CHASING RABBITS THROUGH HYPERCUBES: BETTER ALGORITHMS FOR HIGHER DIMENSIONAL 2-ISOGENY COMPUTATIONS

PIERRICK DARTOIS AND MAX DUPARC

Abstract. The devastating attacks against SIDH (Supersingular Isogeny Diffie-Hellman) [1, 2, 3] have popularised the practical use of isogenies of dimension 2 and above in cryptography. Though this effort was primarily focused on dimension 2, 4-dimensional isogenies have been used in several isogeny-based cryptographic constructions including SQIsignHD [4], SQIPrime [5], Pegasis [6], qt-Pegasis [7] and MIKE [8]. These isogenies are also interesting for number theoretic applications related to higher dimensional isogeny graphs. In 2024, a work by Pierrick Dartois [9] introduced algorithms to compute efficiently chains of 2-isogenies with Mumford's level-2 theta coordinates in all dimensions, focusing on cryptographic applications in dimension 4. In this paper, we improve Dartois' results by providing a simpler and faster method to compute generic isogenies in any dimension, and new computation and evaluation algorithms adapted to gluing isogenies from a product of four elliptic curves, with techniques that generalise a previous work by Max Duparc [10] in dimension 2. Unlike previous algorithms by Dartois, the algorithms we propose are both easy to implement and naturally constant time. We apply our results to propose the first constant time C implementation of a 4-dimensional chain of 2-isogenies, adapted to the qt-Pegasis algorithm and running in less than 25 ms for a 500 bit prime. With our new gluing evaluation method, we are able to work fully over $\mathbb{F}_p$ instead of $\mathbb{F}_{p^2}$, allowing further efficiency gains. Indeed, our new formulae accelerate the proof of concept SageMath implementation of qt-Pegasis by up to 19% for a 500 bit prime.

1. Introduction

Foundational works by David Lubicz and Damien Robert [11, 12, 13] on efficient computations of ℓ-isogenies between abelian varieties of any dimension g using Mumford's theta coordinates [14] are aimed primarily at asymptotic efficient in ℓ and g. Finding practically efficient algorithms to compute 2-isogenies in dimensions $g \leq 4$ has become crucial since the attacks against SIDH (Supersingular Isogeny Diffie-Hellman) in 2022 [1, 2, 3] that popularised the constructive use of higher dimensional isogenies in isogeny-based cryptography. Pierrick Dartois, Luciano Maino, Giacomo Pope and Damien Robert addressed this question in dimension 2 in [15], later generalised to all dimensions by Dartois [9] (see also [16, Chapter 6]), with a specific focus on dimension 4. As Lubicz's and Robert's prior works, both of these contributions use Mumford's theta coordinates but in level 2 in order to minimise the number of theta coordinates needed (2^g in dimension g). Working in level 2 to compute 2-isogenies leads to some technical difficulties, including the need for torsion

points lying above the kernel of the isogenies we want to compute and special formulae to compute and evaluate *gluing isogenies* (isogenies starting from products of abelian varieties). To address part of these issues, a new approach called Superglue [10] has been introduced by Max Duparc to more efficiently compute gluing 2-isogenies in dimension 2.

Though less popular than 2-dimensional isogenies, 4-dimensional isogenies have several important cryptographic applications. They were first introduced to cryptography in SQIsignHD [4] to provide a fast digital signature algorithm based on SQIsign [17]. SQIPrime [5] also proposes a 4-dimensional variant of SQIsign. Though 2-dimensional variants of SQIsign [18, 19, 5, 20] have supplanted 4-dimensional variants, new cryptographic applications have also emerged. Pegasis [6] and its follow-up improvement qt-Pegasis [7] use 4-dimensional isogenies and are currently the most efficient algorithms to compute the ideal class group action on oriented supersingular elliptic curves by any ideal, without restriction. Such an unrestricted effective group action has many cryptographic applications [21], including threshold cryptographic schemes for secure multi-party computations [22] that will be the focus of the next NIST cryptography standardisation call [23]. Isogenies in dimension 4 are also a key component of the Module Isogeny Key Exchange (MIKE) [8] that could compete with the Commutative Supersingular Isogeny Diffie-Hellman (CSIDH) [24], as a key exchange not vulnerable to sub-exponential quantum attacks [25, 26].

Algorithms to compute 2-dimensional isogenies have been designed to facilitate constant time implementations [15, 10] but this feature has been harder to achieve and has received less attention in dimensions above 2. By a *constant-time* algorithm, we mean an algorithm whose sequence of operations is uniform on the inputs (which could be secret data). A constant time implementation is meant as a countermeasure to *side channel attacks* relying on measurements of running times, power consumption, or electromagnetic signals during the execution of a cryptographic protocol.

Beyond cryptography, 4-dimensional isogenies could also have other number theoretic applications, especially for costly computations where efficiency is critical. This is in particular the case for modular polynomial computations. In [27], Sabrina Kunzweiler and Damien Robert introduced an asymptotically optimal algorithm to compute modular polynomials by deformation that relies on 2-dimensional isogeny computations. This method could be generalised to the computation of Igusa modular polynomials (in the moduli space of abelian surfaces), which would involve 4-dimensional isogenies.

Our contribution. Motivated by cryptographic applications and qt-Pegasis in particular, this work focuses on the efficient computation of 2-isogenies in level 2 theta coordinates, especially in dimension 4. In Section 3, we improve the previous algorithms introduced in [9, 16] to compute codomains of 2-isogenies by proposing a general construction based on graph theory. We achieve significant efficiency gains for all dimensions $g \geq 3$ and tie with the best formulae in dimension $g = 2$, while proposing a method which is simple to implement in constant time, unlike the previous ones. In Section 4, we generalise the 2-dimensional approach from [10] to propose efficient algorithms to

compute the codomain and evaluate gluing 2-isogenies from a product of 4 elliptic curves, which appear in qt-Pegasis in particular. Both of these algorithms are designed to be easy to implement in constant time, and the evaluation algorithm is designed to work indifferently on curves and their quadratic twist for efficiency reasons. In Section 5, we detail how these algorithms can be used to implement the 4-dimensional 2-isogeny chain computation involved in qt-Pegasis in constant time and by using only $\mathbb{F}_p$ arithmetic, enabling further efficiency gains.

Acknowledgements. This work is supported by the France 2030 program under grant ANR-22-PETQ-0008 (PQ-TLS) and BPI France project RESQUE. We would like to especially thank Sina Schaeffler, whose support was decisive in ensuring the completion of this project in time. This is a low-co2 research paper: https://tcs4f.org/low-co2-v1. This research was developed, written, submitted and will be presented without the use of air travel.

