A note on optimising 2^n-isogenies in higher dimension

DAMIEN ROBERT

Abstract. We give various optimisations for the computations of 2^n-isogenies in higher dimension. In particular, we explain how to compute 2^n-isogenies by pushing forward g points (a basis of the kernel) rather than 2^g points at each step. We detail the case of $g = 1$ and $g = 2$.

Contents

Date: November 13 2023.

1. Context

With the explosion of higher dimensional isogeny cryptography, a group of isogenies enthusiasts have gathered around a Zulip chat (this includes Pierrick Dartois, Sabrina Kunzweiler, Luciano Maino, Giacomo Pope, ...).

The goal was to first start with dimension 2 2^n-isogenies, with a focus on improving Festa and the SIDH attacks, and to pave the way for the dimension 4 implementation of the verification in SQISignHD.

The git repository `https://github.com/GiacomoPope/Theta-Isogenies` contains code for 2^n-isogenies in dimension 2 via Richelot isogenies (+ splitting and gluing) in Mumford coordinates, via Kummer coordinates (with formula due to Sabrina Kunzweiler), and via theta coordinates (in level 2, hence also on the Kummer).

The goal was to optimize these three different models and compare them to each other. These notes were written in June 2023 to describe both 2^n-isogenies algorithms in theta coordinates and various optimisations (in any dimension) I had found compared to the algorithm described in [DLRW23, Appendix C.2].

Strangely, theta functions have somewhat a reputation of being hard to work with and slow (maybe because they can work in any dimension and any degree). Contrary to these expectations, isogeny formula are actually pretty fast in theta coordinates, and most notably for 2^n-isogenies in level 2: level 2 theta functions are precisely tailored so that the action by translation of the 2-torsion (more precisely the theta group) gives extremely fast isogeny images (see also the simplicity of the duplication formula Section 5). Notably a 2-isogeny image in dimension 1 is even faster in theta coordinates than in Montgomery coordinates (see Section 15).

Of course, operation count never beat actual profiling, which was the goal of our Sage implementation (further comparison between the different dimension 2 models are out of scope of these notes, but theta functions are indeed very fast! The current implementation gives a factor $9\times$ for the computation of a 2^{602}-isogeny chain, and images $17\times$ faster, compared to Richelot isogenies.)

Although image computation is naturally very fast in theta coordinates, the codomain computation was originally a lot more involved. The original algorithm of [DLRW23, Appendix C.2]; involved a normalisation process involving 2^g points of 8-torsion.

The original goal of these notes involved a faster normalisation process involving only $1+g$ points of 8-torsion, with some further optimisations like inlining what was needed for the tripling formula used in the normalisation process. This is the version which was first implemented in the git repository, by the people mentioned above.

Since then, I have found (in July 25) newer formula that completely bypass the normalisation process, see Section 17. These formula are both much simpler to implement[1] and a lot faster, they essentially boil down to g images computations.

These makes most of these notes obsolete, notably Sections 7, 10, 11, 14 and 16. These notes are still in their state of June 2023, except for this section and the newer Section 17, written in August 2023.

The obsolete normalisation process described in these notes for 2-isogenies might still have an interest to better explain the similar normalisation process using for higher degree ℓ-isogenies. Indeed, for $\ell > 2$ (and with theta functions of level 2) a normalisation process is needed both for codomain computations but also for images computations.

[1] I recommend looking at the git history to compare the old formula with the newer ones

One might wonder why this normalisation process is no longer needed for $\ell = 2$ but still needed for $\ell > 2$. The answer is that with theta coordinates in level 2, the points of 2 torsion are already normalised with respect to each others, hence the normalisation process was redundant. To have a similar process for $\ell = 3$ (say), we would need to work with theta functions of level 3 or 6. The normalisation process of the points of 3-torsion is essentially a way to work in level 2 almost as if we were in level 6.

Apart from the results of Section 17, we give several formula in the dimension 1 case that might be of interest in Section 15.

A word of warning: these are notes, not a research paper, and there are probably still a lot of remaining typos in the formula. When in doubt look at the code itself, it should be correct!

Update November 2023: we now have a paper [DMPR23a] detailing the formulas for a dimension two 2^n-isogeny in the theta model. The code is also available [DMPR23b]. Pierrick Dartois is working on a follow up paper for the adapation to dimension 4. We strongly recommend reading this article rather than these notes, which as mentioned organically grew as we went along and implemented the algorithm, so are not very readable!

2. Introduction

Computing isogenies in higher dimension has received considerable interest recently: breaking SIDH, SQISignHD, Festa [CD23; MMPPW23; Rob23a; DLRW23; BMP23]. Although algorithms in any dimensions are described in [LR12; CR15; LR15b; LR22a] in a theta model of even level n, for simplicity only the case of an ℓ-isogeny with ℓ prime to n is considered in these articles. For cryptographic applications, the most interesting case is when $\ell = 2^u$ and $n = 2$, which does not satisfy these conditions. The general case of ℓ non prime to n case is briefly treated in [Rob10, Proposition 6.3.5; Rob21, Remarks 2.10.3, 2.10.7 and 2.10.14]. A particular difficulty when ℓ is even is that we need a symplectic basis of the ℓn-torsion which is compatible with the symmetric level n theta structure, a condition for compatibility, due to David Lubicz, is described in [Rob21, Remark 2.10.7]. In an upcoming article with David Lubicz, we will treat this general case in more detail, along with algorithms to raise and descend the level (which are strongly linked to isogeny algorithms).

The purpose of these notes by contrary is to look only at speeding up the formula for the computation of the specific case of 2^n-isogenies in level 2. As usual, this rely on splitting the isogeny $\phi : A \to B$ into a product of n 2-isogenies ϕ_i, and push forward points by the ϕ_i, so we reduce to 2-isogenies. Building on [Rob10, § 6, § 7; Rob21, § 2, § 4], an algorithm to do so was presented in [DLRW23, Appendix C.2]; we will reuse the general notations of this article. For our cryptographic application, our isogeny $\phi : A \to B$ has for domain $A = \prod E_i$ a product of elliptic curves. This also simplifies various steps of the algorithm, notably the initialisation of the algorithm. Also, the compatibility conditions alluded to above is easy to verify in dimension 1 (see Lemma 8.3), and can be propagated through the product theta structure. This allows to essentially bypass it entirely in what follows.

Given $K = \langle T_1, \ldots, T_g \rangle$ an isotropic kernel of A, the standard method to split the isogeny into 2-isogenies is to first compute a basis of $K[2]$ via doubling formula, compute the isogeny $\phi_1 : A \to A_1 = A/K[2]$, push the points T_i via ϕ_1, compute a basis of $f(K)[2]$ via a combination of doubling and pushing points via ϕ_1 (the optimal strategy depends on the relative cost of doubling and pushing points, given these costs an algorithm is described in [DJP14, § 4.2.2]).

We assume that we are given a theta null point of level 2 on A and that K is compatible with this theta null point (see Section 9). Given the theta null point of the isogeneous abelian variety $A_1 = A/K[2]$, the theta model has particularly nice formula to compute the image by a point (see Section 5); this cost $2^g S + (2^g - 1)M$ assuming the theta constants of A_1 are normalised so that $\theta_0^{A_1}(0) = 1$ and the inverse $1/\theta_i^{A_1}(0)$ have been computed. Also it is possible to recover the squares $\theta_i^{A_1}(0)^2$ in only $2^g S$. Given the simplicity of these formula, doubling and differential addition on A are computed by going through the 2-isogeny to A_1 (see Section 6). In particular, doubling essentially cost 2 isogeny evaluations. Furthermore, for the arithmetic on A, the squares ${\theta_i^{A_1}}^2(0)$ are enough. However, for computing a 2^n-isogeny as a chain of 2-isogenies, we actually need the correct square roots $\theta_i^{A_1}(0)$.

A big part of this article is to optimize the formula to obtain these correct square roots. Let us explain the main idea, using $g = 2$ as an exemple. We have the theta null point $(a : b : c : d)$, and we can easily compute the squares of the dual coordinates of the isogenous theta null point $(A : B : C : D)$ via $(A^2 : B^2 : C^2 : D^2) = H \circ S(a : b : c : d) = H(a^2 : b^2 : c^2 : d^2)$ where H is the Hadamard transform and S is the squaring operation, ie $A^2 = a^2+b^2+c^2+d^2, B^2 = a^2-b^2+c^2-d^2, C^2 = a^2+b^2-c^2-d^2, D^2 = a^2-b^2-c^2+d^2$. If we have suitable points of 4-torsion T_1, T_2, then $(AB : CD : AB : CD) = H \circ S(T_1)$, $(AC : BD : AC : BD) = H \circ S(T_2)$, $(AD : BC : BC : AD) = H \circ S(T_1 + T_2)$. This is not enough to recover (A, B, C, D) because we are dealing with projective coordinates. What we really need is to recover (AB, CD, AB, CD) exactly. This can be done via a normalisation procedure. In other words, computing the isogenous theta constant can be done from the coordinates on some points of 4-torsion and a suitable normalisation procedure. This is not specific to the case $\ell = 2$, as mentioned above the general case of ℓ prime to n is [LR12; CR15; LR15b; LR22a] and the relatively straightforward adaptation (assuming that we are given *compatible* points of 4-torsion) to all cases is in [Rob10, Proposition 6.3.5; Rob21, Remarks 2.10.3, 2.10.7 and 2.10.14], and a more detailed algorithm for $\ell = n = 2$ given in [DLRW23, Appendix C.2].

The normalisation procedure exploit the (algebraic) Riemann relations, as constructed by Mumford in [Mum66] (see also [Rob10, Théorème 4.4.6]). These Riemann relations follow from the duplication formula, whose algebraic version was proved by Mumford in [Mum66] (see also [Rob10, Théorème 4.4.3]). The duplication formula is particularly well suited for the algorithmic of 2-isogenies, and in these notes we will exploit it as much as possible in order to speed up the generic algorithm working for any ℓ.

We describe two optimisations compared to [DLRW23, Appendix C.2].

(1) To compute the correct square roots, the equations in [DLRW23, § C.2] (derived from the duplication formula, see [Rob10, § 4.3]) require 2^g 4-torsion points in $K[4]$ (suitably normalised from our 8-torsion points), including the theta null point. This means that when we decompose ϕ, we need to push along 2^g-points at each step (or more precisely compute the isogenous theta null point and then push $2^g - 1$ points). In this note we give a new algorithm that only require the g generators of $K[4]$ along with the theta null point. So once we have computed the isogenous theta null point, we only require to push g points for the next step. The total gain is almost $(2^g - 1)/g$: while there is no difference for $g = 1$, for $g = 2$ we go from needing to keep track of 3 (non null) points to only 2, and for $g = 4$ from 15 points to only 4.

(2) Still to compute the correct square roots, a normalisation procedure is applied in [DLRW23, § C.2] (described in more details in [Rob10, § 6.3, § 7.4]) to some points of 8-torsion in $K[8]$. This normalisation procedure amount to choosing some "correct" choice of affine lift; and it is repeated for each 2-isogeny ϕ_i: for ϕ_2 we will normalise points of 8-torsion in $\phi_1(K)[8]$ and so on. Instead, we propose to

normalize once and for all the g generators T_i of K. Essentially this amount, once an affine lift of the theta null point of A is chosen, to choose consistent lifts of the T_i with respect to this lift. This means that from now on, all our algorithm have to work on affine lifts. Luckily all our algorithms are derived from the Riemann relations and duplication formula which naturally preserve this compatibility, so the compatibility is already "baked-in". Note that if compatible lifts $\widetilde{0_A}, \widetilde{T_i}$ are chosen, then the lifts $\lambda \star \widetilde{0_A}, \star\widetilde{T_i}$ are still compatible as long as λ does not depend on i. This allows for some optimisation: for instance it is harmless to choose a different normalisation of the theta null point of A_1, as long as this different normalisation is taken into account when pushing points.

The main advantage of normalising generators of K at the start is that when A is a product of elliptic curves, the normalisation procedure can be done in dimension 1.

Points on an abelian variety in the theta model are represented by projective points, but as explained above, at various points in the isogeny computations we need to work with affine lifts. All our algorithms will be on affine lifts by default; the projective version follows trivially.

3. The two torsion on a level 2 theta structure

Let $(A, \mathscr{L}, \Theta_A)$ be a principally polarised abelian variety with a symmetric theta structure of level 2. Let $0_A = (a_i)_{i \in Z(\overline{2})}$ be the theta null point.

The translation map by points of two torsion is defined as follows: the two torsion is isomorphic to $Z(\overline{2}) \times \hat{Z}(\overline{2})$, with $Z(\overline{2}) = \mathbb{Z}^g/2\mathbb{Z}^g$, and $\hat{Z}(\overline{2})$ its dual. If $P = (x_i)$ is an affine lift of a point on A, and T the two torsion point corresponding to (j, χ), $P + T = (\chi(i)x_{i+j})$.

Applying this to the theta null point, we recover the theta coordinates of the points of 2-torsion. Fixing the canonical basis $(e_1, \dots, e_g)$ of $Z(\overline{2})$, and letting f_i be the dual character of e_i, via our identification above the basis (e_i, f_i) is the canonical symplectic basis of the 2-torsion induced by theta theta structure.

Example 3.1. When $g = 1$, the theta null point is given by $(a, b) = (a_0, a_1)$. We have $e_1 = (b, a), f_1 = (-a, b)$. Dimension 1 is special in that we also have an explicit description of points of 4-torsion: $e_1' = (1 : 1)$ is the canonical point of 4-torsion above e_1 (the other one is $e_1' + f_1 = (-1 : 1)$), and $f_1' = (1 : 0)$ the canonical point of 4-torsion above f_1 (the other one is $f_1' + e_1 = (0 : 1)$).

Example 3.2. When $g = 2$, the theta null point is given by $(a_{00}, a_{01}, a_{10}, a_{11})$. We have $e_1 = (a_{01}, a_{00}, a_{11}, a_{10}), e_2 = (a_{10}, a_{00}, a_{11}, a_{01})$ and $e_1 + e_2 = (a_{11}, a_{10}, a_{01}, a_{00})$. We have $f_1 = (a_{00}, -a_{01}, a_{10}, -a_{11}), f_2 = (a_{00}, a_{01}, -a_{10}, -a_{11})$ and $f_1 + f_2 = (a_{00}, -a_{01}, -a_{10}, a_{11})$.

4. The Hadamard transform

Let H be the Hadamard matrix, given by $H_{i,\chi} = \chi(i)$. The action of H corresponds to the action of the modular matrix $\mathcal{S} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$; in particular this transpose the e_i with the f_i.

Starting with the theta coordinate θ_i, the coordinates θ'_χ resulting from the action of H are called the dual theta coordinates.

Example 4.1. When $g = 1, H(x, z) = (x + z, x - z)$.

One needs to be careful that $H \circ H = 2^g$ Id. This is not a problem in projective coordinate, but in affine coordinate we need to use $H^{-1} = H/2^g$.

5. The duplication formula

Let $K = \langle f_1, \ldots, f_g \rangle$ and $f : A \to B = A/K$ the quotient. There are several ways to descend $\mathcal{L}^2$ to a principal polarisation $\mathcal{M}$ on B, but they all give the same totally symmetric line bundle $\mathcal{M}^2$ which is also the descent of $\mathcal{L}^4$ by the unique symmetric lift of K in the theta group $G(\mathcal{L}^4)$ which extends to a totally isotropic subgroup. Fix a compatible symmetric theta structure of level 2 on B.

Define the operation $\star$ by $(x_i) \star (y_i) = (x_i y_i)$. As a special case of the duplication formula, we have:

$$\theta^A(P+Q) \star \theta^A(P-Q) = H(\theta^{',B}(f(P)) \star \theta^{',B}(f(Q))) \tag{1}$$

$$H(\theta^A(\tilde{f}(R)) \star \theta^A(\tilde{f}(S))) = \theta^{',B}(R+S) \star \theta^{',B}(R-S) \tag{2}$$

This is the key for all our formula. First, using $Q = 0$, we can compute the image of a point by f via the operations

$$P = (\theta_i(P)) \xrightarrow{S} (\theta_i^2(P)) \xrightarrow{H} (\theta_\chi^{',B}(f(P))\theta_\chi^{',B}(0)) \xrightarrow{C_1} \theta_\chi^{'B}(f(P)) \xrightarrow{H} \theta_i^B(f(P)) \tag{3}$$

(Note: here and in what follows, we are probably off by some factor 2^g here; as long as this factor is uniform across all points this is ok. Also here S is the squaring map, not the modular matrix $\mathcal{S}$ from before).

