 \cdot P) = (1 : 0)$ as in the statement of the lemma. First of all, we note that this is equivalent to the condition

$$(u_0^4 + u_1^4 : 2u_0^2 u_1^2) = (a_0^2 : a_1^2). \tag{11}$$

The easiest way to see this, is to compute the image of P under the 2-isogeny $\phi_1 : A \to A'$ with kernel $\langle (a_0 : -a_1) \rangle$, and use the fact that $\theta^{A'}(\phi_1(P))$ belongs to $\{(1 : 0), (0 : 1)\}$, since $\theta^{A'}(\phi_1(2 \cdot P)) = (-a_0' : a_1')$ for $(a_0' : a_1') = Th(A', \theta^{A'})$. This provides us with the relation

$$((u_0^2 + u_1^2)^2 : (u_0^2 - u_1^2)^2) = (a_0^2 + a_1^2 : a_0^2 - a_1^2), \tag{12}$$

which is equivalent to the claimed relation $(u_0^4 + u_1^4 : 2u_0^2u_1^2) = (a_0^2 : a_1^2)$.
As the next step, we compute the 8-isogeny with kernel $\langle P\rangle$ as the composition of three 2-isogenies

$$(A, \theta^A) \xrightarrow{\phi_1} (A', \theta^{A'}) \xrightarrow{\phi_2} (A'', \theta^{A''}) \xrightarrow{\phi_3} (B, \theta^B)$$

with each ϕ_i computed as $\mathcal{H} \circ \mathcal{T}_{s_i} \circ \mathcal{H} \circ \mathcal{S}$. The proof is purely computational (using Equation 12 and the definition of λ). Below, we provide the results of the two intermediate steps:

$$Th(A', \theta^{A'}) = (u_0^2 : u_1^2), \quad Th(A'', \theta^{A''}) = (u_0^4 + u_1^4 + \lambda^4 : u_0^4 + u_1^4 - \lambda^4).$$

We remark that the computations involve taking square roots, which are chosen in a canonical way depending on the value of λ and the coordinates of the 8-torsion point P. In particular it is ensured that $\theta^{A'}(\phi_1(P)) = (1 : 0) = \ker(\phi_3 \circ \phi_2)$.
It remains to show that Q is an 8-torsion point of B satisfying $\theta^B(2{\cdot}Q) = (1 : 0)$. This can be checked with by a (tedious) computation using the relations derived above. For the convenience of the reader, we provide a symbolic verification in SageMath in our GitHub repository . □

C.2 Proofs in dimension $g = 2$

Proposition 24 (= Proposition 14). *Let $(a_{00} : a_{01} : a_{10} : a_{11}) \in \mathbb{P}^3$ be the theta null point of a p.p. abelian surface A equipped with θ^A, and denote*

$$x = (x_{00} : x_{01} : x_{10} : x_{11}) = \mathcal{H} \circ \mathcal{S}(\theta^A(0_A)).$$

Then for any $r = (r_1, r_2) \in \{{\zeta_4}^i \mid 0 \leq i \leq 3\}^2$ and $t = (t_1, t_2) \in \{\pm 1\}^2$,

$$\theta^B(0_B) = (b_{00} : b_{01} : b_{10} : b_{11})$$

with

$$\begin{aligned} b_{00} &= \sqrt{2}a_{00} + \alpha_1 + \alpha_2 + \alpha_3, \\ b_{01} &= \sqrt{2}a_{00} - \alpha_1 + \alpha_2 - \alpha_3, \\ b_{10} &= \sqrt{2}a_{00} + \alpha_1 - \alpha_2 - \alpha_3, \\ b_{11} &= \sqrt{2}a_{00} - \alpha_1 - \alpha_2 + \alpha_3, \end{aligned} \quad \textit{and} \quad \begin{aligned} \alpha_0 &= t_1\sqrt{x_{00}x_{01}x_{10}x_{11}}, \\ \alpha_1 &= r_1\sqrt[4]{2\alpha_0 + x_{00}x_{10} + x_{01}x_{11}}, \\ \alpha_2 &= r_2\sqrt[4]{2\alpha_0 + x_{00}x_{01} + x_{10}x_{11}}, \end{aligned}$$

$$\alpha_3 = t_2\sqrt{\frac{(x_{00}x_{10} + \alpha_0)(x_{10}x_{11} + \alpha_0)}{\alpha_1^2\alpha_2^2x_{10}} + \frac{(x_{01}x_{11} + \alpha_0)(x_{10}x_{11} + \alpha_0)}{\alpha_1^2\alpha_2^2x_{11}}}.$$

is the theta null point of a 4-isogenous p.p. abelian surface B. The 4-isogeny $\phi : A \to B$ is described by $\mathcal{H}\circ\mathcal{T}_{s_2}\circ\mathcal{H}\circ\mathcal{S}\circ\mathcal{H}\circ\mathcal{T}_{s_1}\circ\mathcal{H}\circ\mathcal{S}$ for some $s_1, s_2 \in \{\pm 1\}^3$. The evaluation of the formulas costs $2\boldsymbol{Quart} + 2\boldsymbol{Sqrt} + 14\boldsymbol{M} + 4\boldsymbol{S} + 5\boldsymbol{M}_c + 27\boldsymbol{a}$ using Algorithm 7.

Note that $\alpha_3^4 = 2\alpha_0 + x_{00}x_{11} + x_{01}x_{10}$, and the expressions for $\alpha_1, \alpha_2, \alpha_3$ are more symmetric than they appear in the radical description in the proposition.

Proof. It suffices to prove that we can compute $\mathcal{H}\circ\mathcal{T}_{s_2}\circ\mathcal{H}\circ\mathcal{S}\circ\mathcal{H}\circ\mathcal{T}_{s_1}\circ\mathcal{H}\circ\mathcal{S}(0_A)$ for some $s_1, s_2 \in \{\pm 1\}^3$ by the formula in the statement.
Denote by $\phi_1\colon A \to A_1$ the isogeny described by $\mathcal{H}\circ\mathcal{T}_{s_1}\circ\mathcal{H}\circ\mathcal{S}$. Put $s_1 = (s'_1, s'_2, s'_3)$, $s_2 = (s''_1, s''_2, s''_3)$, and $\theta^{A_1}(0_{A_1}) = (b'_{00} : b'_{01} : b'_{10} : b'_{11})$. One can check that

$$\begin{aligned}
b'_{00} &= x_{00} + s'_1\sqrt{x_{00}x_{01}} + s'_2\sqrt{x_{00}x_{10}} + s'_3\sqrt{x_{00}x_{11}}\\
b'_{01} &= x_{00} - s'_1\sqrt{x_{00}x_{01}} + s'_2\sqrt{x_{00}x_{10}} - s'_3\sqrt{x_{00}x_{11}},\\
b'_{10} &= x_{00} + s'_1\sqrt{x_{00}x_{01}} - s'_2\sqrt{x_{00}x_{10}} - s'_3\sqrt{x_{00}x_{11}},\\
b'_{11} &= x_{00} - s'_1\sqrt{x_{00}x_{01}} - s'_2\sqrt{x_{00}x_{10}} + s'_3\sqrt{x_{00}x_{11}}.
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
b'^2_{00} + b'^2_{01} + b'^2_{10} + b'^2_{11} &= 4x_{00}^2 + 4x_{00}x_{01} + 4x_{00}x_{10} + 4x_{00}x_{11},\\
b'^2_{00} - b'^2_{01} + b'^2_{10} - b'^2_{11} &= 8s'_1x_{00}\sqrt{x_{00}x_{01}} + 8s'_2s'_3x_{00}\sqrt{x_{10}x_{11}},\\
b'^2_{00} + b'^2_{01} - b'^2_{10} - b'^2_{11} &= 8s'_2x_{00}\sqrt{x_{00}x_{10}} + 8s'_3s'_1x_{00}\sqrt{x_{11}x_{01}},\\
b'^2_{00} - b'^2_{01} - b'^2_{10} + b'^2_{11} &= 8s'_3x_{00}\sqrt{x_{00}x_{11}} + 8s'_1s'_2x_{00}\sqrt{x_{01}x_{10}}.
\end{aligned}$$