2. Preliminaries

2.1. **Basic notation.** Let k be a field with $\mathrm{char}(k) \neq 2$ with $\overline{k}$ its algebraic closure. In the following, all objects and scalars we shall consider will be defined over k. In our applications, k will be a finite field but our algorithms remain valid over other fields. The cost of multiplication, squaring, low-cost multiplication by special constants, and addition or subtraction over k will be denoted by $\mathbf{M}, \mathbf{S}, \mathbf{m}$ and $\mathbf{a}$ respectively. We shall denote by $\odot$ the *element-wise product* of vectors $(x_\mathbf{i})_\mathbf{i} \odot (y_\mathbf{i})_\mathbf{i} := (x_\mathbf{i} \cdot y_\mathbf{i})_\mathbf{i}$ and define iterates $(x_\mathbf{i})_\mathbf{i}^{\odot n} := (x_\mathbf{i}^n)_\mathbf{i}$ for all $n \in \mathbb{Z}$. We also denote by $\otimes$ the *tensor product* $(u_i)_{1\leq i\leq n_1} \otimes (v_j)_{1\leq j\leq n_2} := (u_i \cdot v_j)_{1\leq i\leq n_1, 1\leq j\leq n_2}$, where the i, j are ordered in increasing order for $i + n_1 j$, and denote by $\langle \mathbf{x} | \mathbf{y} \rangle := \sum_i x_i \cdot y_i$ the standard scalar product.

2.2. **Principally polarised abelian varieties and their isogenies.** We recall here some basics on abelian varieties and refer to [28, 29] for a more detailed introduction. Formally, an *abelian variety* A over k is a complete and geometrically integral group k-variety. More concretely, the reader should keep in mind that abelian varieties are projective and have an algebraic commutative group law. They generalise elliptic curves to higher dimensions.

An *isogeny* $f : A \to B$ between abelian varieties over k is a homomorphism (of k-varieties and groups) which is surjective and has finite kernel (both over $\overline{k}$). Isogenies preserve the dimension ($\dim(A) = \dim(B)$) and are determined by their kernel up to post-composition by an isomorphism when they are seperable. To an abelian variety A/k we can associate its *dual* $\widehat{A}/k$ which has the same dimension and to an isogeny $f : A \to B$ its *dual* $\widehat{f} : \widehat{B} \to \widehat{A}$, so that $\widehat{\widehat{A}} \cong A$ and $\widehat{\widehat{f}} = f$.

A *polarisation* of an abelian variety A/k is an isogeny $\lambda_A : A \to \widehat{A}$ such that $\widehat{\lambda}_A = \lambda_A$. We say it is *principal* and that (A, λ_A) is *principally polarised* when λ_A is an isomorphism $A \xrightarrow{\sim} \widehat{A}$. In the following, we shall always work with *Principally Polarised Abelian Varieties* (PPAVs) and drop the mention of the polarisation. To an isogeny $f : (A, \lambda_A) \to (B, \lambda_B)$ between PPAVs we can associate a *polarised dual* $\widetilde{f} : (B, \lambda_B) \to (A, \lambda_A)$ given by $\widetilde{f} := \lambda_A^{-1} \circ \widehat{f} \circ \lambda_B$. We say that f is an ℓ-isogeny

for $\ell \in \mathbb{N}_{>0}$ if $\widetilde{f} \circ f = [\ell]_A$, with $[\ell]_A$ being the multiplication by ℓ on A. This property is always satisfied by isogenies between elliptic curves but is not a general fact in higher dimension.

2.3. **An introduction to theta structures.** Let (A, λ_A) be a PPAV of dimension g over k and $n \in \mathbb{N}_{>0}$. We know that $A[n] \simeq (\mathbb{Z}/n\mathbb{Z})^{2g} \simeq (\mathbb{Z}/n\mathbb{Z})^g \times \widehat{(\mathbb{Z}/n\mathbb{Z})^g}$, where $\widehat{(\mathbb{Z}/n\mathbb{Z})^g}$ denotes the group of characters $(\mathbb{Z}/n\mathbb{Z})^g \to \overline{k}^*$. A *symplectic representation* is an isomorphism $\mu : A[n] \xrightarrow{\sim} (\mathbb{Z}/n\mathbb{Z})^g \times \widehat{(\mathbb{Z}/n\mathbb{Z})^g}$ that respects the n-th Weil pairing $e_n^{\lambda_A}$ on $A[n]$ with respect to the polarisation λ_A and the canonical pairing of its codomain.

$$\forall P_1, P_2 \in A[n] \text{ with } \mu(P_j) = (\mathbf{i}_j, \chi_j) \text{ for } j \in \{1,2\}, \quad e_n^{\lambda_A}(P_1, P_2) = \chi_2(\mathbf{i}_1)\chi_1(\mathbf{i}_2)^{-1}$$

Let $(\mathbf{e}_1, \cdots, \mathbf{e}_g)$ be the canonical basis of $(\mathbb{Z}/n\mathbb{Z})^g$ given by $\mathbf{e}_{l,m} = \delta_{l,m}$ for all $l, m \in [\![1, g]\!]$ with $\delta_{l,m} = 1$ if $l = m$ and 0 otherwise (Kronecker's delta). Let $\zeta \in \overline{k}^*$ be a primitive n-th root of unity and define the dual canonical basis $(\chi_1, \cdots, \chi_g)$ of $\widehat{(\mathbb{Z}/n\mathbb{Z})^g}$ by $\chi_l(\mathbf{x}) = \zeta^{\langle \mathbf{e}_l | \mathbf{x} \rangle}$ for all $l \in [\![1, g]\!]$. Symplectic representations μ represent the unique representation of $A[n]$ in a *(ζ-)symplectic basis* $(S_1, \cdots, S_g, T_1, \cdots, T_g)$ of $A[n]$ with $\mu(S_m) = (\mathbf{e}_m, 1)$ and $\mu(T_m) = (0, \chi_m)$ for all $m \in [\![1, g]\!]$. Such basis should satisfy

$$\forall l, m \in [\![1, g]\!], \quad e_n^{\lambda_A}(S_l, S_m) = e_n^{\lambda_A}(T_l, T_m) = 1 \quad \text{and} \quad e_n^{\lambda_A}(S_l, T_m) = \zeta^{\delta_{l,m}}.$$

We can now define theta structures using the definition introduced in [10, Definition 4], which is simpler than Mumford's original definition [14, p. 297] involving line bundles and theta groups.