Here the constants C_1 are given by $(1/\theta_\chi^{',B}(0))$, the inverse of the dual theta coordinates of the theta null point on B. It is easy to compute their squares: $\theta_\chi^{',B}(0)^2 = H(\theta_i^A(0)^2)$.

The remainder of this paper is devoted to compute the correct square roots of these squares. Note that compared to [DLRW23, Appendix C.2] we consider the isogeny with kernel $\langle f_i \rangle$ instead of the one with kernel $\langle e_i \rangle$. (We made a different choice in [DLRW23, Appendix C.2] because we used the analytic formalism, where the above choice was slightly more natural. In the algebraic formalism, it is slightly more natural to use our choice here. This does not matter much, because via H we can go from the coordinates to the dual coordinates. That's why our formula differ from [DLRW23, Appendix C.2] by the conjugation by H.)

6. Differential additions

We can also compute differential additions on A this way. First we compute $f(P)$ and $f(Q)$ using the differential addition formula as above, ie using them on the couple $(P, 0)$ and $(Q, 0)$. Then we use them again (in the other direction) to recover $\theta_i^A(P+Q)\theta_i^A(P-Q)$ from $(f(P), f(Q))$.

We actually don't need the $\theta_\chi^{',B}(0)$, only their square, the trick is to start with P and only do the map $H \circ S$ to get $(\theta_\chi^{',B}(f(P))\theta_\chi^{',B}(0))$, same with Q. Then we apply the $\star$ operation on these coordinates to get $(\theta_\chi^{',B}(f(P))(\theta_\chi^{',B}(f(Q))\theta_\chi^{',B}(0)^2)$, and now we can use C_1^2 to clear the extra factor $(\theta_\chi^{',B}(0)^2)$. Using $Q = P$ we get the doubling map.

From the doubling and differential addition map, we can use the Montgomery ladder to compute the scalar multiplication on affine coordinates.

Example 6.1. When $g = 1$, $P = (x : z)$, we compute $f(P) = (r : s)$ by doing $(x : z) \xrightarrow{S} (x^2 : z^2) \xrightarrow{H} (x^2 + z^2 : x^2 - z^2) \xrightarrow{C} ((x^2 + z^2)/A : (x^2 - z^2)/B)$. We can compute $f(Q) = (u : v)$ in a similar way. Then $H((P+Q) \star (P-Q)) = H(f(P)) \star H(f(Q))$, so we compute $((r+s)(u+v) : (r-s)(u-v))$ (at this step we only need $(A^2 : B^2)$ which can be computed via $(A^2 : B^2) = (a^2 + b^2 : a^2 - b^2)$) and apply H to it to recover $(x(P+Q)x(P-Q) : z(P+Q)z(P-Q))$.

We also recover exactly Gaudry's addition formula for $g = 2$.

The doubling and differential addition algorithm assume that we are in the generic case and that none of the coordinates are zero. The general case is treated in [LR16], another solution is to apply any linear change of variable coming from the symplectic modular action (e.g., the action of H), we refer to Appendix B for the algebraic description of this action.

7. Normalising points

In [DLRW23, Appendix C] we explain how to use points of 4-torsion to compute the correct choice of $\theta'^{,B}_{\chi}(0)$. A key step is a normalisation procedure, and we actually need the points of 8-torsion to correctly normalize our points of 4-torsion (see also [Rob10; Rob21; LR22a]).

Lemma 7.1. *Let $\widetilde{P}$ be an affine point. Then $m(\lambda \star \widetilde{P}) = \lambda^{m^2} \star (m\widetilde{P})$.*

Let T be a 2-torsion point in our kernel $K = \langle f_1, \dots, f_g\rangle$. Let T'' be a point of $4m$-torsion above T, ie $T = mT''$. Write $2m = 2m_1 + 2$. We have $(m_1 + 2)T'' = -(m_1 T'') + T$.

Definition 7.2. Fix an affine lift $\widetilde{T''}$ of T''. We say that $\widetilde{T''}$ is normalised if $(m_1 + 2)\widetilde{T''} = -(m_1\widetilde{T''}) + \widetilde{T}$, where the action of translation by $\widetilde{T}$ is the affine one described in Section 3.

Lemma 7.3. *Fix an arbitrary affine lift $\widetilde{T''}$. By computing $(m_1 + 2)\widetilde{T''}$ and $(m_1)\widetilde{T''}$, we can find an equation $\lambda^{4m} = C$ such that for any solution λ, $\lambda \star \widetilde{T''}$ is normalised.*

Proof. Follows from Lemma 7.1. □

Example 7.4. Assume T'' is a point of 2^n-torsion above T. Applying the normalisation procedure of Lemma 7.3 to an arbitrary lift $\widetilde{T''}$, we get that $\lambda \star \widetilde{T''}$ is normalised for λ satisfying some equation $\lambda^{2^n} = C$. Then $2^{n-2}(\lambda \star \widetilde{T''}) = \lambda^{2^{2n-4}} \star (2^{n-2}\widetilde{T''})$ by Lemma 7.1.

It follows that if $n \geq 4$, the point T'' uniquely determines an affine lift $\widetilde{T'}$ of the point of 4-torsion $T' = 2^{n-2}T''$ above T. If $n = 3$, $\widetilde{T'}$ is uniquely determined up to a sign. Since the isogeny formula starts by the square operator S, this sign won't matter, so $n = 3$ is enough to normalize our points of 4-torsion.

Example 7.5. Let us start with $T' = (1 : 0)$, the canonical point of 4-torsion above $T = (a : -b)$ in dimension 1. We take the lift $\widetilde{T'} = (1, 0)$. We compute $2\widetilde{T'} = (\frac{a}{A^2B^2}, \frac{-b}{A^2B^2})$. The correct normalisation is thus $\lambda \star \widetilde{T'} = (\lambda, 0)$ with $\lambda^4 = A^2B^2$.

8. The choice of the theta constant for a 2-isogeny

When we apply Equation (3) to compute the image of a point by our isogeny, we have fixed the kernel of our 2-isogeny f to be $K = \langle f_1, \dots, f_g\rangle$.

If we start with another kernel, we need to apply an automorphism of the theta structure so that K corresponds to the $\langle f_1, \dots, f_g\rangle$ of the new theta null point; for instance if $K = \langle e_1, \dots, e_g\rangle$ the automorphism is the one given by the Hadamard transform.

A general procedure is as follow. First recall that the theta null point is induced by a symplectic basis of the 4-torsion. Fix such a basis $(e'_1, \dots, e'_g, f'_1, \dots, f'_g)$ inducing our theta null point. Let $(T_1, \dots, T_g)$ be a basis of K, choose any isotropic basis $(T'_1, \dots, T'_g)$ of 4-torsion point above the T_i, and complete the T'_i via a symplectic basis $(S'_1, \dots, S'_g, T'_1, \dots, T'_g)$. Compute the symplectic base change of matrix from $(e'_1, \dots, e'_g, f'_1, \dots, f'_g)$ to $(S'_1, \dots, S'_g, T'_1, \dots, T'_g)$, one can use the Weil pairing (an algorithm in theta coordinate is given in [LR10; LR15a; Rob21]) to compute this matrix M. Then apply the theta transformation formula for M.

It remains to explain how to fix $(e'_1, \dots, f'_g)$. As explained in the introduction, the general case will be treated in an upcoming article with David Lubicz. For our applications, we will use that A is a product of elliptic curve, so we only need to deal with $g = 1$ and use the fact that the product theta structure behaves as expected with respect to the symplectic basis:

Lemma 8.1. *If 0_A is induced by a basis $(e'_1, \dots, e'_{g_1}, f'_1, \dots, f'_{g_1})$ on A and 0_B is induced by a basis $(m'_1, \dots, m'_{g_2}, n'_1, \dots, n'_{g_2})$ on B, then the theta null point $(\theta_i^A(0)\theta_j^B(0))$ of the product theta structure is induced by the symplectic basis $((e'_i, 0), \dots (0, m'_j), \dots (f'_i, 0), \dots (0, n'_j))$ on $A \times B$.*

We are thus reduced to give a compatible symplectic basis of the four torsion in dimension 1. This case is easy because on the theta model of level 2 we always have the full 4-torsion (on the Kummer) when $g = 1$ (this is specific to the dimension 1 case).

Lemma 8.2. *On a theta model in dimension 1, a symplectic basis of $E[4]$ is given by $T'_1 = (1 : 1), T'_2 = (1 : 0)$.*

Proof. Let $(a : b)$ be the theta null point. Above $T_1 = (b : a)$ we have two points of 4-torsion (on the Kummer): $(1 : 1)$ and $(-1 : 1)$. Only one of the two is compatible with the theta structure. To determine which we use an idea due to David Lubicz: from a compatible four torsion point $T' = (u : v)$ we can compute a level 4 isogenous theta null point $(a, \lambda u, b, \lambda v)$, for λ an appropriate normalisation factor (see [Rob10; Rob21]). This level 4 theta null point has to be symmetric, which implies $\lambda u = \lambda v$. So we have $T'_1 = (1 : 1)$.

Above $T_2 = (-a : b)$ we have two points of 4-torsion: $(1 : 0)$ and $(0 : 1)$. The Hadamard transform of the first one is $(1 : 1)$ while for the second one we get $(1 : -1)$, so the correct compatible point is $T'_2 = (1 : 0)$. ☐

We can use the lemma above to convert a basis of 4-torsion (T'_1, T'_2) in a Montgomery model to a theta null point induced by this basis.

Lemma 8.3. *Let E be a Montgomery curve. Let $T'_1 = (1 : 1)$ be the canonical point of 4-torsion on the Kummer line in the Montgomery model. Let $T'_2 = (r : s)$ be another point of 4-torsion (with $2T'_2 \neq 2T'_1$). Then the theta null point associated to the basis (T'_1, T'_2) is $(a : b) = H(T'_2) = (r + s, r - s)$.*

Proof. This follows by looking at the ramification of the Kummer map $E \to E/\pm 1$ on our different models, see [BRS23]. ☐

We can use the above lemma on an arbitrary curve E with two explicit points T'_1, T'_2 of 4-torsion (with $2T'_1 \neq 2T'_2$) by first converting E to Montgomery form with T'_1 sent to $(1 : 1)$ and T_1 to $(0 : 1)$. This map is given by the homography $x \mapsto (x - x_0)/\beta$ with $x_0 = x(2T'_1)$ and $\beta = x(T'_1) - x_0$. See Appendix A for more details on converting to theta coordinates.

Example 8.4 (Dimension 2)**.** If we have two elliptic curves E_1, E_2 given by the theta constants $(a_1 : b_1), (a_2 : b_2)$, then the theta constant on $E_1 \times E_2$ is $(a_1a_2 : a_1b_2 : a_2b_1 : b_1b_2)$. And if $P_1 = (x_1 : z_1) \in E_1$, $P_2 = (x_2 : z_2) \in E_2$, $(P_1, P_2) = (x_1x_2 : x_1z_2 : x_2z_1 : z_1z_2) \in E_1 \times E_2$.

Remark 8.5. We briefly explain how the general case would work.

Let T'_i be a point of 4-torsion above $T_i \in K_2$. Then we have $T'_i + T_i = -T'_i$, hence in level 2, since we are on the Kummer, $(\theta_j(T'_i + T_i)) = (\theta_j(T'_i))$ in projective coordinates. From the action of T_i described in Section 3, we get that either $\theta_j(T'_i) = 0$ for all j such that $\chi_i(j) = 1$ or $\theta_j(T'_i) = 0$ for all j such that $\chi_i(j) = 0$. The compatibility conditions holds if we are in the first case for all i (this follows by a counting argument).

For instance, when $g = 2$, we should have $T'_1 = (x : 0 : z : 0)$ and $T'_2 = (u : v : 0 : 0)$. If $T'_1 = (0 : x : 0 : z)$ or $T'_2 = (0 : 0 : u : v)$ then these points are not compatible. This criteria can be used to check if our symplectic base change was correct.

Another difficulty in the general case, is that the 4-torsion is not immediately accessible (unlike the case for a product of elliptic curve). So we would first need to compute a symplectic basis (e'_i, f'_i) of the 4-torsion above the one (e_i, f_i) of the 2 torsion compatible with our current theta null point to apply the above strategy. In dimension 2 a method is described in Section 16.6, but this involves square roots.

We suggest the following alternative strategy: work only with the 2-torsion, and compute the symplectic base change matrix $M \in \mathrm{Sp}_{2g}(\mathbb{Z}/2\mathbb{Z})$. It is easy to express our 2-torsion points T_i in terms of the e_i, f_i: essentially the Weil pairing is trivial to compute in level 2 (namely check the translation which match the coordinates up to a sign, and then look at the signs). Lift M to an arbitrary matrix $\widetilde{M} \in \mathrm{Sp}_{2g}(\mathbb{Z}/4\mathbb{Z})$. While the points $\widetilde{M}.T_i$ will be correct by construction, the points $\widetilde{M}.T'_i$ probably won't be correct: the zeros will not be in the right position. But we can correct this via the action of $\Gamma(2,4) / \mathrm{Sp}_{2g}(\mathbb{Z}/2\mathbb{Z})$, essentially this acts like the translation of the 2-torsion so the correction is easy.

9. The choice of theta constants for a 2^n-isogeny

Let K be an isotropic 2^n-kernel of rank g on A. We want to first compute the quotient $f : A \to B = A/K[2]$, and then compute $f(K)$ in B, and recurse our formula.

First, our kernel $K[2]$ has to be compatible with our chosen theta null point on A, as explained in Section 8. Then as explained in Section 5, it is easy to recover the squares of the dual theta coordinates of B.

While we can prove that any choice of square roots of these coordinates correspond to an honest (dual) theta null point on B when $g \leq 2$, this is no longer the case in higher dimension, once we have fixed some square roots the other ones have to satisfy some compatibility condition.

Most importantly, our choice of theta constant on B determines the next 2-isogeny. But we want to compute the isogeny with kernel K, so our next isogeny has to be $f(K[4])$! So we do not want arbitrary (compatible) square roots anyway, but the ones which correspond to $f(K[4])$.

There is one remaining subtlety. Our theta constant on A determines a bit more than the symplectic basis of 2-torsion (hence the kernel of the first 2-isogeny). It is enough to fix a symplectic basis of the 4-torsion (and several such basis will determine the same theta null point). This means that we also require some compatibility between our kernel K and our theta null point on A: let $(f'_1, \dots, f'_g)$ be a basis of $K[4]$ with $f_i = 2f'_i$. Our first compatibility condition was that f_i is the canonical point of 2-torsion induced by our theta structure as described in Section 3, ie $K[2]$ and our theta null point are compatible. We require furthermore that our theta null point is induced by some symplectic basis $(e'_1, \dots, e'_g, f'_1, \dots, f'_g)$, in which case we say that $K[4]$ and our theta null point are compatible.

But now for our choice of sign for the theta null point of B, we want this theta null point to be compatible with $f(K)[4]$. Since $f(K)[4] = f(K[8])$, we will also need to use the points of 8-torsion in the kernel to fix our sign choice.

Given $K[8]$, it is possible to use these points of 8-torsion to normalize the points of 4-torsion up to factors $\lambda^2 = C$ as explained in Section 7. Since the choice of signs for the dual theta null point of B depends only on the square of the theta coordinates of these points of 4-torsion (see [DLRW23, Equation (6) and (8)] or the duplication formula in [Rob10, Théorème 4.4.3]) this is enough to completely determine the theta null point of B.