This implies that $\mathcal{S}\circ\mathcal{H}(\theta^B(0_B))$ equals to

$$\begin{aligned}
&(2a_{00}^2 : s'_1\sqrt{x_{00}x_{01}} + s'_2s'_3\sqrt{x_{10}x_{11}}\\
&\qquad : s'_2\sqrt{x_{00}x_{10}} + s'_3s'_1\sqrt{x_{11}x_{01}} : s'_3\sqrt{x_{00}x_{11}} + s'_1s'_2\sqrt{x_{01}x_{10}})\\
&= \Bigg(2a_{00}^2 : s'_1\sqrt{x_{00}x_{01} + x_{10}x_{11} + 2s'_1s'_2s'_3\sqrt{x_{00}x_{01}x_{10}x_{11}}}\\
&\qquad : s'_2\sqrt{x_{00}x_{10} + x_{11}x_{01} + 2s'_1s'_2s'_3\sqrt{x_{00}x_{01}x_{10}x_{11}}} : s'_3\sqrt{x_{00}x_{11}} + s'_1s'_2\sqrt{x_{01}x_{10}}\Bigg)
\end{aligned}$$

because $x_{00} + x_{01} + x_{10} + x_{11} = 4a_{00}^2$. Therefore, by letting $t_1 = s'_1s'_2s'_3$, $r_1 = s''_1\sqrt{s'_1}$, and $r_2 = s''_2\sqrt{s'_2}$, we have

$$\mathcal{H}(\theta^B(0_B)) = \left(\sqrt{2}a_{00} : \alpha_1 : \alpha_2 : s''_3\sqrt{s'_3\sqrt{x_{00}x_{11}} + s'_1s'_2\sqrt{x_{01}x_{10}}}\right).$$

Furthermore, it follows from the direct computation that

$$\begin{aligned}
&(s'_1\sqrt{x_{00}x_{01}} + s'_2s'_3\sqrt{x_{10}x_{11}})(s'_2\sqrt{x_{00}x_{10}} + s'_3s'_1\sqrt{x_{11}x_{01}})(s'_3\sqrt{x_{00}x_{11}} + s'_1s'_2\sqrt{x_{01}x_{10}})\\
&= \alpha_0(x_{00} + x_{01} + x_{10} + x_{11}) + x_{00}x_{01}x_{10}x_{11}\left(\frac{1}{x_{00}} + \frac{1}{x_{01}} + \frac{1}{x_{10}} + \frac{1}{x_{11}}\right)\\
&= \frac{(x_{00}x_{10} + \alpha_0)(x_{10}x_{11} + \alpha_0)}{x_{10}} + \frac{(x_{01}x_{11} + \alpha_0)(x_{10}x_{11} + \alpha_0)}{x_{11}} = \alpha_1^2\alpha_2^2\alpha_3^2.
\end{aligned}$$

Hence, by letting $t_2 = s''_3$, we obtain the formula. □

D Comparison of our radical isogeny formulas to the literature

There exist different descriptions of radical ℓ-isogenies in the literature. In particular in the case $g = 1$, formulas are known for different forms of elliptic

curves. In dimension $g = 2$, formulas by Richelot [53] naturally provide us with radical 2-isogenies between Jacobians of genus-2 curves, and there also exists a description of radical 3-isogeny formulas [8]. In dimension $g = 3$, the description of radical 2-isogenies is more recent. For instance, there exists a description in terms of level-4 theta null points in [49]. Here, we compare the state-of-the-art formulas with out method for computing radical 2-isogenies.
Typically, the cost for computing a radical N-isogeny in dimension g is dominated by the cost of computing $g(g+1)/2$ many N-th roots. Efficient radical isogeny formulas aim at minimizing the remaining field operations that are necessary to evaluate the formula. Throughout, we use the following notation for field operations:

$$\begin{aligned}&\mathbf{Sqrt} = \text{square root}, \quad \mathbf{Crt} = \text{cube root}, \quad \mathbf{Quart} = \text{fourth root},\\ &\mathbf{Eirt} = \text{eighth root}, \quad \mathbf{M} = \text{multiplication}, \quad \mathbf{S} = \text{squaring},\\ &\mathbf{M}_c = \text{scalar multiplication}, \quad \mathbf{a} = \text{addition}, \quad \mathbf{I} = \text{inversion}.\end{aligned}$$

In the tables below, $\mathbf{M}_c$ and $\mathbf{a}$ are omitted to provide an easier overview.

D.1 Radical ℓ-isogenies in dimension 1

$\ell = 2$ Let $S_0(4)$ be the moduli space for Montgomery curves. A radical 2-isogeny formula on Montgomery curves is provided in [9], which is given by

$$\begin{array}{ccc} S_0(4) \times \{1, -1\} & \longrightarrow & S_0(4) \\ (a, s) & \longmapsto & 2(3 + a(-a + s\sqrt{a^2 - 4})) \end{array},$$

where elements in $S_0(4)$ are represented via their Montgomery coefficients. This formula can be computed in the cost $1\mathbf{Sqrt} + 1\mathbf{M} + 1\mathbf{S} + 4\mathbf{a}$. We also know a radical 2-isogeny formula on elliptic curves given in the form $y^2 = x^3 + a_2x^2 + a_4$. From [14], the formula is given by

$$((a_2, a_4), s) \longmapsto (6s\sqrt{a_4} + a_2, 4sa_2\sqrt{a_4} + 8a_4).$$

The computational cost of this formula is $1\mathbf{Sqrt} + 1\mathbf{M} + 3\mathbf{m} + 2\mathbf{a}$ [12, Table 1]. Since the computational cost for radical 2-isogenies via theta coordinates is $1\mathbf{Sqrt} + 1\mathbf{M} + 2\mathbf{S} + 4\mathbf{a}$ (see Algorithm 1), formulas in previous studies for computing 2-radical isogenies on dimension 1 are more efficient than that on theta model.