Definition 2.1. A *theta structure Θ_A of level n* on a PPAV A/k of dimension g is given by a symplectic representation $\mu_A : A[n] \xrightarrow{\sim} (\mathbb{Z}/n\mathbb{Z})^g \times \widehat{(\mathbb{Z}/n\mathbb{Z})^g}$ and a map to the projective space $\theta^A : A \to \mathbb{P}^{n^g-1}_{\overline{k}}$ whose components $\theta^A_{\mathbf{i}}$ ($\mathbf{i} \in (\mathbb{Z}/n\mathbb{Z})^g$) are called *theta coordinates*. They should satisfy the *theta group action relation*:

$$(1) \qquad \forall P \in A, Q \in A[n] \text{ with } \mu_A(Q) = (\mathbf{i}_Q, \chi_Q), \mathbf{i} \in (\mathbb{Z}/n\mathbb{Z})^g, \quad \theta^A_{\mathbf{i}}(P+Q) = \chi_Q(\mathbf{i})^{-1}\theta^A_{\mathbf{i}+\mathbf{i}_Q}(P).$$

We say that Θ_A is *symmetric* when $\theta^A_{\mathbf{i}}(-P) = \theta^A_{\mathbf{i}}(P)$ for all $\mathbf{i} \in (\mathbb{Z}/n\mathbb{Z})^g$ and $P \in A$. In the following, we shall always assume theta structures to be symmetric.

In order to minimise the number of theta coordinates for efficiency reasons, we will solely work with symmetric theta structures of level $n = 2$. Then, if A is not a polarised product, such a theta structure $\Theta_A = (\mu_A, \theta^A)$ induces an embedding $\theta^A : A/\pm \hookrightarrow \mathbb{P}^{2^g-1}_{\overline{k}}$ of the Kummer variety [30, Theorem 4.8.1]. As a consequence, level-2 theta coordinates represent points up to sign, so differential addition formulae $\pm P, \pm Q, \pm(P-Q) \longmapsto \pm(P+Q)$ are used for arithmetic. Now, if $(A, \lambda_A) \simeq \prod_{m=1}^r (A_m, \lambda_{A_m})$ is a polarised product, then θ^A induces an embedding of the product of Kummer varieties $\prod_{m=1}^r A_m/\pm \hookrightarrow \mathbb{P}^{2^g-1}_{\overline{k}}$ [30, Theorem 4.8.2].

2.4. **Change of theta coordinates.** A PPAV A admits many symmetric theta structures of level n. Such a theta structure $\Theta_A = (\mu_A, \theta^A)$ is fully determined by a ζ-symplectic basis

$(S_1, \cdots, S_g, T_1, \cdots, T_g)$ of $A[2n]$ whose symplectic representation $\nu_A : A[2n] \xrightarrow{\sim} (\mathbb{Z}/2n\mathbb{Z})^g \times \widehat{(\mathbb{Z}/2n\mathbb{Z})^g}$ is such that $\nu_A|_{A[n]} = \mu_A$ [14, Remark 3, p. 319]. If Θ_A and Θ'_A are theta structures of level n respectively induced by (ζ-)symplectic basis $\mathscr{B}$ and $\mathscr{B}'$ of $A[2n]$, and if $\mathbf{M} \in \mathrm{Sp}_{2g}(\mathbb{Z}/2n\mathbb{Z})$ is the symplectic change of coordinates matrix from $\mathscr{B}$ to $\mathscr{B}'$, then we can compute a *change of theta coordinates matrix* $\mathbf{N}_{\mathbf{M},\zeta} \in \mathrm{PGL}_{n^g}(\overline{k})$ such that $({\theta'}^A_{\mathbf{i}})_{\mathbf{i}\in(\mathbb{Z}/2n\mathbb{Z})^g} = \mathbf{N}_{\mathbf{M},\zeta} \cdot (\theta^A_{\mathbf{i}})_{\mathbf{i}\in(\mathbb{Z}/n\mathbb{Z})^g}$ using the formula from [9, Theorem 12]. Applying this formula in level $n = 2$ to the block matrix $\mathbf{M} := [[0, -I_g], [I_g, 0]]$ and $\zeta := \sqrt{-1}$, we obtain that $\mathbf{N}_{\mathbf{M},\zeta}$ is the *Hadamard transform* $H : (x_{\mathbf{i}})_{\mathbf{i}\in(\mathbb{Z}/2\mathbb{Z})^g} \longmapsto \left(\sum_{\mathbf{j}\in(\mathbb{Z}/2\mathbb{Z})^g} (-1)^{\langle \mathbf{i}|\mathbf{j}\rangle} x_{\mathbf{j}}\right)_{\mathbf{i}\in(\mathbb{Z}/2\mathbb{Z})^g}$ inducing the *dual theta coordinates*

$$\forall \mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g, \quad U^A_{\mathbf{i}} = H(\theta^A)_{\mathbf{i}} = \sum_{\mathbf{j}\in(\mathbb{Z}/2\mathbb{Z})^g} (-1)^{\langle \mathbf{i}|\mathbf{j}\rangle} \theta^A_{\mathbf{j}}.$$

Remark 2.2. While we defined for consistency theta structures over $\overline{k}$, we work in practice over the smallest k possible. Following [14, Remark 3, p. 319], if the torsion subgroup $A[2n]$ is k-rational, then the associated level-n theta structure is likewise defined over k, i.e. the theta coordinates of k-rational points of A are also k-rational. In this work, as our PPAVs are isogenous to products of supersingular elliptic curves, our theta structures are defined over $\mathbb{F}_{p^2}$, with the goal of defining them over $\mathbb{F}_p$. Note, however, that the formulae provided herein remain valid over other fields (e.g perfect fields).

2.5. **Computing a 2-isogeny in level 2 theta coordinates.** In practice, we do not represent PPAVs with theta structures by equations but by a single theta point *e.g.* the *theta null-point* $\theta^A(0_A)$. An algorithm to *compute* a 2-isogeny $f : (A, \Theta_A) \to (B, \Theta_B)$ is an algorithm to compute its codomain (B, Θ_B) (*i.e.* a theta point on it like $\theta^B(0_B)$) and an *evaluation* algorithm to then use B (*e.g.* represented by $\theta^B(0_B)$) to evaluate $f(P)$ faster.

Assume that we are given $T_1, \cdots, T_g \in A[8]$ forming an (maximal) *isotropic* subgroup[1] such that $\ker(f) = \langle [4]T_1, \cdots, [4]T_g\rangle$ and a level-2 theta structure Θ_A on A that is *adapted to* f *i.e.* that is induced by $[2]\mathscr{B}$, where $\mathscr{B}$ is a symplectic basis of $A[8]$ of the form $(S_1, \cdots, S_g, T_1, \cdots, T_g)$. If it is not the case, we may use the change of theta coordinates formula from [9, Theorem 12]. Then we can compute B and a level-2 theta structure Θ_B on it that is induced by the symplectic basis $f_*(\mathscr{B}) := ([2]f(S_1), \cdots, [2]f(S_g), f(T_1), \cdots, f(T_g))$ of $B[4]$ [16, § 6.1].