A problem remains at the last 2 steps of the isogeny chain, when we only have access to 4-torsion points (resp. 2-torsion points) in K. It is possible to show that when the 2-torsion on B is not fixed by $K[4]$, there are $g(g+1)/2$ possible choice of signs for the dual theta null point of B. This follows by looking at the possible automorphisms of the theta structure as in [Rob10, § 6.3]. If we have $K[4]$ but not $K[8]$, we only have g possible choice of signs: the isotropic part $f(K[4])$ of the 2-torsion on B is fixed but we can change the symmetric lifts above them. These sign can be determined as follow: take $T'_1, \dots, T'_g$ a basis of $K[4]$ and normalize these points, we obtain equations $\lambda_i^4 = C_i$. The points $T'_i + T'_j$ can then be normalised up to some equation $\lambda_{ij}^2 = C_{ij}$, and from these all other points can be computed from extended Riemann relations, notably the three way additions. Since the theta null point on B only depend on the squares of the theta coordinates of the normalised points of 4-torsion, we obtain our g-choice of sign corresponding to the choices of $\lambda_i^2 = \pm\sqrt{C_i}$.

There are many reason to want more control on these last two steps. Typically for cryptographic applications, the codomain B of ϕ is also a product of elliptic curves, and we want to map back to these curves. This is easy to do if the theta null point $\theta_i(0_B)$ comes from a product theta structure, but there is no reason for this to be the case. One would then need to take an automorphism of the theta structure which brings it to a product theta structure. Also it is often the case that the isogeny $\phi : A \to B$ is split as an isogeny $\phi_1 : A \to C$ and a dual isogeny $\widetilde{\phi}_2 : B \to C$. One then need to glue together the theta null point computed on C from ϕ_1 and $\widetilde{\phi}_2$, they have no reason to be induced by the same theta structure, hence be the same. Again they will differ by some automorphism of the theta structure. As carefully explained in [DLRW23, § C.1], by keeping track of a bit more torsion it is possible to compute in advance the correct automorphism of the theta structure that we need in these computation. This means that our algorithm will start with K' an isotropic kernel of rank g of $A[2^{n+2}]$, and we compute the quotient $B = A/K$ where $K = K'[2^n]$ and the theta null point on B is the one induced by the theta null point on A along with our choice of K' (half the information given by the theta null point on A is killed by our isogeny ϕ, and K' allows precisely to uniquely recover this missing information).

10. Normalising the points for a 2^n-isogeny

From now on we suppose that we have K' a maximal isotropic subgroup of rank g of $A[2^{n+2}]$, $T'_1, \dots, T'_g$ generators of K', and we want to compute the isogeny $K = K'[2^n]$ with generators $T_i = 4T'_i$. As explained in the introduction, we will normalise once and for all the T'_i. The computations in Section 7 show that it is enough to completely normalise the points of 4-torsion in each K_i (up to a sign at the very last step when computing ϕ_n, but as always this sign does not matter because we only need the squares of these coordinates). So each T_i will give an equation $\lambda_i^{2^{n+2}} = C_i$, and we keep track of these normalisation factors at each step of our algorithm. Once again, from Section 7 we know that we will only need the values of the C_i and we never need to know the λ_i.

This global normalisation of K' is particularly useful when A is a product of g elliptic curves. Indeed, the normalisation procedure essentially boils down to a scalar multiplication (computed via a Montgomery ladder), and it is slightly faster to compute g such multiplications in dimension 1 than one in dimension g via the product theta structure. Furthermore, most cryptographic applications come from Kani's lemma, so that A is of the form $E_1^{g/2} \times E_2^{g/2}$. So we really only need to normalise 4 points in dimension 1 (a basis of $E_i[2^{n+2}]$) rather than g^2, and then keep track of our normalisations across each copy of E_i.

11. Computing the isogenous theta null point

Let K be our kernel, assume that it is compatible with the theta null point on A, and that we have computed normalisation $\widetilde{P_i}$ of a basis $(P_1, \dots, P_g)$ of $A[4]$ (either from $K[8]$ or via a global normalisation). Let $B = A/K[2]$. We can use these normalised points to compute the correct choice of square roots for ${\theta'}^B_\chi(0)$. Let us first recall the formula from [DLRW23, § C.2] (which as already mentioned result from the duplication formula [Rob10, Théorème 4.4.3]), remembering that we need to conjugate them by H in our situation because here we consider the "dual" kernel on A.

In the original algorithm, we actually need $\widetilde{P_t}$ for any $t \in Z(\overline{2})$, where $P_t = \sum_{i=1}^g t_i P_i$. First use H to convert $\theta_i(\widetilde{P_t})$ to $\theta'_\chi(\widetilde{P_t})$, we then have:

$$ {\theta'}^B_{\chi_t} = \sum_\chi \theta'_\chi(P_t)^2 $$

where χ_t is the character dual to t.

Example 11.1. When $g = 1$, we have $\widetilde{T}_0 = (a, b)$ a lift of the theta null point $(a : b)$. We have ${\theta'}^B(0) = (A, B)$, with $A^2 = a^2 + b^2$, $B^2 = a^2 - b^2$ by Section 5. We have $T_1 = (1 : 0)$ (this is the only compatible point of 4-torsion above $(-a : b)$, the other one is $(0 : 1)$ and is not compatible as we will see shortly), and $\widetilde{T}_1 = (\lambda, 0)$ with $\lambda^4 = A^2B^2$. So $\theta'(\widetilde{T}_0) = (a + b, a - b)$, $\theta'(\widetilde{T}_1) = (\lambda, \lambda)$, and our formula above shows that ${\theta'}^B_0(0) = (a + b)^2 + (a - b)^2 = 2a^2 + 2b^2 = 2A^2$, and ${\theta'}^B_1(0) = \lambda^2 + \lambda^2 = 2\lambda^2 = 2AB$. The point $(2A^2 : 2AB) = (A : B)$, and we choose for affine lift on the dual theta null point of B the point $(1, B/A)$.

Given a point $P = (x : z)$, as explained in Section 5 its image by the isogeny in theta coordinates on B is given by $(x : z) \xrightarrow{S} (x^2 : z^2) \xrightarrow{H} (x' = x^2 + z^2 : z' = x^2 - z^2) \xrightarrow{C} (x" = x'/A, z" = z'/B) \xrightarrow{H} (x" + z", x" - z")$.

When working with projective coordinate, we only need the projective point $C = (1/A : 1/B)$. However when working with affine coordinates, since we want to send (a, b) to our choice of $(1, B/A)$, we need to take $C = (1/A^2, 1/AB)$. Let $(a_2 : b_2)$ be the theta null point on B.

We remark that T_1 is sent to $(-a_2 : b_2)$, the kernel of the next isogeny, while $(0 : 1)$ is sent to $(-b_2 : a_2)$, which is not the kernel of the next isogeny.

We now describe our optimisation. Let $i \in Z(\overline{2})$, and $\widetilde{T_i}$ be the corresponding normalised point of 4-torsion. Its image by our isogeny f has to be the normalised point of 2-torsion induced by i given in Section 3. Since this image is given by the operator $H \circ C \circ H \circ S$ with $C = 1/{\theta'}^B_i(0)$, this means that if we apply $C \circ H \circ S$ to $\widetilde{T_i}$, we obtain the point $({\theta'}^B_{\chi_i + \chi}(0))_\chi$, where χ_i is the character dual to χ. So $H \circ S(\widetilde{T_i}) = ({\theta'}^B_{\chi_i+\chi}(0){\theta'}^B_\chi(0))$.

In particular, applying this to all the T_i, we recover all two by two product $({\theta'}^B_\chi(0){\theta'}^B_{\chi'}(0))$, which gives an alternative way to recover the theta null point of B. But actually, it is enough to recover this theta null point by applying $H \circ S$ to only a basis $\widetilde{T}_1, \dots, \widetilde{T_g}$ along with the theta null point $\widetilde{T}_0$. Indeed, an explicit computation shows that we recover all ${\theta'}^B_\chi(0)/{\theta'}^B_0(0)$ for all characters χ of Hamming weight 1, then 2, and so on.

Example 11.2. When $g = 1$, we have $\widetilde{T_1} = (\lambda, 0)$ with $\lambda^4 = A^2B^2$. We apply $H \circ S$ to $\widetilde{T_0} = (a, b)$ to get (A^2, B^2), and to $\widetilde{T_1}$ to get $(\lambda^2, \lambda^2) = (AB, AB)$. From this we recover B/A, hence (a_2, b_2).

We remark that applying $H \circ S$ to $(0, \lambda)$ gives $(AB, -AB)$. In fact, for all sign choices of $(A : B)$, while $f(1 : 0) = (a_2 : -b_2)$, the kernel of the next isogeny, we have $f(0 : 1) = (b_2 : a_2)$.

The case $g = 1$ is particular in that we have some explicit points of 4-torsion in the theta model. So in that case, rather than looking at the preimage of our isogeny of the points of 2-torsion, we could look at the preimage of $(1 : 0)$. So let $T' = (r : s)$ a point of 8-torsion, this point fixes the (projective) theta null point of B. In particular, we should have $f(T') = (1 : 0)$. Doing the computation, we get $(r^2 + s^2 : r^2 - s^2) = (A : B)$. So in that case we can directly recover $(A : B)$ from T' in projective coordinates, without requiring any normalisation. The sign choice of $(A : B)$ induced from T' ensures that $f(T') = (1 : 0)$ becomes the compatible point of 4-torsion.

Example 11.3. When $g = 2$, let $(a_{00}, a_{01}, a_{10}, a_{11})$ be our theta null point on A, $(a'_{00}, a'_{01}, a'_{10}, a'_{11})$ our theta null point on B, and $(A_{00}, A_{01}, A_{10}, A_{11}) = H(a'_{00}, a'_{01}, a'_{10}, a'_{11})$ our dual theta null point on B. Recall that H is given by $H(x_{00}, x_{01}, x_{10}, x_{11}) = (x_{00} + x_{01} + x_{10} + x_{11}, x_{00} + x_{01} - x_{10} - x_{11}, x_{00} - x_{01} + x_{10} - x_{11}, x_{00} - x_{01} - x_{10} + x_{11})$.

Assume that we have normalised our points of 4-torsion $\widetilde{T_i}$. Recall that the isogeny is given by $f = H \circ C \circ H \circ S$ with $C = (1/A_i)$, and let $g = H \circ f = C \circ H \circ S$ the isogeny given in dual theta coordinates on B, and $h = H \circ S$ the isogeny given in twisted dual theta coordinates on B. We have $f(\widetilde{T_0}) = f(a_{00}, a_{01}, a_{10}, a_{11}) = (a'_{00}, a'_{01}, a'_{10}, a'_{11})$, so $g(\widetilde{T_0}) = (A_{00}, A_{01}, A_{10}, A_{11})$, and $h(\widetilde{T_0}) = (A_{00}^2, A_{01}^2, A_{10}^2, A_{11}^2)$.

We know that $f(\widetilde{T_1}) = (a'_{00}, -a'_{01}, a'_{10}, -a'_{11})$, so $g(\widetilde{T_1}) = (A_{01}, A_{00}, A_{11}, A_{10})$, and $h(\widetilde{T_1}) = (A_{00}A_{01}, A_{00}A_{01}, A_{10}A_{11}, A_{10}A_{11})$.

We know that $f(\widetilde{T_2}) = (a'_{00}, a'_{01}, -a'_{10}, -a'_{11})$, so $g(\widetilde{T_2}) = (A_{10}, A_{11}, A_{00}, A_{01})$, and $h(\widetilde{T_2}) = (A_{00}A_{10}, A_{01}A_{11}, A_{00}A_{10}, A_{01}A_{11})$.

Finally, we know that $f(\widetilde{T_1 + T_2}) = (a'_{00}, -a'_{01}, -a'_{10}, a'_{11})$, so $g(\widetilde{T_1 + T_2}) = (A_{11}, A_{10}, A_{01}, A_{00})$, and $h(\widetilde{T_1 + T_2}) = (A_{00}A_{11}, A_{01}A_{10}, A_{01}A_{10}, A_{00}A_{11})$.

We see that the four points $\widetilde{T_0}, \widetilde{T_1}, \widetilde{T_2}, \widetilde{T_1 + T_2}$ allow to recover all 2 by 2 products A_iA_j. But the first three are already enough: dividing by A_{00}^2, we recover A_{01}/A_{00} from $\widetilde{T_1}$ and A_{10}/A_{00} from $\widetilde{T_2}$, which allows us to recover A_{11}/A_{00} from either of these two points.

Example 11.4. Assume that $g = 3$, and lets look at the image of the operator $h = H \circ S$, i.e, the isogeny f given in twisted dual theta coordinates on B.

We compute

$$h(\widetilde{T_0}) = (A_{000}^2, A_{001}^2, A_{010}^2, A_{011}^2, A_{100}^2, A_{101}^2, A_{110}^2, A_{111}^2),$$

$$h(\widetilde{T_1}) = (A_{000}A_{001}, A_{001}A_{000}, A_{010}A_{011}, A_{011}A_{010}, A_{100}A_{101}, A_{101}A_{100}, A_{110}A_{111}, A_{111}A_{110}),$$

$$h(\widetilde{T_2}) = (A_{000}A_{001}, A_{001}A_{011}, A_{010}A_{000}, A_{011}A_{001}, A_{100}A_{110}, A_{101}A_{111}, A_{110}A_{100}, A_{111}A_{101}),$$

$$h(\widetilde{T_3}) = (A_{000}A_{100}, A_{001}A_{100}, A_{010}A_{110}, A_{011}A_{111}, A_{100}A_{000}, A_{101}A_{001}, A_{110}A_{010}, A_{111}A_{011}).$$

Looking at the image of the $\widetilde{\sum \varepsilon_i T_i}$ we would also get all the 2 by 2 products A_iA_j, but these points are enough. We first recover A_{001}/A_{000}, A_{010}/A_{000}, A_{100}/A_{000}, then A_{011}/A_{000}, A_{101}/A_{000}, A_{110}/A_{000} and finally A_{111}/A_{000}.

12. The image of a point

We already saw how to compute the image of a point by the 2-isogeny f once we have the dual theta coordinates $\theta'^{B}_{\chi}(0)$ on B. Namely the formula is given by the operator $H \circ C \circ H \circ S$ where $C = 1/\theta'^{B}_{\chi}(0)$. This assume that these theta constants are non zero however.

In this section we explain how to deal with the annulation of some of these theta constants. This will typically be the case when the starting variety is a product of elliptic curves and the first isogeny a gluing isogeny.

Let $(\widetilde{T_1}, \dots, \widetilde{T_g})$ be our basis of normalised points in $K[4]$. Let P be a point on A, fix an arbitrary lift $\widetilde{P}$, and assume we have computed coherent lifts $\widetilde{P+T_i}$ relatively to $\widetilde{P}$ and the $\widetilde{T}_i$. Note that if $P \in K$ and we have already normalised all points in K, we can use these as normalisations.

The operator $h = H \circ S$ gives the image of P in terms of the twisted dual theta coordinates on B. In particular, if $\widetilde{Q} = f(\widetilde{P})$, we have $h(\widetilde{P}) = (\theta'^{B}_{\chi}(\widetilde{Q})\theta'^{B}_{\chi}(0))$, and for $i \in Z(\overline{2})$, $h(\widetilde{P+T_i}) = (\theta'^{B}_{\chi+\chi_i}(\widetilde{Q})\theta'^{B}_{\chi}(0))$. So we can use these points to recover all the coordinates of $h(\widetilde{P})$.

Example 12.1. When $g = 2$, and $\theta'^{B}_{\chi}(f(\widetilde{P})) = (x_{00}, x_{01}, x_{10}, x_{11})$, we compute $h(\widetilde{P}) = (A_{00}x_{00}, A_{01}x_{01}, A_{10}x_{10}, A_{11}x_{11}, h(\widetilde{P+T_1}) = (A_{00}x_{01}, A_{01}x_{00}, A_{10}x_{11}, A_{11}x_{10}, h(\widetilde{P+T_2}) = (A_{00}x_{10}, A_{01}x_{11}, A_{10}x_{00}, A_{11}x_{01}$, and $h(\widetilde{P+T_1+T_2}) = (A_{00}x_{11}, A_{01}x_{10}, A_{10}x_{01}, A_{11}x_{00}$.

We see that even if one of the dual isogenous theta null point A_i is zero, knowing the (affine) theta coordinates of $P, P+T_1, P+T_2$ still allows to compute $h(P)$.