$\ell = 4$ A radical 4-isogeny formula on Montgomery curves is provided in [50], which is given by

$$\begin{array}{ccc} S_0(4) \times \{\zeta_4^i \mid 0 \leq i \leq 3\} & \longrightarrow & S_0(4) \\ (a, s) & \longmapsto & \dfrac{(s\sqrt[4]{4(a+2)} + 2)^4}{4s\sqrt[4]{4(a+2)}(s^2\sqrt[4]{4(a+2)}^2 + 4)} - 2 \end{array}.$$

This formula can be computed in $1\mathbf{Quart} + 1\mathbf{M} + 2\mathbf{S} + 2\mathbf{M}_c + 6\mathbf{a} + 1\mathbf{I}$. A radical 4-isogeny formula in Tate normal form $y^2 + xy - ay = x^3 - ax^2$ is also provided in [14] as follows:

$$(a, s) \longmapsto s\sqrt[4]{a}\frac{4s^2\sqrt[4]{a}^2 + 1}{(2s\sqrt[4]{a} + 1)^4}.$$

The cost to compute this formula is 1**Quart** + 1**M** + 2**S** + 1$\mathbf{M}_c$ + 3**a** + 1**I**. From Algorithm 5, the cost of a radical 4-isogeny formula on theta model is 1**Quart** + 1**M** + 2**S** + 1$\mathbf{M}_c$ + 4**a**. Therefore, the algorithm using theta model is expected to be the most efficient for computing radical 4-isogenies.

$\boldsymbol{\ell = 8}$ In [14], a radical 8-isogeny is provided in Tate normal form $y^2+(1-c)xy-by = x^3 - bx^2$ with $b = a(a-1)/(a-2)^2$ and $c = -a(a-1)/(a-2)$. Let $S_1(8)$ be the moduli space of curves in Tate normal form with $b = a(a-1)/(a-2)^2$ and $c = -a(a-1)/(a-2)$. Then, the formula is given by

$$\begin{array}{ccc} S_1(8) \times \{\zeta_8^i \mid 0 \le i \le 7\} & \longrightarrow & S_1(8) \\ (a, s) & \longmapsto & \dfrac{-2s^2a(a-2)\alpha^2 - a(a-2)}{s^4(a-2)^2\alpha^4 - s^2a(a-2)\alpha^2 - sa(a-2)\alpha + a}, \end{array}$$

where α is an eighth root of $-a^4(a-1)/(a-2)^4$. This formula can be computed in 1**Eirt** + 5**M** + 6**S** + 1$\mathbf{M}_c$ + 8**a** + 2**I**. From Algorithm 6, the radical 8-isogeny formula in theta model can be computed in 1**Eirt** + 7**M** + 11**S** + 4$\mathbf{M}_c$ + 7**a**. Therefore, the formula in theta model is expected to be more efficient than that in Tate normal form.

The results from the above discussion are summarized in Table 3.

$\mathbf{g = 1}$	Tate [14]	Montgomery [50]	this work
$\ell = 2$		1**Sqrt** + 1**M** + 1**S**	1**Sqrt** + 1**M** + 2**S**
$\ell = 4$	1**Quart** + 1**M** + 2**S** + 1**I**	1**Quart** + 1**M** + 2**S** + 1**I**	1**Quart** + 1**M** + 2**S**
$\ell = 8$	1**Eirt** + 5**M** + 6**S** + 2**I**		1**Eirt** + 8**M** + 9**S**

Table 3. Comparison of different radical isogeny formulas in dimension $g = 1$

D.2 Radical ℓ-isogenies in dimension 2

$\boldsymbol{\ell = 2}$ Formulas for computing 2-isogenies between Jacobians of genus-2 curves go back to the 19th century and were found by Richelot [53]. A modern description of Richelot isogenies viewed as radical 2-isogenies can be found in [13]. In order to be able to count operations, we provide a more explicit description in Algorithm 9. While this is based on [13, Algorithm 1], there are some differences between the two algorithms. These differences in the description do not contribute to any additional computations. The advantage of our description is that it is closer to the setup that we use for computing isogenies in the theta model.

The main difference is that in [13], the input bits (s_0, s_1, s_2) determine an (admissible) quadratic splitting [13, Proposition 3], while in our case, they determine the sign of the square roots $\delta_1, \delta_2, \delta_3$. Implicitly, the latter determines an ordering of the roots and in this way also a quadratic splitting. Furthermore, in

Algorithm 9 Richelot isogenies as radical 2-isogenies ($g = 2$)

```
Input: Quadratic polynomials $g_1 = g_{1,2}x^2 + g_{1,1}x + g_{1,0}$, $g_2 = g_{2,2}x^2 + g_{2,1}x + g_{2,0}$, $g_3 = g_{3,2}x^2 + g_{3,1}x + g_{3,0} \in k[x]$ defining the genus-2 curve $C : y^2 = g_1 g_2 g_3$, and a choice of bits $s = (s_0, s_1, s_2) \in \{-1, 1\}^3$.
Output: Quadratic polynomials $h_1, h_2, h_3 \in k[x]$ defining the genus-2 curve $C : y^2 = h_1 h_2 h_3$.
$\Delta_1, \Delta_2, \Delta_3 \leftarrow g_{11}^2 - 4g_{12}g_{10}, g_{21}^2 - 4g_{22}g_{20}, g_{31}^2 - 4g_{32}g_{30}$    ▷ $3\mathbf{M} + 3\mathbf{S} + 3\mathbf{a}$
$\delta_1, \delta_2, \delta_3 \leftarrow \sqrt{\Delta_1}, \sqrt{\Delta_2}, \sqrt{\Delta_3}$    ▷ $3\mathbf{Sqrt}$
$\delta_1 \leftarrow \mathsf{CondNeg}(\delta_1, s_0)$
$\delta_2 \leftarrow \mathsf{CondNeg}(\delta_2, s_1)$
$\delta_3 \leftarrow \mathsf{CondNeg}(\delta_3, s_2)$
$\alpha_1, \alpha_2, \alpha_3 \leftarrow -g_{1,1} + \delta_1, -g_{2,1} + \delta_2, -g_{3,1} + \delta_3$    ▷ $3\mathbf{a}$
$\alpha_4, \alpha_5, \alpha_6 \leftarrow -g_{1,1} - \delta_1, -g_{2,1} - \delta_2, -g_{3,1} - \delta_3$    ▷ $3\mathbf{a}$
$\beta_1, \beta_2, \beta_3 \leftarrow 2g_{1,2}, 2g_{2,2}, 2g_{3,2}$    ▷ $3\mathbf{a}$
▷ The choice of square-roots determines the quadratic splitting: $g_1 = (\beta_1 x - \alpha_1)(\beta_2 x - \alpha_2)$, $g_2 = (\beta_3 x - \alpha_3)(\beta_1 x - \alpha_4)$, $g_3 = (\beta_2 x - \alpha_5)(\beta_3 x - \alpha_6)$.
$g_{1,2}, g_{2,2}, g_{3,2} \leftarrow \beta_1\beta_2, \beta_1\beta_3, \beta_2\beta_3$    ▷ $3\mathbf{M}$
$g_{1,1}, g_{2,1}, g_{3,1} \leftarrow -\alpha_1 - \alpha_2, -\alpha_3 - \alpha_4, -\alpha_5 - \alpha_6$    ▷ $3\mathbf{a}$
$g_{1,0}, g_{2,0}, g_{3,0} \leftarrow \alpha_1\alpha_2, \alpha_3\alpha_4, \alpha_5\alpha_6$    ▷ $3\mathbf{M}$
▷ $h_i = g'_{i+1}g_{i+2} - g_{i+1}g'_{i+2}$ with indices viewed modulo 3
$h_1 \leftarrow (g_{2,2}g_{3,1} - g_{2,1}g_{3,2})x^2 + 2(g_{2,2}g_{3,0} - g_{2,0}g_{3,2})x + (g_{2,1}g_{3,0} - g_{2,0}g_{3,1})$    ▷ $6\mathbf{M} + 4\mathbf{a}$
$h_2 \leftarrow (g_{3,2}g_{1,1} - g_{3,1}g_{1,2})x^2 + 2(g_{3,2}g_{1,0} - g_{3,0}g_{1,2})x + (g_{3,1}g_{1,0} - g_{3,0}g_{1,1})$    ▷ $6\mathbf{M} + 4\mathbf{a}$
$h_3 \leftarrow (g_{1,2}g_{2,1} - g_{1,1}g_{2,2})x^2 + 2(g_{1,2}g_{2,0} - g_{1,0}g_{2,2})x + (g_{1,1}g_{2,0} - g_{1,0}g_{2,1})$    ▷ $6\mathbf{M} + 4\mathbf{a}$
return $(h_1, h_2, h_3)$    ▷ Total Cost: $3\mathbf{Sqrt} + 27\mathbf{M} + 3\mathbf{S} + 27\mathbf{a}$
```