When A is not a product, we say that f is a *generic isogeny* and we can compute the inverse dual theta null-point $U^B(0_B)^{\odot -1}$ from $T_1, \cdots, T_g$. The need for 8-torsion points above $\ker(f)$ comes from the fact that we compute 2-isogenies in level 2. To evaluate f at a point $P \in A(k)$, the procedure is straightforward [16, Algorithm 6.1] and follows the successive operations below:

$$\theta^A(P) \xrightarrow{S} \theta^A(P)^{\odot 2} \xrightarrow{H} H(\theta^A(P)^{\odot 2}) \xrightarrow{\odot U^B(0_B)^{\odot -1}} U^B(f(P)) \xrightarrow{H} \theta^B(f(P)).$$

[1] *i.e.* such that $e_8(T_l, T_m) = 1$ for all $l, m \in [\![1, g]\!]$.

Since level-2 theta structures embed products of Kummer varieties into the projective space, gluing isogenies $f : A_1 \times A_2 \to B$ map points with two (or more) sign ambiguities to points with only one sign ambiguity $(\pm P, \pm Q) \to \pm f(P, Q)$. Hence, they require special evaluation algorithms. Due to vanishing *dual theta constants* $U_{\mathbf{i}}^B(0_B)$, we may use other codomain theta points than the theta null-point to represent gluing isogenies (see Section 4).

2.6. **How to compute a chain of 2-isogenies in theta coordinates.** Let $F : A \to B$ be a 2^e-isogeny between abelian varieties of dimension g and assume we are given $T_1, \cdots, T_g \in A[2^{e+2}]$ forming an isotropic subgroup with $\ker(F) = \langle [4]T_1, \cdots, [4]T_g \rangle$ to compute it. Then F can be decomposed, by [16, Lemma 6.3.1], into a chain of 2-isogenies of length e :

$$A_0 := A \xrightarrow{f_1} A_1 \xrightarrow{f_2} A_2 \quad \cdots \quad A_{e-2} \xrightarrow{f_{e-1}} A_{e-1} \xrightarrow{f_e} A_e := B$$

that can be computed iteratively.

The first step is to compute f_1, that is usually a gluing isogeny $f_1 : A := \prod_{m=1}^r C_m \to A_1$ in our applications. The natural level-2 theta structure on $A = \prod_{m=1}^r C_m$ is a *product theta structure* $\bigotimes_{m=1}^r \Theta_{C_m}$ whose associated theta coordinates satisfy:

$$\left(\otimes_{m=1}^r \theta^{C_m} \right)_{\mathbf{i}}(P_1, \cdots, P_r) = \prod_{m=1}^r \theta_{\mathbf{i}_m}^{C_m}(P_m),$$

for all $(P_1, \cdots, P_r) \in A(k)$ and $\mathbf{i} := (\mathbf{i}_m)_{1 \leq m \leq r} \in \prod_{m=1}^r (\mathbb{Z}/2\mathbb{Z})^{g_m}$ (with $g_m := \dim(C_m)$ for all $m \in [\![1, r]\!]$). If for all $m \in [\![1, r]\!]$, Θ_{C_m} is induced by the (ζ-)symplectic basis $\mathscr{B}_m := (U_{m,1}, \cdots, U_{m,g_m}, V_{m,1}, \cdots, V_{m,g_m})$ of $C_m[4]$, then $\bigotimes_{m=1}^r \Theta_{C_m}$ is induced by the (ζ-)symplectic basis of $A[4]$ given by:

$$\bigotimes_{m=1}^r \mathscr{B}_m := \Big((U_{1,1}, 0, \cdots, 0), \cdots, (U_{1,g_1}, 0, \cdots, 0), \cdots, (0, \cdots, 0, V_{r,1}), \cdots, (0, \cdots, 0, V_{r,g_r}) \Big).$$

In general, $\bigotimes_{m=1}^r \Theta_{C_m}$ is not adapted to f_1 so we need to compute a symplectic change of basis matrix $\mathbf{M} \in \mathrm{Sp}_{2g}(\mathbb{Z}/4\mathbb{Z})$ from $\bigotimes_{m=1}^r \mathscr{B}_m$ to a symplectic basis $\mathscr{B}$ of $A[4]$ of the form $(S_1, \cdots, S_g, [2^e]T_1, \cdots, [2^e]T_g)$, which induces a theta structure Θ_A adapted to f_1. We apply [9, Theorem 12] to obtain the relevant change of theta coordinates matrix $\mathbf{N}_{\mathbf{M},\zeta}$ and compute f_1 from the 8-torsion theta points $\theta^A([2^{e-1}]T_l) = \mathbf{N}_{\mathbf{M},\zeta} \cdot \left(\otimes_{m=1}^r \theta^{C_m}([2^{e-1}]T_l) \right)$ for all $l \in [\![1, g]\!]$.

Then, we expect the $f_i : A_{i-1} \to A_i$ for $i \geq 2$ to be generic isogenies. Once f_{i-1} has been correctly computed, we can prove that the level-2 theta structure Θ_{A_i} on its codomain is automatically adapted to f_i by [16, Lemma 6.3.4], so that no change of theta coordinates is needed and f_i can be computed from the $\theta^{A_i}([2^{e-i}]f_{i-1} \circ \cdots \circ f_1(T_m))$ for all $m \in [\![1, g]\!]$. As for elliptic-curve isogeny computations, divide and conquer strategies apply to perform only $O(ge \log(e))$ point duplications and isogeny evaluations.

Finally, B being often a product in our applications, a change of theta coordinates is performed to recover a product theta structure on B. It can also happen that gluing isogenies appear later on the chain, a case already treated in [9, § 5.1 & Appendices B-C] but not relevant to qt-Pegasis.