13. The full algorithm

Let us summarize the steps to compute a 2^n-isogeny with kernel K.

(1) Start with a theta null point of level 2 and A induced by some explicit symplectic basis $(e'_1, \dots, e'_g, f'_1, \dots, f'_g)$ of the 4-torsion. This can be done using Section 8 when A is a product of elliptic curves.
(2) Let $v'_1, \dots, v'_g$ be a basis of $K[4]$, and complete this basis into a symplectic basis $(u'_1, \dots, u'_g, v'_1, \dots, v'_g)$. Let M be the symplectic matrix $(e'_1, \dots, e'_g, f'_1, \dots, f'_g)$ to $(u'_1, \dots, u'_g, v'_1, \dots, v'_g)$. Apply the theta transformation formula induced by M to get the linear change of variable inducing new theta coordinates compatible with our kernel K.
(3) Let $T_1, \dots, T_g$ be a basis of K. For reasons explained in Section 9, it is convenient to assume that we are given $T"_i$ an isotropic basis of $A[2^{n+2}]$ with $T_i = 4T"_i$. Using Section 7, normalise each $T"_i$ to get an affine point $\widetilde{T"}_i$. If A is a product of elliptic curves, it will be easier and faster to normalise before the linear change of variable from the preceding step, because the normalisation for the product theta structure can be done in dimension 1.
(4) Compute $2^n\widetilde{T"}_i$ using Section 6, and use this normalised basis of $K[4]$ to compute the first isogeneous theta null point using Section 11.
(5) Compute the image of the $\widetilde{T"}_i$ using Section 12.
(6) Go back to step Item 4, with n decremented by 1.

We will also look at the variant where instead of normalising the points of 2^{n+2}-torsion T''_i at the beginning, we will only normalize points of 8-torsion at each step. In this variant we compute $U_i = 2^{n-1}T''_i$ to get g points of 8-torsion, which we normalise using Section 7. We then compute the affine 4-torsion point $2\widetilde{U_i}$ to recover the isogeneous theta null point using Section 11. Then we compute the image of T_i'' using Section 12 as above, except that in this case we only need the projective image rather than the affine image since T_i'' is no longer normalised).

14. Complexity

Because of the dynamic nature of the algorithm optimising the number of isogeny images vs doubling, we need to plug parameters to compare algorithm. Still, we can do some naive complexity estimate to estimate the cost of the full isogeny computations with respect to the dimension.

14.1. The old algorithm.

For the isogeny algorithm of [DLRW23], the naive ratio was given by $\kappa 2^g 2^g$: 2^g points to track, each point using 2^g coordinates.

For the more refined estimation, we have the following complexities:

- Doubling a point still costs $2.2^g S + 2.2^g M = 4.2^g$ operation by points and computing the image of a point $2^g S + 2^g M = 2.2^g$ operations (without any inversions).
- Computing an isogenous theta null point costs $2^g(7.2^g - 2) - 2$ arithmetic operations (neglecting additions and soustractions). This involves computing the necessary inverse needed for the doublings and images of points.

The discrepancy with the more precise estimated ratio comes from the fact that computing the theta null points behave differently from computing the other $2^g - 1$ points.

2^n	$g = 1$	$g = 2$	$g = 4$	$g = 8$
2^{128}	8028	44328	850464	228774144
2^{216}	14476	80376	1546608	416370768
2^{250}	17060	94860	1826700	491877900
2^{305}	21350	118950	2292990	617612190
2^{372}	26576	148296	2861016	770779416
2^{486}	35904	200844	3879828	1045623348

g	Naive ratios	Estimated ratios
2	×4	×5.5
4	×64	×110
8	×16384	×29000

In these notes, we will try to minimise the number of inversions and divisions, since they are much more expensive than the other arithmetic operations (squares and multiplications). Also we will count one division as $1I + 1M$, so we will only track the number of inversions.

To normalize a point of 8-torsion T_i'', we compute $2T_i'', 3T_i''$. The computations of the theta coordinates of the T_i'' require some divisions: the duplication formula naturally give the $\theta_j(3T_i'')\theta_j(T_i'')$. But we don't actually need these divisions, $3T_i''$ is needed only for the normalisation constant, so we need just one of his coordinate. And we can compute this constant as $C = \theta_j(3T_i'')\theta_j(T_i'')/\theta_j(5T_i'')\theta_j(T_i'')$. Recall that $5T_i''$ is computed at $T_i'' + T_i$ where $T_i = 4T_i''$ is a point of 2-torsion (hence the translation is given by the explicit linear action of Section 3).

So the first doubling for $2T_i''$ (taking into account we are going to do a tripling) will cost $2^g S + 3.2^g M$ (using the fact that we precomputed the inverse of some theta constants). Then for computing one coordinate of $3T_i''$ we need $2^g S + 2^g M$. As an aside, this will compute the square of the coordinates of the 4-torsion points $2T''_i$, which are needed to compute the isogenous theta null point.

We then compute C by the equation above, this costs $2M + 1I$. Using this constant to normalize our sum adds $1M$.

Since we normalize $2^g - 1$ points, the total cost to compute the isogenous theta null point is $(2^g - 1)(2.2^g S + 4.2^g M + 3M + 1I)$.

Note that here we don't take into account the cost of computing the squares of the theta null point, this was already done for the doubling computations. For the image of points we need to invert the (dual) coordinates of the isogenous theta null point, this costs $2^g I$.

We also need to compute $2^g S + 2^g I$ for the doubling operations on B, for computing the inverse of the square of the isogenous theta coordinates (it might seem that we would need $2^g I$ more to compute the inverse of the theta constants of B, but we already have the inverse of the dual coordinates, so we can compute the doubling in dual coordinates, and we just need the $1/\theta_i^A(0)^2$).

So the total cost to compute the theta null point and all inverse needed for images and doublings is $(2^g - 1)(2.2^g S + 4.2^g M + 3M + 1I) + 2^g S + 2.2^g I = 2^g(6.2^g + 1) - 4$ arithmetic operations, including $3.2^g - 1$ inversions.

With these new estimates, the above tables become:

2^n	$g = 1$	$g = 2$	$g = 4$	$g = 8$
2^{128}	8028	43560	823584	220483584
2^{216}	14476	79080	1501248	402380448
2^{250}	17060	93360	1774200	475685400
2^{305}	21350	117120	2228940	597857340
2^{372}	26576	146064	2782896	746684976
2^{486}	35904	197928	3777768	1014145128

g	Naive ratios	Estimated ratios
2	×4	×5.5
4	×64	×105
8	×16384	×28000

We will use these operations count to compare the new algorithm with the old one.

14.2. **The new algorithm.** To compute a 2^n-isogeny in dimension g, we need to keep track of the theta null points and of a basis T''_i. Then we only compute image of points, which cost roughly $2^g(S + M)$ by point, doublings, which cost roughly the same as 2 images, and the isogeneous theta null point, which cost roughly $1 + g$ images, g normalisations (which is roughly one doubling + one differential additino) along with some inverse to speed up the upcoming computations.

A rough estimate of the complexity to compute the isogenous theta constants is then around $\kappa(1 + g)2^g$, where κ will not depend too much on the dimension: we have $(1 + g)$ points to push and each point is represented by 2^g-coordinates. But for 2^n-isogenies we need to keep track of the basis, the optimal strategy uses a dynamic strategy due to [DJP14] optimising the number of doublings vs images according to their cost. This part of the algorithm can be estimated as $\kappa_2 g 2^g$: g points for the basis with 2^g coordinates each. In practice this part is dominating, it is roughly twice as expensive as the theta constant phase (see below for more precise ratios). So we will use this to estimate our complexity ratio as our dimension increase.

A more refined estimate relies on using the optimal algorithm to choose between doubling points and pushing them by the isogeny.

14.3. **The new algorithm: normalising 8-torsion points at each steps.** Let us first begin with the case where we normalize at each step like the previous algorithm, rather than once at the beginning. This allows for a better comparison with the old algorithm, the difference being that we need to keep track of only g points (along with the theta null point), rather than $2^g - 1$.

(1) Doubling a point costs $2.2^g S + (2.2^g - 1)M = 4.2^g - 1$ operation by points and computing the image of a point $2^g S + (2^g - 1)M = 2.2^g - 1$ operations (without any inversion). The difference with the complexity of the old algorithm is that in this case we naturally compute the isogenous theta null point with the first coordinate normalised to 1.

(2) For each isogeny, we normalize points of 8 torsion (a basis of $K[8]$), we already saw above that this costs $2.2^g S + 4.2^g M + 1M + 1I = 6.2^g + 2$ by point (we gain $1M$ in our doubling because of the normalised theta constant).

We compute $1/A_0^2$ in $1I$, recover the $A_i/A_0, A_0/A_i$ for i of Hamming weight one in $1M + 1I$, and then the $A_i/A_0, A_0/A_i$ for the other i in $2M + 1I$ (write $A_i/A_0 = A_iA_j \times A_0/A_j \times 1/A_0^2$). Also each of these constants need to be normalised by the appropriate projective factor, this costs $(2^g - 1)M$. The final cost is $1I + g(1M + 1I) + (2^g - g - 1)(2M + 1I) + (2^g - 1)M = 4.2^g - g - 3$.

For our doubling on the isogenous variety, we need to precompute some constants, for a total of $2^g S + 2^g I$.

The total cost to compute the isogenous theta null point along with all the inverses needed for the doublings and images is thus of $g(6.2^g + 2) + 4.2^g - g - 3 + 2.2^g = 6(g+1)2^g + g - 3$, including 2.2^g inversions.

We can plug these costs into the dynamic algorithm optimising the number of doublings vs images for a 2^n-isogeny. This gives us the following estimation of the number of arithmetic operations for computing 2^n-isogenies in different dimensions, we can also estimate the ratios and compare them with the naive expected ratios, and more importantly look at the efficiency gain compared to the previous algorithm.

2^n	$g = 1$	$g = 2$	$g = 4$	$g = 8$
2^{128}	7076	28032	224544	7099584
2^{216}	12704	50688	407976	12930264
2^{250}	14953	59776	481742	15277834
2^{305}	18663	74841	604437	19187709
2^{372}	23254	93340	754196	23951236
2^{486}	31275	126096	1022034	32500950

g	Naive ratios	Estimated ratios	Gain
1			×0.87
2	×4	×4	×0.64
4	×32	×32	×0.27
8	×1024	×1024	×0.032

If we look at the proportion of operations needed to compute the isogenous theta constants (along with all the constants needed for doubling and images), we see that depending on the isogeny size, this proportion is 35–40% for $g = 1$, 27–32% for $g = 2$, 22–27% for $g = 4$ and 20–25% for $g = 8$ (the longer the isogeny, the less the proportion).

We remark that once the chain of 2-isogenous theta null point is computed (along with the associated constants), we can compute the image of any point by our big 2^n-isogeny f or its dual in $n \times (2^g S + 2^g M)$ (we gain one M by image if our constants have been normalised to have one dual theta coordinate equal to 1).

14.4. **The new algorithm, normalizing points at the beginning.** Now we look at the complexity of the new algorithm, where we normalize points at the beginning and use affine images and doublings at every step. We treat here the case of a general kernel, already compatible with the theta null point.

(1) Normalising the point T_i'' costs a scalar multiplication to compute $(2^{n+1}-1)T_i'', 2^{n+1}T_i''$, and one more differential addition to compute $(2^{n+1}+1)T_i''$. The scalar multiplication costs 7.2^g arithmetic operations by bits (in the general case, it is slightly faster if we normalise one of the theta null coordinate to be 1). We need to compute the inverse of the theta coordinates of T_i'' first (for $2^g I$), and then we only need multiplications and squares afterwards. The extra differential addition costs 4.2^g arithmetic operations, but since this is only used to get the normalisation factor, as explained above, we actually only need $2^g S + 2^g M + 2M + 1I$.

    Since we have g points to normalize, the normalisation phase costs $g \times ((n+1).7.2^g + 3.2^g + 3)$, including $g(2^g+1)$ inversions.

(2) Keeping track of the normalisation factor. Recall that we have an equation $\lambda_1^{2^{n+2}} = C$ for our each of our normalisation factor. To compute the points of 4-torsion from the T_i'', we need to adjust our points by the normalisation factor $\lambda_1' = \lambda_1^{2^{2n}}$. For computing the theta null point, we only need ${\lambda_1'}^2 = \lambda_1^{2^{2n+1}} = C^{2^{n-1}}$.

    When we compute the image of T_i'', the new normalisation factor is $\lambda_2 = \lambda_1^2$. We have $\lambda_2^{2^{n+1}} = \lambda_1^{2^{n+2}} = C$. so for the second isogeny, the normalisation factor on the points of 4-torsion is then $C^{2^{n-2}}$ and so on, until at the last step we use C. In total we need $n-1$ squares to compute the actual constants which will give us our normalisation factors for each of our isogenies. Since we have g normalisation factors, this adds $g(n-1)S = g(n-1)$ arithmetic operations.

(3) Doubling a point costs $4.2^g - 1$ operation by points (we are in a situation where one of the theta constant is normalised to 1).

(4) Computing the image of a point costs $2.2^g - 1$ operations.

(5) Computing an isogenous theta null point require computing the squares of the domain theta null point and the g points of 4-torsion forming a basis of $K[4]$; this costs $(1+g)2^g$ squares.

    We need to compute the A_i/A_0 (for the isogenous theta null point), $1/A_0A_i$ (for the isogeny images, unlike the previous algorithm where we used A_0/A_i here we need to use the affine isogeny formula, so correct by the renormalisation we used for the theta null point), and the A_0/A_i (for doubling in the dual theta coordinates).

    We can compute them as follow: first compute $1/A_0^2$, then compute $A_i/A_0 = A_iA_j/A_jA_0$, $A_0/A_i = 1/(A_i/A_0)$, $1/A_0A_i = A_0/A_i \times 1/A_0^2$ for a total cost of $1I + 2M$ by coefficient. We need to add $1M$ to take into account the normalisation factor. Thus computing the isogenous theta null points costs $1I + (2^g - 1)(1I + 3M) = 4.2^g - 3$ operations.

For doubling (in dual theta coordinates) on the isogenous abelian variety, we need the coefficients A_0^2/a_i^2 (because we need to do affine doublings and we renormalised our theta null point), this costs $2^g I + 2^g M$.

The grand total to compute the theta null points and all inverse needed for images and doublings is $(1+g)2^g + 4.2^g - 3 + 2.2^g = 2^g(g+7) - 3$, including including $2.2^g I$.

In summary, the amortised cost for computing an isogenous theta null point (taking into account the normalisation at the beginning) is of $g.7.2^g + 2^g(g+7) - 3 = 2^g(7g+g+7) - 3$ arithmetic operations, including $2.2^g I$.

The amortised cost for the normalisation is roughly a doubling and differential addition for each of the g points in our basis. So this is about the same cost as we obtain by normalising the points of 8-torsion anew for each isogeny, except that in the latter case we don't need a full differential addition, only a partial one, and we can compute doublings and images projectively since we don't need to keep track of our normalisation factors because we recompute them at each step.

Thats why, normalising at each steps gives better complexity. But note that for cryptographic applications where $A = E_1^g \times E_2^g$, we could just normalize a basis of $E_i[N]$, and then switch to affine differential additions when computing $\operatorname{Ker} F$ to keep points normalised. This allows to normalize only 4 points in dimension 1, instead of $2g$ points in dimension $2g$ (which amount roughly to normalizing $4g^2$ points in dimension 1). This gains a factor roughly 4 when $g = 2$, i.e., $\dim A = 4$ (roughly because once the points are normalised, to compute the kernel of F we need to use affine differential additions rather than projective ones, this will cost $1M$ more). So we expect that for the cryptographic setting of 2^n-isogenies in dimension 4, the method of normalising points globally will be more effective, because we will be able to do the normalisation in dimension 1.

The estimated number of operations is summarised in the following table, note that here these operations count do not assume that the initial variety is a product, so we compute the normalisation in dimension g.