our approach it is more natural to use quadratic polynomials as inputs, whereas in [13], the input consists of linear polynomials.

Counting the operations in Algorithm 9, we find that computing a radical 2-isogeny with Richelot's formula can be done in $3\mathbf{Sqrt} + 27\mathbf{M} + 3\mathbf{S} + 27\mathbf{a}$. The same computation using our new formulas in terms of level-2 theta coordinates can be performed in only $3\mathbf{Sqrt} + 3\mathbf{M} + 4\mathbf{S} + 16\mathbf{a}$, see Algorithm 2. Another advantage of our description for radical 2-isogenies is that it trivially generalizes to splitting and gluing isogenies as outlined in Subsection 3.3. In contrast to that, Richelot's formulas only describe isogenies between irreducible principally polarized abelian surfaces. The gluing and splitting isogenies in this setting are described by different formulas, see for instance [13, Subsections 6.2,6.3].

$\boldsymbol{\ell = 3}$ For Mumford coordinates, the only other known radical isogeny formulas, are formulas for 3-isogenies [8,10]. A (generous) count of operations in the algorithm provided in [10] yields 3 **Crt** and approximately $1200\mathbf{M} + 800\mathbf{S}$. We note that this algorithm is based on the formulas from [8] which have been considerably improved in [24]. These improvements do not directly aim at the radical isogeny formulas, but rather at the evaluation of the isogeny, therefore we cannot predict the quality of improvements on the former. Furthermore, a recent work on the computation of $(3^n, 3^n)$-isogenies indicates that working with the level-2 theta structure might also be beneficial to find better radical formulas for 3-isogenies [17].

$\ell = 4$ While we do not provide radical 3-isogeny formulas in our model, we did find formulas for computing 4-isogenies which can be evaluated in $2\mathbf{Quart} + 2\mathbf{Sqrt} + 14\mathbf{M} + 4\mathbf{S} + 5\mathbf{M}_c + 27\mathbf{a}$, see Algorithm 7. We are not aware of any similar formulas in the literature, but we remark that they compare very favourably to the 3-isogenies in the Mumford model discussed above.

A comparison of the different available radical isogeny formulas is provided in Table 4.

$\mathbf{g = 2}$	Mumford coordinates [10]	this work
$\ell = 2$	$3\mathbf{Sqrt} + 27\mathbf{M} + 3\mathbf{S}$	$3\mathbf{Sqrt} + 3\mathbf{M} + 4\mathbf{S}$
$\ell = 3$	$\approx 3\mathbf{Crt} + 1200\mathbf{M} + 800\mathbf{S}$	—
$\ell = 4$	—	$2\mathbf{Quart} + 2\mathbf{Sqrt} + 14\mathbf{M} + 4\mathbf{S}$

Table 4. Comparison of different radical isogeny formulas in dimension $g = 2$

D.3 Radical 2-isogenies in dimension 3

In [49, Algorithm 3.3], Ohashi, Onuki, Kudo, Yoshizumi, and Nuida present an algorithm for computing 2-isogenies in dimension 3. In order to compare the number of operations of the two approaches, we consulted the accompanying implementation provided in `https://github.com/Ryo-Ohashi/sspg3list/blob/master/functions.m`. We note that the goal of their implementation is not to provide highly-optimized formulas. In order to make a fair comparison, we give an optimized operation count which is significantly lower than the literal operation count. To make this more transparent for the reader, a brief outline of the algorithm including our operation count is provided in Algorithm 10. The notation is adapted to our setting.

Algorithm 10 ComputeOneRichelotIsogeny from [49] (simplified)

```
Input: A level-4 theta null point $(\theta_0^2 : \dots, \theta_{63}^2)$ of a p.p. abelian threefold $A$,
    with $\theta_0, \dots, \theta_7 \neq 0$.
Output: A level-4 theta null point $(\theta_0'^2 : \dots, \theta_{63}'^2)$ of a 2-isogeneous p.p. abelian threefold $B$.
$x_1 \dots, x_6 \leftarrow \theta_0^2 \cdot (\theta_1^2, \dots, \theta_6^2)$                ▷ $6\mathbf{M}$
$y_0 \leftarrow \theta_0^2$
$y_1, \dots, y_6 \leftarrow \sqrt{x_1}, \dots, \sqrt{x_6}$                ▷ $6\mathbf{Sqrt}$
$y_7 \leftarrow \mathsf{LastSqrt\text{-}level4}(x, y)$                ▷ $23\mathbf{M} + 1\mathbf{S} + 9\mathbf{a}$
$t_{ij} \leftarrow y_i \cdot y_j$ for all $i, j \in \{0, \dots, 7\}$                ▷ $28\mathbf{M} + 2\mathbf{S}$
$\theta_k'^2 \leftarrow \sum_{(i,j) \in I_k} \pm t_{ij}$ for $k = 0, \dots, 63$                ▷ $36 \cdot 8\mathbf{a}$
return $(\theta_0'^2 : \dots : \theta_{63}'^2)$                ▷ Total Cost: $6\mathbf{Sqrt} + 57\mathbf{M} + 3\mathbf{S} + 297\mathbf{a}$
```

Clearly, this description is not self-contained and we encourage the reader to consult the original paper. To understand the operation count, we remark the following:

- Line 5: We use that $t_{ij} = t_{ji}$ for all i, j, moreover $t_{ii} = x_i$ for $i = 1, \dots, 6$.
- Line 6: We have $\#I_k = 8$ for all $k = 0, \dots, 63$. Moreover, it is only necessary to do the computation for the 36 even theta coordinates (out of 64 coordinates in total).