3. Improving (generic) isogeny codomain computations

3.1. **The Hypercube Inverse Interpolation Problem (HIIP).** Assume that we want to compute a 2-isogeny $f : (A, \Theta_A) \to (B, \Theta_B)$ between principally polarised abelian varieties of dimension g and that we are given 8-torsion points $T_1, \cdots, T_g$ such that $\ker(f) = \langle [4]T_1, \cdots, [4]T_g \rangle$ expressed in level-2 theta coordinates Θ_A adapted to f. As explained in Section 2.6, we can evaluate f once we have obtained its inverse dual theta null point $U^B(0_B)^{\odot -1}$. To that end, we proceed as in [15, § 4] and [9, § 4.2], by using the following formula [9, Proposition 15]:

$$U^B_{\mathbf{i}}(0_B) \cdot U^B_{\mathbf{i}}(f(T_{\mathbf{j}})) = \sum_{\mathbf{t} \in (\mathbb{Z}/2\mathbb{Z})^g} (-1)^{\langle \mathbf{i}|\mathbf{t} \rangle} \theta^A_{\mathbf{t}}(T_{\mathbf{j}})^2, \tag{2}$$

for all $\mathbf{i}, \mathbf{j} \in (\mathbb{Z}/2\mathbb{Z})^g$, where:

$$T_{\mathbf{j}} = \sum_{m=1}^{g} [j_m] T_m. \tag{3}$$

As level-2 theta coordinates represent points up to sign and satisfy the theta group action relation, Equation (1) implies that $U^B_{\mathbf{i}}(f(T_{\mathbf{j}})) = U^B_{\mathbf{i}}(-f(T_{\mathbf{j}})) = U^B_{\mathbf{i}}(f(T_{\mathbf{j}}) + [2]f(T_{\mathbf{j}})) = U^B_{\mathbf{i}+\mathbf{j}}(f(T_{\mathbf{j}}))$ with the last equality derived from the position of $[2]f(T_{\mathbf{j}})$ in the symplectic basis underlying Θ_B (see [9, Proof of Lemma 16]). Combining this with Equation (2), we obtain the products $\pi_{\mathbf{i},\mathbf{j}} := U^B_{\mathbf{i}}(0_B) \cdot U^B_{\mathbf{i}}(f(T_{\mathbf{j}}))$ and $\pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} := U^B_{\mathbf{i}+\mathbf{j}}(0_B) \cdot U^B_{\mathbf{i}}(f(T_{\mathbf{j}}))$ for all $\mathbf{i}, \mathbf{j} \in (\mathbb{Z}/2\mathbb{Z})^g$. The problem of computing $U^B(0_B)^{\odot -1}$ is therefore to efficiently retrieve $\{U^B_{\mathbf{k}}(0_B)^{\odot -1}\}_{\mathbf{k} \in (\mathbb{Z}/2\mathbb{Z})^g}$, given $\{\pi_{\mathbf{i},\mathbf{j}}\}_{\mathbf{i},\mathbf{j} \in (\mathbb{Z}/2\mathbb{Z})^g}$.

In order to solve this problem, we must see it for what it really is: a graph computational problem. We can indeed see the coordinates $U^B_{\mathbf{i}}(0_B)$ as labels on vertices $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$ of the hypercube (given that $(\mathbb{Z}/2\mathbb{Z})^g$ can be identified with $\{0,1\}^g$). We relate vertices $\mathbf{i}$ and $\mathbf{i}+\mathbf{j}$ if we have computed $\pi_{\mathbf{i},\mathbf{j}}$ and $\pi_{\mathbf{i}+\mathbf{j},\mathbf{j}}$, with neither being zero. We formalise the problem we have to solve as follows. Consider a matrix $(\pi_{\mathbf{i},\mathbf{j}})_{\mathbf{i},\mathbf{j} \in (\mathbb{Z}/2\mathbb{Z})^g}$ and any subset $S \subseteq (\mathbb{Z}/2\mathbb{Z})^g$ chosen for convenience (see Remark 3.3) called the *support* with an associated submatrix $(\pi_{\mathbf{i},\mathbf{j}})_{\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g, \mathbf{j} \in S}$. Consider the graph $G := ((\mathbb{Z}/2\mathbb{Z})^g, E)$ with labelled vertices forming a vector $\mathbf{x} := (x_{\mathbf{i}})_{\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g}$ such that $x_{\mathbf{i}} \cdot \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} = x_{\mathbf{i}+\mathbf{j}} \cdot \pi_{\mathbf{i},\mathbf{j}}$ for all $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$ and $\mathbf{j} \in S$. The edges of G are of the form $\{\mathbf{i}, \mathbf{i}+\mathbf{j}\} \in E$ with $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$ and $\mathbf{j} \in S$ such that $\pi_{\mathbf{i},\mathbf{j}} \neq 0$ and $\pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} \neq 0$. Our goal is to solve the following problem.

Problem 3.1 (Hypercube Inverse Interpolation Problem - HIIP)**.** Given G and $(\pi_{\mathbf{i},\mathbf{j}})_{\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g, \mathbf{j} \in S}$, as above, find an *inverse vector* $\mathbf{x}^{\odot -1} := (x_{\mathbf{i}}^{-1})_{\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g}$ such that

$$\forall \mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g, \mathbf{j} \in S, \quad x_{\mathbf{i}+\mathbf{j}} \cdot \pi_{\mathbf{i},\mathbf{j}} = x_{\mathbf{i}} \cdot \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}}, \tag{4}$$

up to a projective factor in k^*.

As we have seen, in our applications, the existence of a vector $\mathbf{x}$ satisfying Equation (4) is always ensured by the properties of theta coordinates. However, such a vector is not always invertible ($x_{\mathbf{i}} \neq 0$ for all $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$), so does not always yield a solution to the HIIP (Problem 3.1). We expect well-defined solutions to the HIIP to exist for generic isogenies (with rare exceptions [7, Lemma 5]) but not for gluing isogenies due to vanishing dual theta constants $U^B_{\mathbf{i}}(0_B) = 0$. In that case, we may use another dual theta point $U^B(f(T))$, which is invertible, defining a solution to an instance of the HIIP (see Section 4.2). In the following, we assume a solution to the HIIP exists.

3.2. **A general solution of the HIIP.** For our applications, we obviously want to determine a unique solution to the HIIP up to a projective factor (in order to fully determine the inverse dual theta null point $U^B(0_B)^{\odot -1}$). The existence of multiple solutions imply that we do not have enough equations to determine a solution (i.e. the support S is too small). We now show that the uniqueness of the HIIP solution can actually be seen as a simple and natural consequence of the topology of G.