2^n	$g=1$	$g=2$	$g=4$	$g=8$
2^{128}	7866	30676	243624	7677136
2^{216}	14022	55092	439728	13890792
2^{250}	16475	64860	518390	16386330
2^{305}	20515	81025	649005	20535565
2^{372}	26576	146064	2782896	746684976
2^{486}	35904	197928	3777768	1014145128

g	Naive ratios	Estimated ratios	Gain
1			×0.95
2	×3	×4	×0.7
4	×20	×32	×0.29
8	×576	×1000	×0.034

15. 2^n-ISOGENIES IN DIMENSION 1

15.1. **2-isogenies in the theta model.** The above algorithm is generic and work in any dimension; the resulting number of operations in dimension 1 simply amount to plugging $g = 1$ in the formula.

However in practice computing 2^n-isogenies in the theta model in dimension 1 is faster than the generic algorithm, because we can dispense with point normalisation in dimension one. (Update: see Section 17 for the same tricks in higher dimension.)

The algorithm is thus as follow: let $0_E = (a : b)$ be the theta null point, given by a theta structure such that our small kernel is generated by $T = (-a : b)$.

We assume that we have a point of 8-torsion T'' above T, if $n \geq 3$ we want $T'' \in K$, and if $n = 2$ we want $2T''$ in K, this ensure that if our isogenous theta null point on E_2 is $(a_2 : b_2)$, our next kernel will be generated by the point $(-a_2 : b_2)$.

The formula are sufficiently simple that we will also keep track of the additions/soustractions.

Let $T'' = (r : s)$, we have by Example 11.2 $(A : B) = (r^2 + s^2 : r^2 - s^2)$, and $(a_2 : b_2) = (A + B : A - B) = (r^2 : s^2)$. This requires $2S + 2a$.

The image of a point $P = (x : z)$ is given by $(x : z) \mapsto (x^2 : z^2) \mapsto (X = x^2 + z^2 : Z = x^2 - z^2) \mapsto (BX : AZ) \mapsto (x' = BX + AZ : z' = BX - AZ)$, and is computed in $2S + 2M + 4a$.

Doubling on E_2 cost two images, the first one for the dual isogeny $\tilde{f}$ using the coefficients $(a : b)$ instead of $(A : B)$ in the formula above, and the second using f to go back to E_2, for a total cost of $4S + 4M + 8a$.

If we have many doublings and images to compute, it might be worth to compute $(1, B/A)$. This can be done with one division, that is $1I + 1M$. We then gain $1M$ for images and doubling. At this point we might as well compute also $(1, b/a)$. We can compute both $1/a, 1/A$ in $1I + 3M$, so compute $(1, b/a), (1, B/A)$ in $1I + 5M$. We then gain $2M$ for images and doubling. In summary, adding one inversion, computing the normalised theta null points and associated constant costs $1I + 5M + 2S + 2a$ (instead of $2S + 2a$), and the computing an image costs $2S + 1M + 4a$ and a doubling $4S + 2M + 8a$ (instead of $2S + 2M + 4a$ and $4S + 4M + 8a$ respectively).

We obtain the following costs in dimension 1 (which give roughly a twenty percent speedup compared to the generic algorithm, not counting the fact that here the arithmetic operations are without any inversion).

2^n	$g = 1$
2^{128}	5468
2^{216}	10156
2^{250}	12060
2^{305}	15250
2^{372}	19136
2^{486}	26184

If we don't have an available point of 8-torsion T'', we simply compute $A^2 = a^2 + b^2, B^2 = a^2 - b^2$ and take an arbitrary square root of B^2/A^2. What we can do also, without requiring a square root, is to compute the codomain in the Montgomery model, see the next section.

15.2. **Theta versus Montgomery.** To summarize, the complexities for computing isogenies in the theta model are as follows:

(1) $2S + 2a$ for the codomain
(2) $2S + 2M + 4a$ for an image
(3) $4S + 4M + 8a$ for doubling

The input is the theta null point $(a : b)$, which implicitly contains the 2-torsion point $(-a : b)$ used for our kernel; and the images computations needs (some constants computed during) the codomain. We refer to [Rob23b] for similar formulas on twisted theta models.

In the Montgomery model, the costs are, using [CLN16; CH17; Ren18]:

(1) $2S + 1a$ for the codomain
(2) $4M + 4a$ for an image (using a precomputation of $2a$)
(3) $2S + 4M + 4a$ for doubling

Here the input is a two torsion point (different from $(0 : 1)$) giving the kernel (and implicitly the curve); the image computation does not needs the codomain.

In [RS24] (see the notes [Rob23b] for more on the arithmetic of Kummer lines), we explain how to combine the best of both worlds. Provided we have a point of 4-torsion T above our kernel $\langle T_1 \rangle$, we can:

(1) Compute a representation of the codomain in $2S$. The representation is given by the 2-torsion point $f(T) = T_2$, which is the kernel of the next isogeny.
    If we need to compute doublings on the codomain, we need to add a $2S + 2a$ precomputation to compute $(\mathcal{A} + 2 : 4)$, and if we need to compute images we need to add a $2a$ precomputation (which is already done if we did the previous $2S + 2a$ precomputation needed for doublings).
(2) Compute "images" in $2M + 2S + 4a$.
(3) Compute "doublings" in $4M + 2S + 4a$.

The words "images" and "doublings" are in quotes because if we consider that we are on a twisted theta models the "doublings" we compute are actually $2P + T_1$, while if we consider that we are in the Montgomery model it is the images that are actually given by $f(P) + T_2$. The images need some of the constants computed for the codomain.

As an aside, we can also explain how to compute the isogeny from a theta model to a Montgomery model if we do not have access to a 8-torsion point. From the theta null point $(a : b)$ of E_1, we can compute $(a^2 : b^2)$ the theta null point of E_2 in the $\theta_{E_1}^2 = \theta' tw'_{E_2}$ model, and the isogeny map is $(x : z) \mapsto (x^2 : z^2)$. Translating by $T_2 = (1 : 0)$ we obtain the coordinates on the Montgomery model of E_2, with $\mathcal{A}_2 = -\alpha_2 - 1/\alpha_2$, $\alpha_2 = b^2/a^2$.

We conclude this with a discussion on 4-isogenies. On the Montgomery model, a 4-isogeny can be computed in [CH17]:

(1) $4S + 5a$ for the codomain
(2) $6M + 2S + 6a$ for images.

Using these formula, it is faster to split a 2^n-isogeny in dimension 1 into blocks of 4-isogenies rather than blocks of 2-isogenies.

We leave as an open question the task of generalising these efficient 4-isogenies formula to the theta model in dimension 1 (or even better in higher dimension).

16. 2^n-ISOGENIES IN DIMENSION 2

16.1. **Isogeny formula.** The estimation above are very rough because we count an inversion as much as a square or a multiplication. In this section we detail the detail the case of $g = 2$, and we try to use Montgomery's trick as much as possible to reduce the number of inversions needed by isogeny. Recall that this trick replace m parallel inversions by $1I + 3(m - 1)M$.

For $g = 2$ we normalize our basis (P_1, P_2) of 8-torsion of $K[8]$ by one doubling, which cost (by point) $2.2^g S + (2.2^g - 1)M = 8S + 7M$, and then a partial differential addition which costs $2^g S + 2^g M + 2M + 1I = 4S + 6M + 1I$. Since we have two points, we can replace $2I$ by $1I + 3M$. The total cost is then $1I + 2.(8S + 7M + 4S + 6M) + 3M = 1I + 24S + 29M$.

These operations already give us the squares of the coordinates of the points of 4-torsion $2P_1, 2P_2$, and if we add the squares of the theta constants, we obtain by Example 11.3 (A^2, B^2, C^2, D^2), (AB, AB, CD, CD), (AC, BD, AC, BD) (up to the projective factors computed above) via $2^g S = 4S$. We need to add $3M$ to take into account the correcting factors for the coefficients AB, AC, BD, so the total cost is $4S + 3M$. We can exploit the fact that we work with projective coordinates to dispense with the $1I$ in the computation of the normalisation factor. If we don't compute this inversion, what we obtain are the points $(A^2, B^2, C^2, D^2, \kappa AB, \kappa CD, \kappa AC, \kappa BD)$ where κ is the element we did not inverse (which is the product of all elements we needed to inverse in parallel and which is computed as part of Montgomery's trick). So via $4M$, we can recover the projective vector $(A^2 : B^2 : C^2, D^2 : AB : AC : BD)$, we actually won't need C^2 so we just need $3M$. The final cost for this vector, in order to gain our $1I$, is $4S + 6M$.

We want to compute the isogeneous dual theta null point $(1 : B/A : C/A : D/A)$, and also for the image of the points the constants $(1 : A/B : A/C, A/D)$. We compute $1/A^2, 1/AB, 1/AC, 1/BD$. We recover $B/A = AB * 1/A^2, A/B = 1/AB * A^2, C/A = AC * 1/A^2, A/C = 1/AC * A^2, D/A = 1/BD * B/A * D^2, A/D = 1/BD * A/B * B^2$ in $4I + 8M = 1I + 17M$. In fact, using the same trick as above, we can entirely dispense with the inversion. If we do we obtain the coordinates $\kappa' B/A, \kappa' A/B, \kappa' C/A, \kappa' A/C, \kappa'^2 D/A \kappa'^2 A/D$ where κ' is the product of all coordinates we inverted. To recover the projective vectors $(A : B : C : D)$ and $(1/A : 1/B : 1/C : 1/D)$ we thus need to compute κ'^2 and do $4M$. This adds $1S + 4M$, for a cost of $1S + 21M$.

For the doubling on the isogenous abelian variety, we need to inverse 4 coordinates (the 4 squares of the theta constants needed are already taken into account above), for $4I$, i.e., $1I + 9M$. In this case, the inversion is not needed, since the projective factor will be the same.

However, while this is ok for projective doubling, for the affine doubling and differential addition we need when computing $3.T_i''$ for the normalisation, we will be off by some projective factors. Namely, since I compute the vector $(1/A : 1/B : 1/C : 1/D)$ up to some factor κ_1, and the vector $(1/a^2 : 1/b^2 : 1/c^2 : 1/d^2)$ up to some projective factor κ_2, the first doubling is off by a factor $\kappa_1\kappa_2$, and then the differential addition is off by a factor $(\kappa_1\kappa_2)^3$. This constant is computed via $2M + 1S$, and we need to use it for our 2 normalisation which adds $2M$.

The final cost is $(24S + 29M) + (4S + 6M) + (1S + 21M) + (9M) + (1S + 4M) = 30S + 69M \leq 99M$. In this case, the (dual) theta null point (A, B, C, D) is not normalised to have $A = 1$, so the image of a point then costs $4S + 4M$, and doubling costs $8S + 8M$.

If we have many doublings and images to compute, it might be interesting to add back $1I$ to normalise our coefficient A to be $A = 1$, and while we are at it $a = 1$. The image of a point then costs $4S + 3M$, and (projective) doubling costs $8S + 6M$.

We obtain the following number of arithmetic operations for our isogenies, without any inversion. We see that replacing all inversions by multiplication roughly augment the arithmetic count by twenty percent compared to the previous table, which is mainly due to the fact that our images are 15% slower ($4S + 4M$ vs $4S + 3M$) and our doublings 5% slower ($8S + 8M$ vs $8S + 7M$); the remaining cost being due to the fact that the isogenous theta constant and the associated constants needed for images and doublings take more arithmetic operations when we remove all inversions. The proportion of operations related to computing the isogenous theta null points (and associated constants for doubling and images) compared to doublings and images is between 32–37%.

2^n	$g = 2$
2^{128}	33520
2^{216}	60280
2^{250}	70990
2^{305}	88755
2^{372}	110396
2^{486}	148962

Remark 16.1. In our estimation of roughly $100M$ to compute the theta null point, half of it ($24S + 29M$) is spent normalizing our two points of 8-torsion (P_1, P_2). The normalisation computes (affinely) $2P_i, 3P_i$. At the next isogeny step, say we have for 8-torsion points on B the points (P'_1, P'_2). Then we have $2P'_i = f(P_i)$ projectively. So we can replace a doubling by an image (which is twice as fast, and in fact when computing $3P_i$ we essentially compute $f(P_i)$ along the way), and we just need one affine coordinate of $2P'_i$ to correct $f(P_i)$ to obtain the correct affine lift of $2P'_i$.

In other words, we can reuse part of the work of normalising our 8-torsion points on A to speed up normalising our 8-torsion points on B.

16.2. **Splitting isogenies.** In the contest of cryptography, the last 2-isogeny will be a splitting $A \to E_1 \times E_2$. During the isogeny computation, we will not in general obtain a product theta structure on $E_1 \times E_2$. In [DLRW23, Appendix C.1] we explain how, if we have enough information, we can precompute (by working in dimension 1) the linear change of variable giving a product theta structure.

But in dimension 2 it is easy to obtain it directly. First we know that we are on a product when one of the 10 even level $(2, 2)$-theta constant is zero, and we know that we have a product theta structure where the zero theta constant is $\theta[11; 11]$.

We might as well take a random linear change of variable induced by a symplectic action until we are on this case. A more deterministic algorithm (using Appendix B to stay in level 2) is as follows:

(1) The square of the level $(2, 2)$ theta functions can be computed from the level 2 theta function via (this is a special case of the duplication formula) $U^2_{\chi,i} = \sum_t \chi(t)\theta_t\theta_{i+t}$.
 Suppose that we have a theta null point $(a : b : c : d)$ on a product. Let (χ, i) be the coordinate of the even theta constant which is zero.
(2) if $\chi = i = (00)$, act by $(a : b : c : d) \mapsto (a : ib : c : id)$, the new zero level $(2, 2)$-theta function is given by (χ', i') with $\chi' \neq 0, i' = 0$.
(3) if $i = 0$ but $\chi \neq 0$ uses the action by H, this permutes χ and i.
(4) We can now assume $i \neq 0$. Take any invertible matrix A such that $A(11) = i$. Acting by $\theta_i \mapsto \theta_{Ai}$ we get that the new zero theta function is (χ', i') with $\chi' = \chi o A$, and $i' = (11)$.
(5) We can now assume $i = (11)$. Act by $\theta_i \mapsto (-1)^{(1-\chi)(i)}\theta_i$. The new zero theta function is given by (chi', i')=(1 1, 1 1) and we have won.
(6) If we have a point $(x : y : z : t)$ on the product theta structure, the theta coordinates on E_1, E_2 are given by $(x : z), (x : y)$.

16.3. **Gluing isogenies.** When we start with a product of two elliptic curves $E_1 \times E_2$ and a product theta structure $(a : b : c : d)$, then since the dual theta coordinates on the isogeneous surface $(A : B : C : D)$ can also be interpreted as the level $(2, 2)$-theta coordinates on the original surface given by $U_{\chi,0}(0)$, they are not zero (because the one which is zero corresponds to $\chi = i = (11)$).

However, if we take an isogeny which is not a diagonal isogeny, we will do a linear change of variable as explained in Sections 8 and 9 and one of the A, B, C, D will become zero.

For the arithmetic on $E_1 \times E_2$ this is not a problem (we would compute the arithmetic in dimension 1 before taking the product theta structure and doing the linear change of variable anyway), but this is a problem for the images of a point. The solution is given in Section 12: to compute $f(P)$, we need one of $P + T_1$, $P + T_2$ for T_1, T_2 the 2-points of 4-torsion compatible with our isogeny.

We note that if we only have P and T_1, there are four choices for $P + T_1$ on the Kummer. This corresponds to the fact that the map $E_1 \times E_2 \to E_1/\pm 1 \times E_2/\pm 1$ has degree 4, and given a point $[P] \in E_1/\pm 1 \times E_2/\pm 1$ we have 4 possibilities for P on $E_1 \times E_2$, which induces 4 possibilities on the codomain B, which induces 2 possibilities on $B/\pm 1$. So $[P]$ has two possible images on $B/\pm 1$, and we need a point of 4 torsion to fix one.

If our big kernel K is generated by P_1, P_2 of 2^n-torsion, we can take $T_i = 2^{n-2}P_i$. To compute our isogenies we need $f(P_1), f(P_2)$. But $P_i + T_i = (1 + 2^{n-2})P_i$ which can be computed via a scalar multiplication.