For a comparison with our methods (Algorithm 4), one needs to have in mind that the description above is not constant time. Adding edge cases (in particular in the computation of LastSqrt-level4) will naturally come at the cost of additional operations. Nevertheless, our constant-time implementation already proves to be more efficient, requiring $6\mathbf{Sqrt} + 36\mathbf{M} + 13\mathbf{S} + 7\mathbf{M}_c + 57\mathbf{a}$ for the computation of a 2-isogeny. The main advantage comes from the fact that we only require 8 coordinates to represent a p.p. abelian threefold, while in the level-4 setting it is necessary to work with 36 coordinates.

Last but not least, it is important to mention that the goal in [49] is to obtain the full Richelot isogeny graph of superspecial abelian varieties in dimension 3. In contrast to that, we are interested in *radical* isogenies. That is we require that the composition of two radical 2-isogenies yields a 4-isogeny, cf. Subsection 3.4. In the setting of Theta-CGL, this is important in order to avoid collision attacks through trivial cycles. We leave it as future work to see how Algorithm 10 fits in this framework. We have recently discovered that Ohashi and Onuki [48] are pursuing similar ideas using the formulas in [49] to develop a CGL-style hash function.

E Superspecial isogeny graphs

Here, we recall properties of the superspecial isogeny, and discuss the fast mixing properties in more detail (Remark 26). Moreover, we summarize the state-of-the-art in solving the path finding problem in these graphs.

E.1 Properties of the graph

The vertices of the *superspecial* 2*-isogeny graph* $\mathit{\Gamma}_g(2; p)$ are isomorphism classes of superspecial p.p. abelian varieties of dimension g defined over $\bar{\mathbb{F}}_p$, and the edges represent 2-isogenies. We denote $\mathcal{S}_{g,p} = V(\mathit{\Gamma}_g(2; p))$ for the vertex set. The following theorem gives us the sizes of the vertex set in each of the dimensions that we are interested in.

Theorem 25. *For $p > 7$, the number of superspecial abelian varieties of dimension g is:*

$$\begin{aligned} &\frac{p-1}{12} + O(1) && \textit{for } g = 1, \\ &\frac{p^3 + 24p^2 + 141p}{2880} + O(1) && \textit{for } g = 2, \\ &\frac{p^6 - p^5 + 610p^4 - 2410p^3}{1451520} + O(p^2) && \textit{for } g = 3. \end{aligned}$$

In general, the number of superspecial p.p. abelian varieties of dimension g is $O(p^{g(g+1)/2})$.

Proof. The first result is standard [61, V.4.1(c)]. The second is a result of Ibukiyama, Katsura, and Oort [37, Theorem 3.3]. The third result is from Brock [7]. The general result about dimension g is from Ekedahl [29]. □

Furthermore, it is known that the graph $\Gamma_g(2;p)$ is an expander graph [2]. The output of a random walk on an expander graph with N vertices tends to the uniform distribution after $O(\log(N))$ steps.

Remark 26. With the state of the art it is not possible to determine the (minimal) constant c_g for $g \in \{1,2,3\}$ with the property that a walk in $\Gamma_g(2;p)$ of length $c_g \cdot \log p$ is statistically indistinguishable from random (with respect to a security parameter λ). This would be required to give a precise statement about the minimal length of a message m so that $\mathsf{Theta\text{-}CGL}_{g,\ell}(m)$ is statistically indistinguishable from random.
In order to compute the minimal message length with this property, one needs to study the *spectrum* of the graph, in particular, we need to know the second largest eigenvalue, α, of the 2-isogeny graph as this value affects how quickly a random walk on the graph converges to a random distribution. Little is known about this eigenvalue for the 2-isogeny graph where $g = 2$ and 3. However, Aikawa, Tanaka, and Yamauchi proved an upper bound for normalized[22] α [2, Thm. 1.1],

$$1 - \alpha \geq \frac{1}{4(g+2)} \left(\frac{\ell - 1}{2(\ell-1) + 3\sqrt{2\ell(\ell+1)}} \right)^2$$

for the ℓ-isogeny graph. In our case, we have the upper bounds of

$$\alpha \approx \begin{cases} 1 - 2^{-10.8} & \text{for } g = 1, \\ 1 - 2^{-11.3} & \text{for } g = 2, \\ 1 - 2^{-11.6} & \text{for } g = 3. \end{cases}$$

To compute the number of steps required for the final vertex to be arbitrarily close to the uniform distribution, we turn to a result in [33, Lecture 10,Thm. 5] which implies that we would require a random walk of at least 2^{18} steps. This means that the minimum hash length is $2^{18}, 2^{21}, 2^{24}$ for $g = 1,2,3$ respectively. Taking the other extreme where we make the calculations based on Ramanujan graphs (which is not the case when $g = 2$ and 3), we can take the upper bound on α to be $\alpha < \frac{2\sqrt{d-1}}{d}$, and we have that

$$\alpha \approx \begin{cases} 2^{-0.085} & \text{for } g = 1, \\ 2^{-1} & \text{for } g = 2, \\ 2^{-2.5} & \text{for } g = 3, \end{cases}$$

and using Theorem 3.2 of [36], we find that we need $3011, 320, 128$ steps in the 2-isogeny graph for $g = 1,2,3$ respectively. This implies a minimum message length of $3011, 960, 768$-bits, respectively.

[22] The result in the reference is the second largest eigenvalue of the normalized Laplacian, so we need to look at $\alpha = 1 - \lambda_2$

E.2 The path finding problem

In this part we discuss the state-of-the-art methods to solve Problem 1. That is given two superspecial p.p. abelian varieties A_1, A_2 over $\mathbb{F}_{p^2}$, find a 2^k-isogeny $\phi : A_1 \to A_2$.
The best known generic attack for solving Problem 1 is a Pollard-rho style attack which performs pseudorandom walks in the isogeny graph $\mathit{\Gamma}_g(2;p)$. This attack has complexity the square root of the size of the graph. Hence, using Theorem 25, Pollard-rho has complexity $\widetilde{O}(p^{1/2})$, $\widetilde{O}(p^{3/2})$, $\widetilde{O}(p^3)$, for $g = 1, 2, 3$ respectively.
Similarly, the best generic quantum attack is a Grover search [34,45]. This has complexity $\widetilde{O}(p^{1/4})$, $\widetilde{O}(p^{3/4})$, $\widetilde{O}(p^{3/2})$, for $g = 1, 2, 3$ respectively.
Relaxing the conditions of Problem 1 and also allowing isogenies of different degrees in the path, one might also consider the more memory friendly variant by Delfs and Galbraith [25] in the case $g = 1$. The Delfs–Galbraith attack aims to find paths to supersingular elliptic curves defined over $\mathbb{F}_p$ and solves the isogeny problem in that ground field. This algorithm has a complexity of $\widetilde{O}(\sqrt{p})$, so is similar to that of Pollard-rho. For more details on these generic attacks, the reader is referred to [19], in particular [19, Table 1]
For $g > 1$, the best known method to find isogenies between p.p. abelian varieties is the Costello–Smith attack which repeatedly reduces the problem to an easier instance. More concretely, the Costello–Smith attack performs a random walk on $\mathit{\Gamma}_g(2;p)$ until a reducible variety $A \times A'$ with $\dim(A) = g - d \geq d = \dim(A')$ is encountered, one can then reduce the dimension of the problem to finding isogenies in $\mathit{\Gamma}_{g-d}(2;p)$. The cost of the Costello–Smith attack is $\widetilde{O}(p^{g-1})$, which translates to $\widetilde{O}(p)$, $\widetilde{O}(p^2)$, for $g = 2, 3$ respectively, hence it outperforms Pollard-rho.