Theorem 3.2 (HIIP Theorem I)**.** *Assume there exists a solution to the HIIP (Problem 3.1). Then it is unique in* $\mathbb{P}^{2^g}(k)$, i.e. *up to a projective factor, if and only if G is connected.*

Proof. $\Longleftarrow$ Assume that G is connected. Let $\mathbf{x}^{\odot -1}, \widetilde{\mathbf{x}}^{\odot -1}$ be two solutions to the HIIP. By Equation (4), we then have for all $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$ and $\mathbf{j} \in S$, $x_{\mathbf{i}}^{-1} \cdot \pi_{\mathbf{i},\mathbf{j}} = x_{\mathbf{i}+\mathbf{j}}^{-1} \cdot \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}}$ and $\widetilde{x}_{\mathbf{i}}^{-1} \cdot \pi_{\mathbf{i},\mathbf{j}} = \widetilde{x}_{\mathbf{i}+\mathbf{j}}^{-1} \cdot \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}}$. If $\{\mathbf{i}, \mathbf{i}+\mathbf{j}\} \in E$, we then have by assumption $\pi_{\mathbf{i},\mathbf{j}} \neq 0$ and $\pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} \neq 0$, so we can divide the first equation by the second and obtain $x_{\mathbf{i}}^{-1} \cdot \widetilde{x}_{\mathbf{i}} = x_{\mathbf{i}+\mathbf{j}}^{-1} \cdot \widetilde{x}_{\mathbf{i}+\mathbf{j}}$. Since G is connected, for all $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$, we can find a path $\mathbf{i}_0 := \mathbf{0}, \mathbf{i}_1, \cdots, \mathbf{i}_r := \mathbf{i}$ for some $r \in \mathbb{N}$, with $\{\mathbf{i}_s, \mathbf{i}_{s+1}\} \in E$ for all $s \in [\![0, r-1]\!]$ and we obtain that $x_{\mathbf{0}}^{-1} \cdot \widetilde{x}_{\mathbf{0}} = x_{\mathbf{i}_1}^{-1} \cdot \widetilde{x}_{\mathbf{i}_1} = \cdots = x_{\mathbf{i}_r}^{-1} \cdot \widetilde{x}_{\mathbf{i}_r} = x_{\mathbf{i}}^{-1} \cdot \widetilde{x}_{\mathbf{i}}$, so that $\mathbf{x}^{\odot -1} = \lambda \cdot \widetilde{\mathbf{x}}^{\odot -1}$ with $\lambda := x_{\mathbf{0}}^{-1} \cdot \widetilde{x}_{\mathbf{0}} \in k^*$. This proves the uniqueness of the HIIP solution in $\mathbb{P}^{2^g}(k)$.

$\Longrightarrow$ Now, assume by contraposition that G is not connected. Let $\mathbf{x}^{\odot -1}$ be a solution to the HIIP and let $V_0 \subsetneq (\mathbb{Z}/2\mathbb{Z})^g$ be the connected component of $\mathbf{0}$ in the graph G. Let $\lambda \in k \setminus \{0, 1\}$ (which does exist since $\mathrm{char}(k) \neq 2$) and define $\widetilde{\mathbf{x}}^{\odot -1}$ by $\widetilde{x}_{\mathbf{i}}^{-1} := x_{\mathbf{i}}^{-1}$ for all $\mathbf{i} \in V_0$ and $\widetilde{x}_{\mathbf{i}}^{-1} := \lambda \cdot x_{\mathbf{i}}^{-1}$, for all $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g \setminus V_0$. Let $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$ and $\mathbf{j} \in S$. Then by Equation (4), we have $x_{\mathbf{i}}^{-1} \cdot \pi_{\mathbf{i},\mathbf{j}} = x_{\mathbf{i}+\mathbf{j}}^{-1} \cdot \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}}$. Since V_0 and $(\mathbb{Z}/2\mathbb{Z})^g \setminus V_0$ are not connected, for all $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$, $\mathbf{j} \in S$, we either have $\mathbf{i}, \mathbf{i}+\mathbf{j} \in V_0$ so $\widetilde{x}_{\mathbf{i}}^{-1} = x_{\mathbf{i}}^{-1}$ and $\widetilde{x}_{\mathbf{i}+\mathbf{j}}^{-1} = x_{\mathbf{i}+\mathbf{j}}^{-1}$ or $\mathbf{i}, \mathbf{i}+\mathbf{j} \notin V_0$ so $\widetilde{x}_{\mathbf{i}}^{-1} = \lambda x_{\mathbf{i}}^{-1}$ and $\widetilde{x}_{\mathbf{i}+\mathbf{j}}^{-1} = \lambda x_{\mathbf{i}+\mathbf{j}}^{-1}$, or $\{\mathbf{i}, \mathbf{i}+\mathbf{j}\} \notin E$ and $\pi_{\mathbf{i},\mathbf{j}} = 0$ or $\pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} = 0$. In the last case, we have $\pi_{\mathbf{i},\mathbf{j}} = \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} = 0$ by Equation (4) and because $x_{\mathbf{i}}$ and $x_{\mathbf{i}+\mathbf{j}}$ are non-zero. It follows that $\widetilde{x}_{\mathbf{i}}^{-1} \cdot \pi_{\mathbf{i},\mathbf{j}} = \widetilde{x}_{\mathbf{i}+\mathbf{j}}^{-1} \cdot \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}}$ in all of these cases. This proves that $\widetilde{\mathbf{x}}^{\odot -1}$ is a solution to the HIIP but distinct from $\mathbf{x}^{\odot -1}$ in $\mathbb{P}^{2^g}(k)$, which completes the proof. □

Remark 3.3 (On the choice of support)**.** We usually choose the support S of a HIIP instance as small and convenient as possible for the associated graph to be connected. For generic isogeny computations, we experimentally expect the following choice to be sufficient. We take $S := \{\mathbf{e}_1, \cdots, \mathbf{e}_g\}$,

where for all $l, m \in \{1, \cdots, g\}$ given by $e_{l,m} = 1$ if $l = m$ and 0 otherwise. Hence, G is exactly the hypercube graph (whose edges relate vertices separated by vectors $\mathbf{j}$ of Hamming weight 1). This way, we only use the input theta coordinates $\theta^A(T_1), \cdots, \theta^A(T_g)$ and don't need to compute sums of points. However, in some cases, we may be required to take a bigger support S to compute a solution to the HIIP due to multiple zero entries $\pi_{\mathbf{i},\mathbf{j}}$ (e.g. for non-generic isogeny computations). In those cases, unlike the hypercube graph, G may contain edges separated by vectors $\mathbf{j} \in S$ of Hamming weight greater than 1.