More concretely write $P_i = (R_i, S_i)$, then the four possibilities for $P_i + T_i$ can be written as $(1 \pm 2^{n-2}R_i, 1 \pm 2^{n-2}S_i)$. If we make some choice of sign for P_1 (say $(+,+)$) it is important to make the same for P_2 (say $(+,+)$ or $(-,-)$ but not $(+,-)$ or $(-,+)$) for our images of P_1, P_2 to be compatible. (The four choices for $P_1 + T_1$ corresponds to replacing P_1 by $-P_1$ or f by $-f$ in Kani's lemma. It might seem that we would need to fix $T_1 + T_2$ in order to fix the sign of P_1 relatively to P_2, but this is already done, at least implicitly, in our linear change of variable from our product theta structure: for this theta structure the basis of 4-torsion is of the form $(U_1, 0), (0, U_2), (V_1, 0), (0, V_2)$ which are points that only admits 2 preimages on $E_1 \times E_2$).

16.4. **Annulation of the theta null points.**

Analytically, if A corresponds to the period matrix Ω, we have $(a, b, c, d) = \theta[0, i/2](0, \Omega/2)$ and $(A, B, C, D) = \theta[i/2, 0](0, \Omega)$.

The 10 level four even theta constants are $\theta[i/2, j/2](0, \Omega)$ are non zero, except when A is a product where exactly one of them is zero. And if A has a product theta structure, the zero even theta constant is $\theta[1/21/2; 1/21/2](0, \Omega)$.

From this we deduce that:

- (A, B, C, D) are non zero, except if A is a product with a non product theta structure.
- (a, b, c, d) are non zero, except if the isogenous abelian variety A/K_2 corresponding to $\Omega/2$ is a product with a non product theta structure.

So unless we encounter a product along our path (very unlikely), the only annulation we will see is at the first and last isogeny.

16.5. **Further optimisations in dimension 2.**

Due to the ongoing work on implementing the formula in dimension 2, it is now easier to find new optimisation possibilities.

The image of a point is pretty fast, so the remaining bottleneck is to try to compute the theta constants as fast as possible.

There are two optimisations: first the normalisation procedure, a lot of the computations can be shared. Secondly, as remarked by Pierrick Dartois, the points of 4-torsion we deal with have 2 zero coordinates, so this simplify the computations.

First let's explain look at the points of 4-torsion: we have $T' + T = T'$ in the Kummer, where $T = 2T' \in K_2$. Since T acts by sign, this equation gives that half of the coordinates are zero (we can even know which ones should be zero since we require the compatibility with the theta structure).

Secondly, let's look at the normalisation procedure. We have T'' a point of 8-torsion, and we compute $3T''$ to compute the correct projective factor λ. Note that we only need to apply this factor to $\theta'(f(T'))$.

Now from Section 5, our first (affine) doubling $T' = 2T''$ can be written as $\theta_i(T')\theta_i(0) = H(\theta'_i(T'')^2)$. As explained in Section 16.1, a doubling is $8M + 8S$ (if the appropriate inverses have been computed), but since we have two zero coordiantes the cost reduces to $6M + 8S$.

The main gain we can have is for the differential addition $3T'' = T'' + T'$, remember that $\theta_i(3T'') = \theta_i(5T'') = \theta_i(T'' + T)$ hence is equal to $\theta_i(T'')$ up to an explicit sign.

In particular, by the duplication formula, we have $\theta'_i(f(T''))\theta'_i(f(T')) = H(\theta_i(3T'')\theta_i(T''))$. Now we have already mostly computed $\theta'_i(f(T''))$ and $\theta_i(T'')^2$ during the doubling. To compute $\theta'_i(f(T'))$ usually requires $4S + 4M$, but the multiplication by the required constants can already be done during the doubling of T'', and the $4S$ is a $2S$ because two coordinates are zero. However this changes the ordering of operations for the doubling, which now costs $10M + 4S$ rather than $6M + 8S$.

So $\theta'_0(f(T'))$ requires (essentially) $2S$, and $\theta'_0(f(T''))\theta'_0(f(T'))$ adds $1M$. The correcting factor is then one division D, which we use to multiply two coordinates (because T' give half of the coordinates we are interested in), for a cost of $2M$. And in fact for the second generator, we just need one coefficient of $f(T')$ so we just add $1M$ for the correction.

In total, we have spent $(10M + 4S) + (2S + 1M) + 1D + 2M = 13M + 6S + 1D$ to get the correct affine value of $f(T')$. Doing this twice (once for each projective factor), we get (AB, CD, AC) with a cost of $25M + 12S + 2D$. Since we already know the value of (A^2, B^2, C^2, D^2) (since they were used for doubling; if we count them as precomputed then we need to add the computation of $(A_2^2 : B_2^2 : C_2^2 : D_2^2)$ as required precomputations for our theta null point), we recover as in Section 16.1 the values $(A : B : C : D)$, and then $(a_2 : b_2 : c_2 : d_2)$.

By contrast, the method outlined in Section 16.1 was costing $32M + 24S + 2I$ for the same result. We gain about 19 arithmetic operations.

We also need for the images $(1/A : 1/B : 1/C : 1/D)$ (which can be done in $4M$ because we already have $(1/A^2 : 1/B^2 : 1/C^2 : 1/D^2)$ and for doublings on the isogenous curve $(a_2^2 : b_2^2 : c_2^2 : d_2^2)$ to compute $(A_2^2 : B_2^2 : C_2^2 : D_2^2)$, $(1/A_2^2 : 1/B_2^2 : 1/C_2^2 : 1/D_2^2)$ and $(1/a_2 : 1/b_2 : 1/c_2 : 1/d_2)$. This requires $4S + 4M + 8I$.

A trick is to instead do doubling in θ' coordinates; for that we need $(1/A : 1/B : 1/C : 1/D)$ which we already have for images, and $(1/a^2 : 1/b^2 : 1/c^2 : 1/d^2)$. So from this point of view, anticipating the next isogeny, we need $(a_2^2 : b_2^2 : c_2^2 : d_2^2)$ and their inverse, so this does not change much the number of operations: $4S + 8I$, so we save $4M$.

For this part, we refer to Section 16.1; we can apply the same various M/I tradeoffs to get rid of all inversions at the cost of more multiplication. Remark 16.1 also apply, at the next step we could compute an isogeny image rather to speed up the doubling procedure.

16.6. **What if we don't have 8-torsion points?** If we only have points of 4-torsion T'_1, T'_2 above our kernel K_2, applying $h := H \circ S$ to them gives $(AB : AB : CD : CD)$, $(AC : AC : BD : BD)$. We also have $(A^2 : B^2 : C^2 : D^2)$ from the theta null point.

We cannot recover the theta null point $(A : B : C : D)$ directly because we are in projective coordinates. We can normalize T'_i via the equation $2.T'_i = T_i$, this determines the normalisation factor λ_i up to an equation $\lambda_i^4 = C_i$, hence this we square the coordinates, $h(T'_i)$ up to a sign. Hence we have 2 signs; and by the same method in dimension g we would have g signs, which are all valid by Appendix B.2. In particular, in dimension 2 we need two square roots to compute the codomain theta null point when we only have points of 4-torsion and not of 8-torsion. In fact, we can rewrite the normalisation process as follow: let $h(T_1') = (\lambda_1 x, \lambda_1 x, \lambda_1 y, \lambda_1 y)$ for some unknown projective factor λ_1. Fix a choice of (A^2, B^2, C^2, D^2). Then for the correct choice of λ_1, we should have $\lambda_1 x = AB, \lambda_1 y = CD$, and we have an equation $\lambda_1^2 x^2 = A^2B^2$, which gives λ_1 from a square root computation. The same method works for λ_2. If $(A : B : C : D)$ is one of the computed isogeneous theta null point, the other choice of signs give $(A : -B : C : -D), (A : B : -C : -D), (A : -B : -C : D)$.

If we don't even have the compatible points of 4-torsion above the kernel, we can write down equations which determines T_1', T_2', or even just $h(T_1'), h(T_2')$ which is what we really need for the codomain. Since T_1' is a compatible point of 4-torsion, and $T_1' + T_1 = T_1'$, we have $T_1' = (x : 0 : z : 0)$. From $h(T_1') = (AB : AB : CD : CD)$ we obtain a degree 2 homogeneous equation in x^2, z^2 (say $h(T_1') = (x_1 : x_1 : z_1 : z_1)$ with x_1, z_1 homogeneous of degree 2 in x, z, then $C^2D^2x_1^2 - A^2B^2z_1^2$), so we have two solutions for $h(T_1')$ (which is linear in x^2, z^2). Now $T_2' = (u : v : 0 : 0)$, and write $h(T_2') = (x_2 : z_2 : x_2 : z_2)$. We have $1/A^2x_1x_2 - 1/D^2z_1z_2$. This equation determines the projective point $h(T_2')$ uniquely from x^2, z^2. To our choice of signs above, this adds the possibility $(A : B : C : -D)$.

17. Even better formula: getting rid of the normalisation process

We have seen that in dimension 1 (Example 11.2) we don't need to normalize points of 8 and 4-torsion. So why do we need to normalize points in higher dimension? The answer is that we actually don't need this, which leads to both faster formula and much easier implementations.

The basic idea is as follow. Let's work in dimension $g = 2$ for simplicity. Let P_1, P_2 be a basis of $K[8]$, then we know that applying $S \to H \to C$ gives the images $f(P_1), f(P_2)$ in θ' coordinates.

Our kernels are set up so that $K[2] = K_2$, in particular $4P_1, 4P_2$ acts by sign. So in θ' coordinates, $2f(P_i)$ acts by permutation. $f(P_1), f(P_2)$ are points of 4-torsions, and since we are on the Kummer, we have $f(P_i) + 2f(P_i) = f(P_i)$.

Recall that $\theta'(f(P)) = g(P) := C \circ H \circ S$. This means that $g(P_1) = (x_1 : x_1 : z_1 : z_1)$ and $g(P_2) = (x_2 : z_2 : x_2 : z_2)$. Going one step back in the isogeny image formula, it means that if we apply $h := S \to H$ to P_1, P_2 (remember that $C = (1/A : 1/B : 1/C : 1/D)$ is unknown for now), we have $h(P_1) = (Ax_1 : Bx_1 : Cz_1 : Dz_1)$ and $h(P_2) = (Ax_2 : Bz_2 : Cx_2 : Dz_2)$ where x_1, z_1, x_2, z_2 are unknown projective factors.

But from this we can recover B/A, C/A and $D/A = D/C \times C/A$ in only $2\times 4S + 1M + 3D$. Actually, for isogeny images we need $(1/A : 1/B : 1/C : 1/D)$, and we can compute $(1, A/B, A/C, A/D)$ in $2 \times 4S + 1M + 3D$. The nice thing is that a constant is normalised to 1, so images only cost $4S + 3M$. Then doublings could be implemented as composing $\hat{f}$ with f. For that we need $(1/a : 1/b : 1/c : 1/d)$, or better $(1, a/b, a/c, a/d)$; each doubling would cost $8S + 6M$.

As for doubling precomputations, for the next isogeny we would need to compute $(1/a_2 : 1/b_2 : 1/c_2 : 1/d_2)$ (or better $(1, a_2/b_2, a_2/c_2, a_2/d_2)$) from $(1, A/B, A/C, A/D)$. This can clearly be done through $3I + 3D$, but maybe there are some optimisations to be gained.

In higher dimension, the same strategy as in Section 11 holds. Let's work out the case $g = 3$, we have for P_1, P_2, P_3 a basis of $K[8]$, $g(P_1) = (x_1A_{000}, x_1A_{001}, y_1A_{010}, y_1A_{011}, z_1A_{100}, z_1A_{101}, t_1A_{110}, t_1A_{111})$, $g(P_2) = (x_2A_{000}, y_2A_{001}, x_2A_{010}, y_2A_{011}, z_2A_{100}, t_2A_{101}, z_2A_{110}, t_2A_{111})$, $g(P_3) = (x_3A_{000}, y_3A_{001}, z_3A_{010}, t_3A_{011}, x_3A_{100}, y_3A_{101}, z_3A_{110}, t_3A_{111})$.

So from the $g(P_i)$, which we can compute, we can recover the quotients $A_{001}/A_{000}, A_{010}/A_{000}, A_{100}/A_{000}$, and then iteratively $A_{011}/A_{000} = A_{011}/A_{010} \times A_{010}/A_{000}, A_{110}/A_{000} = A_{110}/A_{100} \times A_{100}/A_{000}, A_{111}/A_{000} = A_{111}/A_{110} \times A_{110}/A_{000}$.

In dimension g, computing the $g(P_i)$ costs $g \times 2^g S$, and then reconstituting the A_i/A_0 for isogeny images costs at most $2^g \times (1M + 1D)$, to which we need to add $2.2^g I$ for the arithmetic precomputations. So the total cost for the codomain, including the arithmetic precomputation, is $2^g(g+4)$ arithmetic operations.

It is time for our table counting the number of arithmetic operations: we count each image as costing $2^g S + (2^g - 1)M = 2.2^g - 1$, and each doubling $2.2^g S + 2.(2^g - 1)M = 4.2^g - 2$. We have seen that the codomain and arithmetic precomputation costs $2^g(g+4)$.

2^n	$g = 1$	$g = 2$	$g = 4$	$g = 8$
2^{128}	5189	21314	177956	5719880
2^{216}	9453	39218	329092	10601480
2^{250}	11170	46460	390360	12582320
2^{305}	14030	58560	492880	15899040
2^{372}	17514	73300	617768	19939408
2^{486}	23769	99906	843780	27259656

As a concrete example, the strategy for computing a 2^{602}-isogeny in dimension 2 involves 3274 doublings and 7108 images (plus 26 gluing images).

17.1. **Removing inversions.** It is also much easier to analyze the complexity where we get rid of all inversions. We treat the case $g = 2$ for simplicity.

We compute $h(P_1) = (Ax_1 : Bx_1 : Cz_1 : Dz_1)$ and $h(P_2) = (Ax_2 : Bz_2 : Cx_2 : Dz_2)$ in $2.2^g S = 8$, since $g = 2$. From this data we want $(1/A : 1/B : 1/C : 1/D)$ projectively for images, and also $(1/a_2 : 1/b_2 : 1/c_2 : 1/d_2)$ for doublings.

Batching inversions, we can compute $B/A, C/A, D/C$ in $1I + 6M + 3M$, except we don't want to actually compute the inversion so we have $(\kappa B/A, \kappa C/A, \kappa D/C)$ for some known factor λ. We then get $(\kappa^2, \kappa^2 B/A, \kappa^2 C/A, \kappa^2 D/C)$ in $1S + 3M$. We compute the inverse of these coordinates in $1I + 9M$ (except we don't actually compute the inverse), and likewise we have $(a_2 : b_2 : c_2 : d_2)$ through a Hadamard transform of $(A : B : C : D)$ and then compute the inverses in $1I + 9M$. The total cost for the codomain is than $8S + 9M + (1S + 3M) + 9M + 9M = 30M + 9S \geq 39M$. Comparing with Section 16 we see that the codomain computation is more than twice as fast, as was expected (since the normalisation process took half the time). Here, images and doublings costs $2^g S + 2^g M \leq 2.2^g M$ and $2.(2^g S + 2^g M) \leq 4.2^g M$ respectively because the coordinates of our theta null points are no longer normalised.

Depending on the number of doubling and isogeny images we need, it might make sense (especially at the beginning of the isogeny chain) to bach one inversion to gain the $1M$ (resp. $2M$) by doublings and images.

We obtain the following number of arithmetic operations, when we get rid of all inversions:

2^n	$g=2$
2^{128}	25840
2^{216}	47320
2^{250}	55990
2^{305}	70455
2^{372}	88076
2^{486}	119802

References

[BRS23] R. Barbulescu, D. Robert, and N. Sarkis. “Models of Kummer lines and Galois representations”. June 2023. In preparation. (Cit. on pp. 9, 30).

[BMP23] A. Basso, L. Maino, and G. Pope. “FESTA: Fast Encryption from Supersingular Torsion Attacks”. In: *Cryptology ePrint Archive* (2023) (cit. on p. 4).

[CD21] W. Castryck and T. Decru. “Multiradical isogenies”. In: *Cryptology ePrint Archive* (2021) (cit. on p. 36).