Theorem 27 (Costello–Smith attack [19]). *Let $A, A' \in \mathcal{S}_g(2;p)$ and $g > 1$, then there exist algorithms $\mathcal{A}_c$ and $\mathcal{A}_q$ which compute an isogeny $\phi : A \to A'$ as the composition of 2-isogenies with success probability $> 1/2^{g-1}$.*

1. *The algorithm $\mathcal{A}_c$ runs in time $\widetilde{O}(p^{g-1})$ on a classical computer.*
2. *The algorithm $\mathcal{A}_q$ runs in time $\widetilde{O}(\sqrt{p^{g-1}})$ on a quantum computer.*

Notice that since this theorem relies on walking through reducible varieties. This means that it will most likely not be a 2^n-isogeny, hence does not lead to a solution of Problem 1. Still, the attack is able to provide a useful bound for the hardness of the problem, since one might be able to transform the isogeny recovered to a valid path in $\mathit{\Gamma}_g(2;p)$ via analogues of the KLPT algorithm [41]. Hence, we will use the bounds generated by these attacks to estimate the security of our hash function.

F Using modular invariants for the CGL hash function in higher dimension

As explained in section 5.3, because the level-2 theta null point encodes level structure information, it fully determines the first isogeny and part of the second isogeny. It is important for our hash function, in order to avoid trivial collisions, to output the theta null point of the codomain rather than just modular invariants.

In this section, we describe in more details the level structure encoded by a theta null point, how our choice of square roots for a radical 2-isogeny affect this level structure, and how we could adapt the hash function to output a modular invariant (like the Igusa invariants in dimension 2).
There is little reasons to use modular invariants rather than theta constants for the hash function (except possibly in high dimension, because theta constants are less and less compact with respect to the dimension of the moduli space), so we mainly state our results for the sake of completeness.

F.1 The level subgroup induced by a level 2 theta structure

Analytically, theta null points of level 2 are weight $1/2$ modular forms for Igusa's level subgroup $\Gamma(2,4)$. This level subgroup correspond algebraically to the following data:

Proposition 28. *A theta null point of level* 2 *on a p.p. abelian variety* (A, Θ_A) *corresponds to a symplectic basis* $(e_1, \dots, e_g, f_1, \dots, f_g)$ *of the* 4*-torsion for the Weil pairing* $e_{W,4\Theta_A}$*, modulo the equivalence relation:*

$$(e_1, \dots, e_g, f_1, \dots, f_g) \sim (e'_1, \dots, e'_g, f'_1, \dots, f'_g)$$

whenever $e'_i = e_i + t_i$*, with* t_i *a point of* 2*-torsion such that* $e_{W,2\Theta_A}(2e_i, t_i) = 1$*, and similarly for* $f'_i = f_i + t'_i$*:* $e_{W,2\Theta_A}(2f_i, t'_i) = 1$*.*
In particular, the 2*-torsion basis* $(2e_1, \dots, 2e_g, 2f_1, \dots, 2f_g)$ *is independent of the equivalence class, so is fully determined by the level* 2 *symmetric theta structure.*

Proof. This is [56, Corollary 2.11.2]. □

Corollary 29. *A p.p. abelian variety* (A, Θ_A) *over* $\mathbb{F}_q$ *has a rational level* 2 *theta null point if and only if there exists a symplectic basis* $(e_1, \dots, e_g, f_1, \dots, f_g)$ *of the* 2*-torsion such that the self Tate pairings are trivial:* $e_{T,2\Theta_A}(e_i, e_i) = 1$ *and* $e_{T,2\Theta_A}(f_i, f_i) = 1$*.*

Proof. This is [56, Corollary 2.11.4]. □

Example 30.

- An elliptic curve $E/\mathbb{F}_q$ has a rational level 2 theta null point if and only if there is a symplectic decomposition $E[4] = E_1[4] \oplus E_2[4]$ of the 4-torsion with $E_i[4]$ rational.
- An abelian variety $A/\mathbb{F}_{p^2}$ is called maximal or minimal if its number of point is $(p+1)^{2g}$ (resp. $(p-1)^{2g}$). This is equivalent to the Frobenius π_{p^2} being $[-p]$ (resp. $[p]$), and all the endomorphisms of A are defined over $\mathbb{F}_{p^2}$: $End(A) = End_{\mathbb{F}_{p^2}}(A)$. By Tate's theorem, A is then isogeneous to E^g, $E/\mathbb{F}_{p^2}$ a maximal (resp. minimal) elliptic curve, so is supersingular. By [39], A is isomorphic over $\mathbb{F}_{p^2}$ to E^g if $g > 1$ (without the polarisation), so A is even superspecial. Since $A(\mathbb{F}_{p^2}) \simeq (\mathbb{Z}/(p+1)\mathbb{Z})^{2g}$ (resp. $A(\mathbb{F}_{p^2}) \simeq (\mathbb{Z}/(p-1)\mathbb{Z})^{2g}$), the 2-torsion on A is rational. Since the Frobenius is a scalar, all kernels are rational, so in particular any symplectic decomposition of the 4-torsion is rational. It follows that A has a rational theta null point of level 2, and also of level 4 if $4 \mid p+1$ (resp. $4 \mid p-1$). This answers a question originally asked in [49].

F.2 The choice of signs for elliptic curves

We first look at the case $g = 1$. In that case, by example 30, a level 2 theta null point is equivalent to the data of two disjoint cyclic kernels of order 4, K_1 and K_2 on E_0: $E_0[4] = K_1 \oplus K_2$. In particular, it completely determines the first two 2-isogenies: $E_1 = E_0/K_1[2]$ and $E_2 = E_0/K_1$. If $\phi_0 : E_0 \to E_1$ is the first isogeny, the choice of square root we make to determine a theta null point of level 2 on E_1 corresponds to a choice of symplectic decomposition $E_1[4] = K_1' \oplus \phi_1(K_2)$ such that K_1' contains $\phi_1(K_1)$ (so we go from a $\mathbb{Z}/2 \times \mathbb{Z}/4$ level structure to a $\mathbb{Z}/4 \times \mathbb{Z}/4$ level structure, hence why we only need a square root). In particular, the choice of K_1' does not change E_2, but will affect the third isogeny $\phi_2 : E_2 \to E_3$. And likewise, the choice of square root for E_2 will determine the isogeny $\phi_3 : E_3 \to E_4$.