Note that when G is connected and thus a unique solution exists, Theorem 3.2 does not provide a formula to compute it. Leveraging the projective nature of the solution, we propose an inversion free formula below. Indeed, inversions should be avoided, even at the expense of additional multiplications, given their very high cost in practice. Let $\mathcal{T} = ((\mathbb{Z}/2\mathbb{Z})^g, E_{\mathcal{T}})$ be a spanning tree of G. For any edge $e \in E_{\mathcal{T}}$ with $e = \{\mathbf{a}, \mathbf{b}\}$, we define $\mathbf{j}_e \in S \subseteq (\mathbb{Z}/2\mathbb{Z})^g$ by $\mathbf{j}_e := \mathbf{b} - \mathbf{a}$. For any two distinct vertices $\mathbf{i}_1 \neq \mathbf{i}_2 \in (\mathbb{Z}/2\mathbb{Z})^g$, there exists a unique path in $\mathcal{T}$ starting at $\mathbf{i}_1$ and ending at $\mathbf{i}_2$. Fixing $\mathbf{i}_1$, this induces a bijection $\delta_{\mathbf{i}_1} : (\mathbb{Z}/2\mathbb{Z})^g \backslash \{\mathbf{i}_1\} \xrightarrow{\sim} E_{\mathcal{T}}$, where $\delta_{\mathbf{i}_1}(\mathbf{i}_2)$ is defined to be the last edge along the path from $\mathbf{i}_1$ to $\mathbf{i}_2$. Observe that $\delta_{\mathbf{i}_1}(\mathbf{i}_2)$ is always an edge incident to $\mathbf{i}_2$. Using the maps $\{\delta_{\mathbf{k}}\}_{\mathbf{k} \in (\mathbb{Z}/2\mathbb{Z})^g}$, we can explicitly define formulae for $\mathbf{x}^{\odot -1}$ as simply:

$$x_{\mathbf{k}}^{-1} = \prod_{\mathbf{i} \neq \mathbf{k}} \pi_{\mathbf{i},\mathbf{j}_{\delta_{\mathbf{k}}(\mathbf{i})}} = \prod_{e \in E_{\mathcal{T}}} \pi_{\delta_{\mathbf{k}}^{-1}(e),\mathbf{j}_e}. \tag{5}$$

Theorem 3.4 (HIIP Theorem II)**.** *Equation* (5) *defines a solution* $\mathbf{x}^{\odot -1}$ *to the HIIP if such a solution exists.*

Proof. Given two adjacent vertices $\mathbf{i}$ and $\mathbf{i}+\mathbf{j}$ in $\mathcal{T}$, we have $\delta_{\mathbf{i}+\mathbf{j}}(\mathbf{k}) = \delta_{\mathbf{i}}(\mathbf{k})$ for all $\mathbf{k} \in (\mathbb{Z}/2\mathbb{Z})^g \setminus \{\mathbf{i}, \mathbf{i}+\mathbf{j}\}$ and $\mathbf{j}_{\delta_{\mathbf{i}}(\mathbf{i}+\mathbf{j})} = \mathbf{j}_{\delta_{\mathbf{i}+\mathbf{j}}(\mathbf{i})} = \mathbf{j}$. Thus, for all $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g, \mathbf{j} \in S$, we have

$$\begin{aligned} x_{\mathbf{i}}^{-1} \cdot \pi_{\mathbf{i},\mathbf{j}} = \pi_{\mathbf{i},\mathbf{j}} \cdot \prod_{\mathbf{k} \neq \mathbf{i}} \pi_{\mathbf{k},\mathbf{j}_{\delta_{\mathbf{i}}(\mathbf{k})}} &= \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}_{\delta_{\mathbf{i}}(\mathbf{i}+\mathbf{j})}} \cdot \pi_{\mathbf{i},\mathbf{j}} \cdot \prod_{\mathbf{k} \neq \mathbf{i},\mathbf{i}+\mathbf{j}} \pi_{\mathbf{k},\mathbf{j}_{\delta_{\mathbf{i}}(\mathbf{k})}} = \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} \cdot \pi_{\mathbf{i},\mathbf{j}} \cdot \prod_{\mathbf{k} \neq \mathbf{i},\mathbf{i}+\mathbf{j}} \pi_{\mathbf{k},\mathbf{j}_{\delta_{\mathbf{i}}(\mathbf{k})}} \\ &= \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} \cdot \pi_{\mathbf{i},\mathbf{j}_{\delta_{\mathbf{i}+\mathbf{j}}(\mathbf{i})}} \cdot \prod_{\mathbf{k} \neq \mathbf{i},\mathbf{i}+\mathbf{j}} \pi_{\mathbf{k},\mathbf{j}_{\delta_{\mathbf{i}+\mathbf{j}}(\mathbf{k})}} = \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} \cdot \prod_{\mathbf{k} \neq \mathbf{i}+\mathbf{j}} \pi_{\mathbf{k},\mathbf{j}_{\delta_{\mathbf{i}+\mathbf{j}}(\mathbf{k})}} = \pi_{\mathbf{i}+\mathbf{j},\mathbf{j}} \cdot x_{\mathbf{i}+\mathbf{j}}^{-1}. \end{aligned}$$

It remains to prove Equation (4) when $\{\mathbf{i}, \mathbf{i}+\mathbf{j}\} \in E \setminus E_{\mathcal{T}}$. Let $\widetilde{\mathbf{x}}^{\odot -1}$ be a solution to the HIIP (which does exist by assumption). Then $\widetilde{\mathbf{x}}^{\odot -1}$ satisfies Equation (4) so it satisfies this equation in particular for all $\{\mathbf{i}, \mathbf{i}+\mathbf{j}\} \in E_{\mathcal{T}}$. Since $\mathcal{T}$ is connected, the proof of Theorem 3.2 ($\Longleftarrow$) ensures that $\mathbf{x}^{\odot -1} = \lambda \cdot \widetilde{\mathbf{x}}^{\odot -1}$ for some $\lambda \in k^*$. It follows that $\mathbf{x}^{\odot -1}$ also satisfies Equation (4) and is a solution to the HIIP. □

A naive evaluation of Equation (5) yields a solution to the HIIP with a cost of $2^g(2^g - 2)\mathbf{M}$. To reduce this complexity, it is thus necessary to specialise our general formulae on spanning trees of G with favorable computational properties.