[CD23] W. Castryck and T. Decru. “An efficient key recovery attack on SIDH”. In: Springer-Verlag (Eurocrypt 2023), Apr. 2023 (cit. on p. 4).

[CR15] R. Cosset and D. Robert. “An algorithm for computing (ℓ,ℓ)-isogenies in polynomial time on Jacobians of hyperelliptic curves of genus 2”. In: *Mathematics of Computation* 84.294 (Nov. 2015), pp. 1953–1975. DOI: 10.1090/S0025-5718-2014-02899-8. URL: http://www.normalesup.org/~robert/pro/publications/articles/niveau.pdf. HAL: hal-00578991, eprint: 2011/143. (Cit. on pp. 4, 5, 36).

[CH17] C. Costello and H. Hisil. “A simple and compact algorithm for SIDH with arbitrary degree isogenies”. In: *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23*. Springer. 2017, pp. 303–329 (cit. on p. 21).

[CLN16] C. Costello, P. Longa, and M. Naehrig. “Efficient algorithms for supersingular isogeny Diffie-Hellman”. In: *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I 36*. Springer. 2016, pp. 572–601 (cit. on p. 21).

[DLRW23] P. Dartois, A. Leroux, D. Robert, and B. Wesolowski. “SQISignHD: New Dimensions in Cryptography”. Accepted for publication at Eurocrypt 2024. Mar. 2023. URL: http://www.normalesup.org/~robert/pro/publications/articles/SQISignHD.pdf. eprint: 2023/436, HAL: hal-04056062v1. (Cit. on pp. 3–5, 7, 8, 11, 12, 15, 23).

[DMPR23a] P. Dartois, L. Maino, G. Pope, and D. Robert. “An Algorithmic Approach to $(2,2)$-isogenies in the Theta Model and Applications to Isogeny-based Cryptography”. Nov. 2023. URL: http://www.normalesup.org/~robert/pro/publications/articles/_2_2__isogenies_in_the_theta_model.pdf. eprint: 2023/1747. (Cit. on p. 4).

[DMPR23b] P. Dartois, L. Maino, G. Pope, and D. Robert. *ThetaIsogenies*. Fast computations of isogenies in dimension two. Nov. 2023. URL: https://github.com/ThetaIsogenies/two-isogenies (cit. on p. 4).

[DJP14] L. De Feo, D. Jao, and J. Plût. "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies". In: *Journal of Mathematical Cryptology* 8.3 (2014), pp. 209–247 (cit. on pp. 4, 16).

[FLR11] J.-C. Faugère, D. Lubicz, and D. Robert. "Computing modular correspondences for abelian varieties". In: *Journal of Algebra* 343.1 (Oct. 2011), pp. 248–277. DOI: 10.1016/j.jalgebra.2011.06.031. arXiv: 0910.4668 [cs.SC]. URL: http://www.normalesup.org/~robert/pro/publications/articles/modular.pdf. HAL: hal-00426338. (Cit. on p. 36).

[LR10] D. Lubicz and D. Robert. "Efficient pairing computation with theta functions". In: ed. by G. Hanrot, F. Morain, and E. Thomé. Vol. 6197. Lecture Notes in Comput. Sci. 9th International Symposium, Nancy, France, ANTS-IX, July 19-23, 2010, Proceedings. Springer–Verlag, July 2010. DOI: 10.1007/978-3-642-14518-6_21. URL: http://www.normalesup.org/~robert/pro/publications/articles/pairings.pdf. Slides: 2010-07-ANTS-Nancy.pdf (30min, International Algorithmic Number Theory Symposium (ANTS-IX), July 2010, Nancy), HAL: hal-00528944. (Cit. on p. 8).

[LR12] D. Lubicz and D. Robert. "Computing isogenies between abelian varieties". In: *Compositio Mathematica* 148.5 (Sept. 2012), pp. 1483–1515. DOI: 10.1112/S0010437X12000243. arXiv: 1001.2016 [math.AG]. URL: http://www.normalesup.org/~robert/pro/publications/articles/isogenies.pdf. HAL: hal-00446062. (Cit. on pp. 4, 5, 36).

[LR15a] D. Lubicz and D. Robert. "A generalisation of Miller's algorithm and applications to pairing computations on abelian varieties". In: *Journal of Symbolic Computation* 67 (Mar. 2015), pp. 68–92. DOI: 10.1016/j.jsc.2014.08.001. URL: http://www.normalesup.org/~robert/pro/publications/articles/optimal.pdf. HAL: hal-00806923, eprint: 2013/192. (Cit. on p. 8).

[LR15b] D. Lubicz and D. Robert. "Computing separable isogenies in quasi-optimal time". In: *LMS Journal of Computation and Mathematics* 18 (1 Feb. 2015), pp. 198–216. DOI: 10.1112/S146115701400045X. arXiv: 1402.3628. URL: http://www.normalesup.org/~robert/pro/publications/articles/rational.pdf. HAL: hal-00954895. (Cit. on pp. 4, 5).

[LR16] D. Lubicz and D. Robert. "Arithmetic on Abelian and Kummer Varieties". In: *Finite Fields and Their Applications* 39 (May 2016), pp. 130–158. DOI: 10.1016/j.ffa.2016.01.009. URL: http://www.normalesup.org/~robert/pro/publications/articles/arithmetic.pdf. HAL: hal-01057467, eprint: 2014/493. (Cit. on p. 8).

[LR22a] D. Lubicz and D. Robert. "Fast change of level and applications to isogenies". In: *Research in Number Theory (ANTS XV Conference)* 9.1 (Dec. 2022). DOI: 10.1007/s40993-022-00407-9. URL: http://www.normalesup.org/~robert/pro/publications/articles/change_level.pdf. HAL: hal-03738315. (Cit. on pp. 4, 5, 8, 36).

[LR22b] D. Lubicz and D. Robert. "Multiradical isogenies in the theta model". Sept. 2022. In preparation. (Cit. on p. 36).

[MMPPW23] L. Maino, C. Martindale, L. Panny, G. Pope, and B. Wesolowski. "A Direct Key Recovery Attack on SIDH". In: Springer-Verlag (Eurocrypt 2023), 2023 (cit. on p. 4).

[Mum66] D. Mumford. "On the equations defining abelian varieties. I". In: *Invent. Math.* 1 (1966), pp. 287–354 (cit. on p. 5).

[Ren18] J. Renes. "Computing isogenies between Montgomery curves using the action of (0, 0)". In: *Post-Quantum Cryptography: 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*. Springer. 2018, pp. 229–247 (cit. on p. 21).

[Rob10] D. Robert. "Theta functions and cryptographic applications". PhD thesis. Université Henri-Poincaré, Nancy 1, France, July 2010. URL: http://www.normalesup.org/~robert/pro/publications/academic/phd.pdf. Slides: 2010-07-Phd-Nancy.pdf (1h, Nancy), TEL: tel-00528942. (Cit. on pp. 4, 5, 8, 9, 11, 12, 32, 34).

[Rob21] D. Robert. "Efficient algorithms for abelian varieties and their moduli spaces". HDR thesis. Université Bordeaux, June 2021. URL: http://www.normalesup.org/~robert/pro/publications/academic/hdr.pdf. Slides: 2021-06-HDR-Bordeaux.pdf (1h, Bordeaux). (Cit. on pp. 4, 5, 8, 9).

[Rob23a] D. Robert. "Breaking SIDH in polynomial time". Apr. 2023. URL: http://www.normalesup.org/~robert/pro/publications/articles/breaking_sidh.pdf. eprint: 2022/1038, HAL: hal-03943959, Slides: 2023-04-Eurocrypt.pdf (15 min, Eurocrypt 2023, April 2023, Lyon, France). (Cit. on p. 4).

[Rob23b] D. Robert. "Improving the arithmetic of Kummer lines". Aug. 2023. URL: http://www.normalesup.org/~robert/pro/publications/notes/2023-11-kummer_lines.pdf (cit. on p. 21).

[RS24] D. Robert and N. Sarkis. "Computing 2-isogenies between Kummer lines". Jan. 2024. URL: http://www.normalesup.org/~robert/pro/publications/articles/kummer_isogenies.pdf. eprint: 2024/037. (Cit. on p. 21).

Appendix A. Conversion formula between the theta model and the Montgomery model in dimension 1

These formula are extracted from [BRS23].

A.1. Theta and Montgomery. Let E/k be an elliptic curve, and $(a:b) = (\theta_0(0_E), \theta_1(0_E))$ be its theta null point. We give formula to convert the theta points $(\theta_0(P) : \theta_1(P))$ into the Montgomery coordinates $(x(P) : z(P))$.

When the theta null point is rational, the elliptic curve E admits both a rational Montgomery model and a rational Legendre model. They are given by

$$y^2 = x(x-\alpha)(x-1/\alpha) = x(x^2 + \mathcal{A}x + 1)$$

and (up to a quadratic twist, which is harmless because we work on the Kummer line anyway) by

$$y^2 = x(x-1)(x-\lambda).$$

These constants are determined as follows: let $(A : B)$ be the dual coordinates of the canonical 2-isogenous curve (we will only need their square). We have

$$A^2 = a^2 + b^2, B^2 = a^2 - b^2, \tag{4}$$

$$\alpha = A^2/B^2 = (a^2+b^2)/(a^2-b^2), \tag{5}$$

$$\lambda = \alpha^2 = A^4/B^4 = (a^2+b^2)^2/(a^2-b^2)^2, \tag{6}$$

$$\mathcal{A} = -(\alpha + 1/\alpha) = -(\alpha^2+1)/\alpha = -(A^4+B^4)/(A^2B^2) = -2(a^4+b^4)/(a^4-b^4), \tag{7}$$

$$(\mathcal{A}+2)/4 = -b^4/(a^4-b^4). \tag{8}$$

Conversely, from $\mathcal{A}$, we can recover $(a : b)$ via

$$\alpha + 1/\alpha = -\mathcal{A}, \tag{9}$$

$$A^2/B^2 = \alpha, \tag{10}$$

$$a^2 = A^2 + B^2, b^2 = A^2 - B^2, (a^2 : b^2) = (\alpha+1 : \alpha-1). \tag{11}$$

We note that if $(a : b)$ is a solution, then $(a : \zeta b)$ also with $\zeta \in \mu_4$, these correspond to different theta structures.

With these constants defined, we can now explain how to convert the points. If $P = (x : z)$ in Montgomery coordinates, then

$$(\theta_0(P) : \theta_1(P)) = (a(x-z) : b(x+z)). \tag{12}$$

Conversely, if $P = (\theta_0 : \theta_1)$, then in Montgomery coordinates

$$(x(P) : z(P)) = (a\theta_1 + b\theta_0 : a\theta_1 - b\theta_0). \tag{13}$$

On the theta model $0_E = (a : b)$, we have a canonical basis of the 2-torsion given by $T_1 = (a : -b)$ and $T_2 = (b : a)$. We have a canonical basis of the 4-torsion given by $T_1' = (1 : 0)$ above T_1 and $T_2' = (1 : 1)$ above T_2. The map above sends T_1 to $(0 : 1)$ in the Montgomery model, T_1' to $(1 : 1)$, T_2 to $(A^2 : B^2)$, T_2' to $(a+b : a-b)$.

So conversely, given a Montgomery curve, the canonical point $T' = (1 : 1)$ of 4-torsion above the 2-torsion point $T = (0 : 1)$ and a second point $T'' = (r : s)$ above another point of 2-torsion, then the theta null point $(a : b)$ induced by the basis (T', T'') of the 4-torsion is given by $(r+s : r-s)$.

For the case of a general elliptic curve E with a basis (T', T'') of the 4-torsion, we first convert E to a Montgomery model by sending T' to $(1 : 1)$ and $T = 2T'$ to $(0 : 1)$, the map is then $x \mapsto (x - x(T))/(x(T') - x(T))$. Then we apply the above formula to the image of T''.

A.2. **The alternative Montgomery model.**

When we have a theta model, we can also introduce the dual theta coordinates

$$(\theta_0' : \theta_1') = (\theta_0 + \theta_1 : \theta_0 - \theta_1),$$

in particular the dual theta null point is given by $(a' : b') = (a+b : a-b)$. We can construct another Montgomery model by replacing in the above formula $(a, b, \theta_0, \theta_1)$ by $(a', b', \theta_0', \theta_1')$.

Plugging in this different model the equations expressing $(a', b', \theta'_0, \theta'_1)$ in terms of $(a, b, \theta_0, \theta_1)$, we obtain alternative formulas:

$$A'^2 = a'^2 + b'^2 = 2(a^2 + b^2), B'^2 = a'^2 - b'^2 = 4ab, \tag{14}$$

$$\alpha' = A'^2/B'^2 = (a^2 + b^2)/(2ab), \lambda' = \alpha'^2, \tag{15}$$

$$\mathcal{A}' = -(\alpha' + 1/\alpha') = -(a^4 + 6a^2b^2 + b^4)/(2(a^3b + ab^3)), \tag{16}$$

$$P = (x : z) \mapsto (\theta_0(P), \theta_1(P)) = (ax - bz : bx - az), \tag{17}$$

$$(\theta_0, \theta_1) \mapsto (x(P) : z(P)) = (a\theta_0 - b\theta_1 : b\theta_0 - a\theta_1). \tag{18}$$

Appendix B. The algebraic theta transformation formula

We briefly describe the algebraic theta transformation formula in level n, see [Rob10] for more details.

Assume that we have a symmetric theta structure of level n, induced by a symplectic basis $(e'_1, \dots, e'_g, f'_1, \dots, f'_g)$ of $A[2n]$. Let $M \in \mathrm{Sp}_{2g}(\mathbb{Z})$, this induces a symplectic matrix of $\mathbb{Z}/n\mathbb{Z}$ hence a symplectic change of variable on our basis above. The new symplectic basis will give a new symmetric theta structure, hence a linear change of variable on our theta functions. We now describe this action.

The group $\mathrm{Sp}_{2g}(\mathbb{Z})$ is generated by these three types of matrices:

(1) The matrix $\mathcal{S} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$. This matrix acts by the Hadamard transform H.
(2) The matrix $M = \begin{pmatrix} A & 0 \\ 0 & A^{-T} \end{pmatrix}$ where A is in $\mathrm{Gl}_g(\mathbb{Z})$. This matrix acts by $\theta_i \mapsto \theta_{A.i}$, where the action is the natural action of A on $(\mathbb{Z}/n\mathbb{Z})^g$ induced by the action of A on $\mathbb{Z}^g$.
(3) The matrix $M = \begin{pmatrix} 1 & C \\ 0 & 1 \end{pmatrix}$ where C is symmetric. Fix ζ a primitive $2n$-root of unity induced by a symplectic basis of the $2n$-torsion inducing our theta strucutre. This matrix acts by $\theta_i \mapsto \zeta^{i^T C i}\theta_i$. For instance if C is diagonal with the only non zero entry being a one at position (j, j), the action is $\theta_x \mapsto \zeta^{x_j^2}\theta_x$. If C is diagonal with only two non zero entries at position (i, j), (j, i), the action is $\theta_x \mapsto \zeta^{2x_i x_j}\theta_x$.

Example B.1. In dimension 1, $\Gamma/\Gamma(2, 4)$ is of cardinal $6 * 4 = 24$, the modular action induces all possible permutation on the four points of ramification of $E \to \mathbb{P}^1$.

Example B.2. In dimension 2, $\Gamma/\Gamma(2, 4)$ is of cardinal $720 * 2^4$. If the abelian surface is a product of two elliptic curves, the subgroup preserving a product theta structure is of cardinal $2 * 24 * 24$ so is of index 10. There are ten even theta constants of level $(2, 2)$, an abelian surface is a product theta if and only if the even theta constant $\theta[11; 11](0) = 0$. The index 10 corresponds to sending this null theta constant to one of the other 10 even theta.

B.1. **Directly computing theta constants.**

The original proposal of these notes suggested to compute the theta constants in level 2 by going through the product theta structure (via the dimension 1 conversion of Appendix A) followed by a symplectic transform.

However, Sage's linear algebra is quite slow, so the current implementation directly computes the theta constants from a symplectic basis on the elliptic product.