We see why directly outputing the j-invariant would lead to trivial collisions: if the message is of length 2, then all our choice of signs in the theta constants still give the same modular invariant $j(E_2)$. However, using the theta constant on E_2, since it already encodes the upcoming next two isogenies $E_2 \to E_3 \to E_4$, solves this problem. To use the j-invariant, the solution is clear: we start from the theta null point on E_2 and make dummy sign choices (e.g., those corresponding to bits 0) for the theta null point of E_3 and E_4. Then we output $j(E_4)$: although the theta null point on E_4 depends on the dummy choices, the discussion above shows that the j-invariant does not.

An alternative strategy, to only leave the first isogeny $E_0 \to E_1$ fixed, but allow the two possible 2-radical isogenies from E_1, is to consume the first bit differently than the others: if it is 0 we do nothing, and if it is 1 we change the theta null point $(a : b)$ into $(b : a)$. With the notations above, this has the effect of changing K_1 without affecting $K_1[2]$.

F.3 The choice of signs for abelian varieties

We can now treat the general case $g > 1$, which is a bit more delicate, because although the first isogeny $A_0 \to A_1$ is completely determined by the theta null point of A, the second isogeny $A_1 \to A_2$ is only partially determined when $g > 1$.

Namely, start with a representative of the symplectic basis $(e_1, \dots, e_g, f_1, \dots, f_g)$ of $A_0[4]$ determining the theta null point of level 2 on A_0; this symplectic basis is only determined up to the equivalence relation $\sim$ of proposition 28. This equivalence relation fully determines then 2-torsion basis $(2e_1, \dots, 2e_g, 2f_1, \dots, 2f_g)$, and in particular the kernel K_1 generated by $(2e_1, \dots, 2e_g)$ of the first isogeny $\phi_0 : A_0 \to A_1$.

Then part of our level structure descends on A_1, namely our symplectic basis modulo $\sim$ gives $\phi_0(e_i), \phi_0(f_i)$ modulo $\sim$. Since the kernel of ϕ is in the subgroup generated by the (e_i), we do not lose information on the f_i. However, our $\phi_0(e_i)$ are now only points of 2-torsion (only defined up to $\sim$) rather than points of 4-torsion. Our $g(g + 1)/2$ choice of signs in the multiradical 2-isogeny formula amount to two steps: first we lift the $\phi_0(e_i)$ modulo $\sim$ into fully defined points $(e_1', \dots, e_g')$ of 2-torsion. This amount to expanding our partial information on the kernel of the next isogeny to a full 2-isogeny $A_1 \to A_2$. Then, we need to lift our points of 2-torsion $(e_1', \dots, e_g')$ into points of 4-torsion $(e_1'', \dots, e_g'')$, but only up to the equivalence $\sim$.

From the analytic point of view, the choices of these e_i'' above each e_i' correspond to an action of $\Gamma^{(1)}(2)/\Gamma^{(1)}(2,4)$ which is of size 2^g. More precisely, from the analytic point of view a system of representative of $\Gamma(2)/\Gamma(2,4)$ is given by matrices of the form $M = \begin{pmatrix} 1 & diag(b) \\ diag(c) & 1 \end{pmatrix}$, which acts on the theta null point as $M \cdot \theta_i(0) = e_2(b,i)\theta_{i+c}(0)$. For our choices of signs we are only interested by the upper part of M, which change the signs of $\theta_i(0)$. From the algebraic point of view, each e_i'' modulo $\sim$ corresponds to the choice of a symmetric element in the theta group $G(2\Theta_{A_1})$ above e_i'. Since there are two possible symmetric elements for each e_i', we also see that we have 2^g-choices. In total, going from e_i' to e_i'' modulo $\sim$ consumes g square roots. And changing the choices of e_i'' amount to, at the square root step of fig. 2, changing the theta null point of A_1 by $\theta_i(0) \mapsto e_2(b,i)\theta_i(0)$ for an appropriate b.

A similar reasoning allows us to count the number of choices from going to $e_i' = \phi_0(e_i)$ modulo $\sim$ to e_i'. For e_1', looking at the definition of $\sim$, we see that we have 2^{g-1} choices for e_1' (because our choice has to be compatible with the $\phi_0(f_i)$). But then we only have 2^{g-2}-choices for e_2', because it has to be isotropic with e_1', and so on, for a total of $2^{g-1+g-2+\cdots 1}$-choices, or $g(g-1)/2$ possible square roots.

So out of our $g(g+1)/2$ total choice of sign, $g(g-1)/2$ serve as completing the partial kernel information we have on A_1 to a full kernel of a 2-isogeny $\phi_1 : A_1 \to A_2$, and g of them serve as giving partial 4-torsion information on A_1, which gives partial kernel information on the next kernel $A_2 \to A_3$ (which we interpret as the partial kernel information on A_1 of the 4-isogeny $A_1 \to A_3$). An important remark is that, once we have fixed the (e_i') (hence A_2), then a different choice of partial information on the 4-torsion (e_i'') for A_1 give mutually exclusive partial information on the next kernel $A_2 \to A_3$, in the sense that two different choices of (e_i'') modulo $\sim$ can not be completed to the same kernel $A_2 \to A_3$.

As an example, a different choice modulo $\sim$ for e_1'' would be to take $e_1'' + t_1'$, with t_1' a two torsion point on A_1 which involves $\phi_0(2f_1)$ as a component: $t_1' = \phi_0(2f_1) + t_1$ with t_1 having no $\phi_0(2f_1)$ component. Let U_1 be the kernel of the 2-isogeny generated by the (e_i'), and $V_1 \supset U_1$ be the kernel of the 4-isogeny generated by the (e_i''). Then U_1 is fixed, and in the first case V_1 is an isotropic kernel that has to contain e_1'' modulo $\sim$, that is $e_1'' + t_1$ for any point of two torsion t_1 having no $\phi_0(2f_1)$ component. In the second case, V_1 is an isotropic kernel that has to contain a point of the form $e_1'' + \phi_0(2f_1) + t_1$. Since $\phi_0(2f_1)$ has a non trivial pairing with $e_1' = 2e_1''$, we see that our choices of V_1 in the first case are disjoint from our choices of V_1 in the second case.

Now that we have a clear picture of how our choices of signs affect the level structure, we can explain how to change our algorithm in higher dimension in order to use modular invariants. We still assume that our message length is a multiple of $g(g+1)/2$ bits. First, as we have seen, our theta null point fully determine the first isogeny $\phi_0 : A_0 \to A_1$, and part of the second isogeny $\phi_1 : A_1 \to A_2$. Fixing ϕ_0 is good for uniformity of the algorithm: we want to iterate radical isogenies, but we need one isogeny to bootstrap (if really we wanted to start with a uniform 2-isogeny from A_0, we could use the first bits of the message to act by an appropriate matrix in $\Gamma(1)/\Gamma(2,4)$). However, we want to get rid of the extra information fixing part of the next isogeny ϕ_1. For that, we consume the first g bits of the message, this gives us a vector $c \in (\mathbb{Z}/2\mathbb{Z})^g$, and we act by $\theta_i(0) \mapsto \theta_{i+c}(0)$. This action is the same as applying the Hadamard transform, acting by $\theta_i(0) \mapsto e_2(b,i)\theta_i(0)$ for an appropriate b and then acting by the Hadamard transform again. So in view of fig. 2 and the discussion above, this give a uniform way to resample the partial information fixed by the theta null point for ϕ_1.