An advantage of this approach is as follow: going to the direct product theta structure involve starting with a tuple (P_1, P_2) in Montgomery coordinate, applying a linear transform (in dimension 1) on the coordinates of P_i to obtain theta coordinates, take the Segre embedding to get the product theta structure coordinates, and apply a linear transform again (in dimension 2) to get the theta coordinates compatible with our kernel. With the direct approach we take the Segre embedding on the Montgomery coordinate and directly apply a dimension 2 base change; this save the 2 dimension 1 linear base change.

We briefly explain how this works: on a dimension 1 theta model $(a : b)$, the point $T_2 = (-a : b)$ as for symmetric lifts in the theta group the linear transformation $(X, Z) \mapsto (X, -Z)$ (associated to the 4-torsion point $(1 : 0)$) and $(X, Z) \mapsto (-X, Z)$ (associated to the 4-torsion point $(0 : 1)$). And the point $T_1 = (b : a)$ as for symmetric lifts the linear transformation $(X, Z) \mapsto (Z, X)$ (associated to the 4-torsion point $(1 : 1)$) and the linear transformation $(X, Z) \mapsto (-Z, -X)$ (associated to the 4-torsion point $(-1 : 1)$).

From the conversion maps Appendix A, we see that on a Montgomery curve, the 4-torsion point $T = (1 : 1)$ above $(0 : 1)$ is associated to the linear map $g_T : (X, Z) \mapsto (-Z, -X)$, while the point $(-1 : 1)$ is associated to $(X, Z) \mapsto (Z, X)$.

For a general elliptic curve, if $T = (x, y, z)$ is a point of 4-torsion and $2T = (u, v, w)$, we can map T to the Montgomery point $(1 : 1)$ via the linear transformation (in the Kummer line): $(X : Z) \mapsto (X', Z') = (zwX - zuZ : (xw - zu)Z)$. It follows that in (X, Z) coordinates, the action of g_T is given by

$$M^T U M^{-T} = \begin{pmatrix} uz/(wx - uz) & zw/(wx - uz) \\ (-wx^2 + 2uzx)/(-zwx + uz^2) & -uz/(wx - uz) \end{pmatrix},$$

with $M = \begin{pmatrix} wz & -zu \\ 0 & xw - uz \end{pmatrix}$, $U = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Given a symplectic decomposition $A[2] = K_1 \oplus K_2$ and a decomposition $A[4] = K_1' \oplus K_2'$ above it, and a section $s \in \Gamma(L)$ (L of level 2), we can construct a basis of level 2 theta functions by taking the trace (provided it is non zero) θ_0 of s under the level 2 elements induced by the linear transformation $g_{T'}$, $T' \in K_2'$ above each $T \in K_2$. Then for $i \in K_1$, $i' \in K_1'$ above i, we let $\theta_i = g_{i'} \cdot \theta_0$.

As an example, on a Montgomery curve we have $T_2 = (-1 : 1)$ which acts by $g_2 \cdot (X, Z) = (-Z, -X)$. Taking the trace of X under this action we get: $\theta_0 = \mathrm{id} \cdot X + g_1 \cdot X = X - Z$.

Let $T_1 = (a+b : a-b)$ be another point of 4-torsion; its double is then $(a^2+b^2 : a^2-b^2)$. So with $x = a + b$,$z = a - b$,$u = a^2 + b^2$,$w = a^2 - b^2$, we compute $\theta_1 = g_1 \cdot \theta_0 = g_1 \cdot (X - Z) = (uz - wxx/z + 2ux)/(wx - uz)X + z(w + u)/(wx - uz)Z = b/aX + b/aZ$. Hence we recover exactly the base change $(X, Z) \mapsto (a(X - Z) : b(X + Z))$ of Appendix A from Montgomery to theta.

We can use the same strategy to compute the theta null point associated to a symplectic basis of the 4-torsion on a product of elliptic curve. If $T = (T_1, T_2) \in E_1 \times E_2$ is a point of 4-torsion, the associated element g_T is given by $g_T = g_{T_1} \otimes g_{T_2}$. Then we can (for instance) take θ_0 as the trace of $X_1 \otimes X_2$ under $\widetilde{K}_2$, with $(g_{T_1} \otimes g_{T_2}) \cdot (X_1 \otimes X_2) = (g_{T_1} \cdot X_1) \otimes (g_{T_2} \cdot X_2)$.

B.2. **The choice of signs.** We can use this action to explore our choice of sign. Fix $(e_1', \dots, e_g', f_1', \dots, f_g')$ a symplectic basis of $A[4]$ inducing our symmetric theta structure, and let $K = \langle f_1, \dots, f_g \rangle$ where $f_i = 2f_i'$ our kernel, and let $f : A \to B = A/K$. The image $f(e_i')$ gives an isotropic subgroup $B_1[4]$ of $B[4]$, while $f(f_i')$ gives $B_2[2]$ such that we have a symplectic decomposition $B[2] = B_1[2] \oplus B_2[2]$. The choice of sign in our isogenous theta constant corresponds to fixing a symplectic basis of $B[4]$ compatible with the $f(e_i'), f(f_i)$.

These choice of signs corresponds to the action of the matrix $M = \begin{pmatrix} 1 & C \\ 0 & 1 \end{pmatrix}$ on B. We note that there are two kind of action: the one where C leaves $B[2]$ invariant, this corresponds on A to leaving the f_i' invariant and changing the points of 8-torsion above them. An example is given by C which is diagonal with entries equal to 0 modulo 2. The second type changes $B[2]$, hence changes the f_i' (but in a way that is still compatible with our theta structure on A).

This shows that we have $g(g+1)/2$ choice of sign possible on B (because the matrix C has to be symmetric), but that if we fix the 4-torsion f_1', f_2' on A we now only have g choice of signs. Algebraically these can be determined as follow: normalize the f_i', because f_i' is of order 4 this still leaves a choice of sign for each f_i' (specifically, we have an equation $\lambda_i^4 = C_i$, but the duplication formula only involve the λ_i^2). This is enough to compute the isogenous theta null point by Section 11.

In fact this is also enough to also normalize all of $K[4]$: normalize $f_i' + f_j'$ via $\widetilde{f_i' + f_j'} + \widetilde{f_j'} = \widetilde{f_i'} + \tilde{f_j}$, this involve an equation $\lambda_{ij}^2 = C_{ij}$ so no choice of sign since the duplication formula only involve the λ_{ij}^2. Next use the differential additions and three way additions to normalize any remaining point. See [Rob10] for more details.

Example B.3. When $g = 1$, our point of 4-torsion is $T' = (1 : 0)$, which is normalized to $(\lambda, 0)$ where $\lambda^4 = A^2B^2$. Since we know $A^2 = a^2 + b^2, B^2 = a^2 - b^2$, fixing $\lambda^2 = AB$ is enough to fix $(A : B)$; changing to $\lambda^2 = -AB$ gives $(A : -B)$, and this correspond to normalising T' by another point of 8-torsion above it.

When $g = 2$, let's say that our point of 8-torsion determined the coefficients $(A : B : C : D)$. Our points of four torsion (suitably normalised by the 8-torsion) is then f_1' which determines (AB, CD, AB, CD) and f_2' which determines (AC, BD, AC, BD). Changing f_1' by $f_1' + e_2$ and f_2' by $f_2' + e_1$ will give instead the coefficients $(AB, -CD, AB, -CD)$ and $(AC, -BD, AC, -BD)$, hence corresponds to changing the sign of D. Keeping f_1' and f_2' but changing the points of 8-torsion above f_1', hence changing their normalisation, then f_1' will now give $(-AB, -CD, -AB, -CD)$, hence this changes the sign of B and D. Similarly changing the point of 8-torsion above f_2' will change the sign of C and D. We do recover that we have 2 possible choice of signs when the f_1', f_2' are fixed, and one more when we change them (while staying compatible with the theta structure).

By the way, we remark that if we don't normalize f_1', f_2', we recover $\lambda_1 AB, \lambda_1 CD, \lambda_2 AC, \lambda_2 BD$ for some unknown projective factors λ_1, λ_2. Since we also know A^2, B^2, C^2, D^2, it is easy to find equations for λ_1^2, λ_2^2. By the above discussion, all 4 solutions of these two equations determine a valid isogenous (dual) theta null point; but they require to take two square roots.

The advantage of requiring points of 8-torsion is to dispense with these square roots. However, for a 2^n-isogeny, this requires to start with points of 2^{n+2}-torsion above our kernel K. If we only have K, we could switch to the square root method for the second to last (which requires 2 square roots because we have points of 4-torsion), and last (which requires 3 square roots because we now only have the 2-torsion) isogenies. This slow down the last two codomain computations, however this does not change the images.

Example B.4. When $g = 2$, it was remarked by Giacomo Pope that we don't need to start our isogeny chain with isotropic 2^{n+2}-torsion points P_1'', P_2'' above the kernel P_1, P_2, we just need non necessarily isotropic points.

The first $n - 2$ steps are the same, we just need to explain why the formula still work at the last two steps.

At the penultimate step, assume that our 8-torsion T'''_1, T'''_2 is correct and that $f(T'''_1) = (x : x : y : y)$, and $f(T'''_2) = (z : t : z : t)$ in θ' coordinate. If we change our points by $T \in K_2$ this changes nothing because K_2 is our kernel so the images are the same. If $T \in K_1$, T acts by a shift in θ coordinates, so by a sign in θ'-coordinate. So if T'''_2 is wrong we could have $f(T"_2) = (z : t : -z : -t)$. If we look at our code, we see that this changes by a sign one of the constant $(A : B : C : D)$ we compute.

So when the 4-torsion is correct, but the 8 torsion is wrong, then in the codomain the 4-torsion is wrong, rather than getting $(x : x : y : y)$ say above $(B : A : D : C)$, we get $(x : -x : y : -y)$. So instead of getting (A, B, C, D), we get $-B, -D$, but $(A : -B : C : -D)$ is still a valid theta null point. But compared to our true image map, our image map has sign flips. In particular, the 4-torsion point which was sent to $(B : A : D : C)$ before, is now sent to $(B : -A : D : -C)$. But compared our new theta null point $(A : -B : C : -D)$, this is still the correct 2-torsion point for the next isogeny!

We can do a similar reasoning for the last isogeny. Here even the 4-torsion is wrong, so our 2-torsion on the codomain is wrong: say rather than getting $(B : A : D : C)$ we get $(B : -A : D : -C)$. Then it follows that our 8-torsion point is sent to a 4-torsion point above this 2-torsion, which means it is of the form (x, ix, y, iy) or $(x, -ix, y, -iy)$. If we use this point, this will change B to iB say (and maybe for D too), but this kind of change also comes from a symplectic automorphism. And since we don't take any more kernel, we don't care what the images of our 4-torsion points are anyway afterwards.

Appendix C. Other applications of the duplication formula

By now the reader should be convinced that the duplication formula for theta functions allows for very fast 2-isogeny formula. The Sage implementation (due to Pierrick Dartois, Sabrina Kunzweiler, Luciano Maino, Giacomo Pope and myself) shows a nice speed up compared to Richelot formula, this will be detailed in a follow up work.

One can wonder if theta coordinates can be used for other applications. The following use case was suggested by Sabrina Kunzweiler in dimension 2: look at CGL like hash function in dimension 2 in the theta model, by computing chuncks of 2^n-isogenies.

Altough the formula from these notes can be used, there are two problems remaining:

- Compute the symplectic change of basis to make the kernels K in a way such that $K[4]$ is canonical. A method is probably to compute a basis of $A[4]$ compatible with our theta structure (unfortunately this involves square roots), as described in Section 16.6, complete $K[4]$ into a symplectic basis, compute the symplectic change of basis, eg using Weil pairings (but since we are in level 2 this involves more square roots), and apply the theta transformation formula. If we start with a Jacobian, and we compute the theta constants through Thomaes formula, a better method would be to use formula due to Sabrina which gives the correct square roots to take in Thomae's formula according to a fixed symplectic basis of $\operatorname{Jac} C[4]$.
- Once K is in suitable form, and more generally we have e'_1, e'_2, f'_1, f'_2 a symplectic basis of $A[2^n]$, such that the induced symplectic basis e_1, e_2, f_1, f_2 a symplectic basis of $A[4]$ is compatible with the theta structure and $K = \langle f'_1, f'_2 \rangle$ (so that $\langle f_1, f_2 \rangle = K[4]$ and K is compatible with our theta structure); we can compute the isogeny $f : A \to B$ and the images $f(e'_1), f(e'_2)$.

  We now need to regenerate the 2^n-torsion of B by computing a symplectic basis $f(e'_1), f(e'_2), g'_1, g'_2$ and take a kernel K' whose intersection with $\langle f(e'_1), f(e'_2) \rangle$ is trivial (so the next isogeny has no (partial) backtracking.

  Since we are in level 2 however it is not clear how to best do this step. Sample random points, multiply by the cofactor, and do some Weil pairing computations (which as mentioned involve square roots since we are in level 2)? Go back to a Jacobian representation to compute the 2^n-torsion and switch back to theta afterwards?

  We leave the best method as an open problem.

What is much easier though is to only do chuncks of 2-isogenies and use multiradical 2-isogeny formula in the spirit of [CD21] to regenerate the 2-torsion at each step (in a non backtracking way). Remember that in dimension g, multiradical formula will involve $g(g+1)/2$ square roots.

Using the duplication formula, 2-isogeny multiradical formula are particularly simple in the level 2 theta model in dimension 1 and 2:

- In dimension 1 start with the theta null point $(a : b)$, apply the square operator S to get $(a^2 : b^2)$, the Hadamard operator $H(x : y) = (x+y : x-y)$ to get $(A^2 : B^2) = (a^2+b^2 : a^2-b^2)$. Take an arbitrary square root of B^2/A^2. To prevent an inversion, a solution is to instead take an arbitrary square root AB (depending on the current bit of the message we want to hash) of A^2B^2, which give the projective dual isogeneous theta null point $(A^2 : AB)$. Apply the Hadamard operator H again to get $(a_2 : b_2)$, this is our isogeneous theta null point. Iterate for each bit of our message. The whole formula cost one square root, $2S + 1M + 4a$.
- In dimension 2, the same formula hold: start with the theta null point $(a : b : c : d)$, apply S to get $(a^2 : b^2 : c^2 : d^2)$, then $H(x : y : z : t) = (x+y+z+t, x-y+z-t, x+y-z-t, x-y-z+t)$ to get $(A^2 : B^2 : C^2 : D^2)$, take arbitrary square roots (depending on our bits) AB, AC, AD of A^2B^2, A^2C^2, A^2D^2 to get $(A^2 : AB : AC : AD)$, and apply H again to get $(a_2 : b_2 : c_2 : d_2)$ for a total cost of three square roots and $4S + 3m + 8a$.

It would be interesting to compare these methods with the usual methods:

- In dimension 1 using the modular polynomial ϕ_2, removing the linear factor coming from the preceding isogeny and solving a degree 2 equation
- In dimension 2 using Richelot formula, factorizing 3 degree 2 polynomials at each step.

We also leave that for future work!

An interesting open problem is to generalize this approach to higher dimension. From the theta transformation formula, one can see that we can only take $g(g+1)/2$ arbitrary square roots (the ones coming from $e_i, e_i + e_j$ where e_i is a basis of $(\mathbb{Z}/2\mathbb{Z})^g$), once these are taken the rest are fixed. But I don't know how to most efficiently determine these remaining choices (apart from a rather expensive Grobner basis computation), unless we already have some information on the 4-torsion on the domain. When $g = 1, 2$, all choices are possible, so this problem goes away.

Another interesting direction is to extend these 2-radical formulas to 4-radical and 8-radical formula. Using the generic isogeny algorithm [LR12; CR15; LR22a] combined with [FLR11], I have a generic multiradical isogeny formula in any dimension in the theta model [LR22b]. But we have just seen that 2^n-isogenies in the theta model can be made much faster than the generic isogeny computation, so it's probably better to find direct radical isogeny formula for $\ell = 4, 8$.

INRIA Bordeaux–Sud-Ouest, 200 avenue de la Vieille Tour, 33405 Talence Cedex FRANCE
Email address: damien.robert@inria.fr
URL: http://www.normalesup.org/~robert/

Institut de Mathématiques de Bordeaux, 351 cours de la liberation, 33405 Talence cedex FRANCE