Then we digest the message by blocks of $g(g+1)/2$ bits as in section 3, except at the last block. The last block m consists of only $g(g-1)/2$ bits. We pad this block with bits consisting of 0 at the emplacements corresponding to the choice of signs for the square root step of fig. 2 at the index of the theta null point corresponding to the canonical basis of $(\mathbb{Z}/2\mathbb{Z})^g$. Since changing the theta null point of A_m by the action $\theta_i(0) \mapsto e_2(b,i)\theta_i(0)$ only affect the second to next isogeny $A_{m+1} \to A_{m+2}$ but not the next isogeny $A_m \to A_{m+1}$, this gives us a way to sample uniformly the isogeny $\phi_m : A_m \to A_{m+1}$ while consuming only $g(g-1)/2$ bits to complete the g bits of information we already had from the theta null point of A_{m-1}. From the theta null point of A_m, we now compute a theta null point for A_{m+1}, using arbitrary choice of square roots (for instance, using a block of length $g(g+1)/2$ of only zeroes), and we output the modular invariants $J(A_{m+1})$ of A_{m+1}. This way, we have computed uniformly across all possible messages all m steps of radical 2-isogenies starting from the fixed one $A_0 \to A_1$.

G Implementation details

Here, we provide more details on the design choices of our implementation. In particular, this section explains the details behind Table 1 given in Section 6.

G.1 Arithmetic in the finite fields

For all cases, roots in $\mathbb{F}_p$ are computed using exponentiations. In dimension one, inversions and Legendre symbols are most efficiently computed using the binary GCD algorithm [52] which runs in constant time and avoids expensive exponentiation. For dimensions two and three where the prime is considerably smaller, we find exponentiations outperform the binary GCD method for inversions and Legendre symbols. Practically, for dimension one (and less-so for dimension two) squaring is cheaper than multiplication, and multiplication by constants smaller than 2^{31} cheaper still, and so when breaking down the arithmetic cost of our algorithms, we count separately the cost of multiplication $\mathbf{M}$, squaring $\mathbf{S}$ and the multiplication by a fixed constant $\mathbf{M}_c$.

Most importantly, all finite field arithmetic has been written to be constant time to avoid side-channels in our hash-function. This engineering restriction is carried through to the hash-function itself with no secret dependent branching or table look ups. Bits of the message which are hashed are consumed in constant time by using conditional negation and conditional swaps.

For all cases, we use a non-canonical internal representation of elements of $\mathbb{F}_p$ by only partially reducing to ensure elements are smaller than $2^{64 \cdot n}$ for values represented with n 64-bit words. This lazy reduction saves a few cycles for each multiplication. Our finite field arithmetic is written using Rust without any assembly optimised instructions, but we do make use of architecture specific intrinsics, such as `mulx` for Intel machines for faster multiplication and carry of two `u64` words.

G.2 Computing 2^k-roots in constant time

An overwhelming contribution to the cost of $\mathsf{Theta\text{-}CGL}_{g,\ell}$ is the cost of the 2^k-roots in $\mathbb{F}_{p^2}$ when computing the 2^k-radical isogenies, and so it is most efficient to spend time finding the cheapest way to compute these in constant time. As our primes are always in the form $p \equiv 3 \mod 4$, we can compute fast square-roots in $\mathbb{F}_{p^2}$ with modulus x^2+1 using the "Complex method" [26,1] at a cost of two square roots, one inversion and one Legendre symbol in $\mathbb{F}_p$.

The tricks from the complex method extend to 2^k-roots in $\mathbb{F}_{p^2}$ in a fairly straight-forward way, see for example [35]. The idea is that for an element $x = x_0 + ix_1 \in \mathbb{F}_{p^2}$, when a root exists, one can always compute the norm $n(x) = x_0^2 + x_1^2$ and compute the 2^k-root of this element in $\mathbb{F}_p$ to recover information about the norm of the root $y = y_0 + iy_1$, $y^{2^k} \equiv x \mod p$, through the fact that the norm is multiplicative: $n(y)^{2^k} = n(x)$.

In previous work related to CGL hash functions [35], they offer a method for computing the 2^k-root of an element in $\mathbb{F}_{p^2}/\mathbb{F}_p$ using one 2^k-root in $\mathbb{F}_p$, one Legendre symbol in $\mathbb{F}_p$, k square-roots and k inversions in $\mathbb{F}_p$. As the roots, inversions and Legendre symbols in $\mathbb{F}_p$ are simply exponentiations, the cost can be approximated as $2(k+1)$ exponentiations in $\mathbb{F}_p$ to compute a 2^k-root in $\mathbb{F}_{p^2}$.

In order for our roots to be constant time, we wish to generalise their method to allow the algorithm to compute roots of $x \in \mathbb{F}_p$ as well (in other words, we allow $x_1 = 0$ for $x = x_0 + ix_1$), which one can do without additional significant cost by simply checking if certain values are zero and using constant time conditional swaps to ensure the correct output. Additionally, one can remove $k-1$ inversions by only computing a single division at the end of algorithm, bringing the total approximated cost down to $k+3$ exponentiations. Details of our algorithm are in both the SageMath and Rust code, for further reference.

We can also turn our attention to the exponentiations themselves to save multiplications. For the following discussion we'll use the 64-bit prime as an example but the same reasoning applies to all three characteristics chosen. For 2^k-roots in $\mathbb{F}_p$ we want to compute $a^{(p+1)/2^{k+1}} \mod p$. For a generic exponentiation using the square and multiply technique, one would estimate $\log(p)$ squarings and $\frac{1}{2}\log(p)$ multiplications. However, the number of multiplications can be reduced by carefully constructing a multiplication chain. As an example, consider computing a square-root with exponent $(p+1)/4 = 2^{62} - 2^6 = 2^6 \cdot (2^{56} - 1)$. The idea is that given some value $x_i = x^{2^i - 1}$ one can compute $x_{i+j} = x_i \cdot x_j^{2^i}$. We can then compute a square-root with only 7 multiplications and 61 squares in the following way:

$$\begin{aligned} x_2 &= x \cdot x^2, & x_4 &= x_2 \cdot x_2^{2^2}, & x_6 &= x_2 \cdot x_4^{2^2}, \\ x_7 &= x \cdot x_6^2, & x_{14} &= x_7 \cdot x_7^{2^7}, & x_{28} &= x_{14} \cdot x_{14}^{2^{14}} \\ x_{56} &= x_{28} \cdot x_{28}^{2^{28}}, & y &= x^{2^6 \cdot (2^{56}-1)} = x_{56}^{2^6}. \end{aligned}$$

The method for inversions and Legendre symbols follows the exact same reasoning, only with exponents equal to $(p-2)$ and $(p-1)/2$ respectively. In practice we use a single method to first compute $(p-3)/4$ using a similar multiplication chain as above and then inversion requires an additional two squares and one multiplication while the Legendre symbol requires one additional square and one additional multiplication.

For a full detail of the costs, see Table 1 for the prime choices and the arithmetic costs of these exponentiations, and see the Rust implementation for a commented discussion of the multiplication chains used for each case.