3.3. **Fast formulae.** To accelerate Equation (5), we look for a *Hamiltonian path* in G. By definition, such a path is given by a bijection $\eta : [\![0, 2^g - 1]\!] \xrightarrow{\sim} (\mathbb{Z}/2\mathbb{Z})^g$ such that $(\eta(i), \eta(i+1)) \in E$ for all $i \in [\![0, 2^g - 2]\!]$.

$$\begin{array}{ccccccc} x_{\eta(0)} & & x_{\eta(1)} & & x_{\eta(2^g-2)} & & x_{\eta(2^g-1)} \\ \eta(0) & \longrightarrow & \eta(1) & \cdots & \eta(2^g-2) & \longrightarrow & \eta(2^g-1). \end{array}$$

From such a path, we can significantly reduce the computation cost of Equation (5). Remark that a Hamiltonian path defines a spanning tree and that

$$\mathbf{j}_{\delta_{\eta(j)}(\eta(i))} = \begin{cases} \mathbf{s}_i & \text{if } i < j \\ \mathbf{s}_{i-1} & \text{if } i > j \end{cases} \quad \text{with } \mathbf{s}_i := \eta(i+1) - \eta(i) \in S \text{ for all } i \in [\![0, 2^g - 1]\!],$$

with the notations introduced above Equation (5). We can therefore write $x_{\eta(i)}^{-1}$ for $i \in [\![0, 2^g - 1]\!]$ as Equation (6) below.

$$x_{\eta(i)}^{-1} = \left(\prod_{j=0}^{i-1} \pi_{\eta(j),\mathbf{s}_j} \right) \cdot \left(\prod_{j=i}^{2^g-2} \pi_{\eta(j+1),\mathbf{s}_j} \right) = \rho_L(i) \cdot \rho_R(i) \tag{6}$$

We see that both ρ_L and ρ_R have a simple iterative structure:

(7)

$$\rho_L(i) = \begin{cases} 1 & \text{if } i = 0 \\ \rho_L(i-1) \cdot \pi_{\eta(i-1),s_{i-1}} & \text{if } i > 0 \end{cases} \text{ and } \rho_R(i) = \begin{cases} 1 & \text{if } i = 2^g - 1 \\ \rho_R(i+1) \cdot \pi_{\eta(i+1),s_i} & \text{if } i < 2^g - 1 \end{cases}$$

Our strategy to compute the HIIP on Hamiltonian paths is therefore to first compute ρ_L and ρ_R iteratively and then reassemble them in order to compute $\mathbf{x}^{\odot -1}$. This is summarised in Algorithm 1 whose computational cost is $3(2^g - 2)\mathbf{M}$. As previous methods for solving the HIIP [9, § 4.2] and [16, § 6.1] have a cost of at least $(6 \cdot 2^g - 9)\mathbf{M}$, we see that our algorithm is approximately two times faster. Moreover, the former approaches are not constant time, and there is no obvious way to make them constant time or to mask them with low overhead. In contrast, our formulae are naturally constant time and require only minimal control logic. For instance, in dimension $g = 4$, which is our main focus, for the same input data, the previous formulae costs $87\,\mathbf{M}$, whereas Algorithm 1 requires only $42\mathbf{M}$.

3.4. **Finding a Hamiltonian path.** Before applying Algorithm 1 to solve the HIIP, one needs to find a Hamiltonian path in the graph G. The general idea is to consider a Hamiltonian path $\eta_0 : [\![0, 2^g - 1]\!] \xrightarrow{\sim} (\mathbb{Z}/2\mathbb{Z})^g$ of K_S, the *complete graph with support* S with $(\mathbb{Z}/2\mathbb{Z})^g$ as set of vertices and $E_S := \{(\mathbf{i}, \mathbf{i} + \mathbf{j}) \mid \mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g, \mathbf{j} \in S\}$ as set of edges. By construction G is a subgraph of K_S but these graphs may not be equal ($E \subsetneq E_S$) because of zero values of $\pi_{\mathbf{i},\mathbf{j}}$. Hence, η_0 may be a

Algorithm 1: Solution to the HIIP

Data: HIIP input data, including a subset $S \subseteq (\mathbb{Z}/2\mathbb{Z})^g$, a graph $G := ((\mathbb{Z}/2\mathbb{Z})^g, E)$ and matrix $(\pi_{\mathbf{i},\mathbf{j}})_{\mathbf{i}\in(\mathbb{Z}/2\mathbb{Z})^g,\mathbf{j}\in S}$. A Hamiltonian path $\eta : [\![0, 2^g - 1]\!] \xrightarrow{\sim} (\mathbb{Z}/2\mathbb{Z})^g$ in G.

Result: A solution $(x^{-1}_{\eta(i)})_{0\leq i\leq 2^g-1}$ to the HIIP defined by the input data (up to reordering by η).

$\rho_L(1) \leftarrow \pi_{\eta(0),\mathbf{s}_0}$;
$\rho_R(2^g - 2) \leftarrow \pi_{\eta(2^g-1),\mathbf{s}_{2^g-2}}$;
for $i \in [\![2, 2^g - 1]\!]$ **do**
  $\rho_L(i) \leftarrow \rho_L(i-1) \cdot \pi_{\eta(i-1),\mathbf{s}_{i-1}}$;
  $\rho_R(2^g - 1 - i) \leftarrow \rho_R(2^g - i) \cdot \pi_{\eta(2^g-i),\mathbf{s}_{2^g-1-i}}$;
end
$x^{-1}_{\eta(0)} \leftarrow \rho_R(0)$;
$x^{-1}_{\eta(2^g-1)} \leftarrow \rho_L(2^g - 1)$;
for $i \in [\![1, 2^g - 2]\!]$ **do**
  $x^{-1}_{\eta(i)} \leftarrow \rho_L(i) \cdot \rho_R(i)$;
end
return $\left(x^{-1}_{\eta(i)}\right)_{0\leq i\leq 2^g-1}$; // Total cost: $(3 \cdot 2^g - 6)\mathbf{M}$

Hamiltonian path of K_S but not of G. We may then post compose η_0 with automorphisms of K_S until we obtain a Hamiltonian path of G.

We have seen in Remark 3.3 that for our applications, S always contains $\{\mathbf{e}_1, \cdots, \mathbf{e}_g\}$ so K_S contains the Hypercube (whose edges have Hamming weight 1). We may then always take as starting path η_0, a Hamiltonian path of the Hypercube. In practice, we consider the *Gray Hamiltonian cycle* which is easy to compute with bitwise operations:

$$\eta_0 : i := \sum_{m=0}^{g-1} i_m 2^m \in [\![0, 2^g - 1]\!] \longmapsto (i_0 \oplus i_1, \cdots, i_{g-2} \oplus i_{g-1}, i_{g-1}) \in (\mathbb{Z}/2\mathbb{Z})^g.$$

When $S = \{\mathbf{e}_1, \cdots, \mathbf{e}_g\}$, we may then try to find a Hamiltonian path of the form $\eta := \Delta \circ \eta_0$, where Δ is an automorphism of the Hypercube.

Lemma 3.5. [31, Corollary 14] *All automorphisms of the Hypercube are of the form* $\Delta_{\sigma,\mathbf{v}} : \mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^g \longmapsto \mathbf{i} \circ \sigma + \mathbf{v} \in (\mathbb{Z}/2\mathbb{Z})^g$ *with* $\mathbf{i} \circ \sigma := (i_{\sigma(m)})_{1\leq m\leq g}$, $\sigma \in \mathfrak{S}(g)$ *and* $\mathbf{v} \in (\mathbb{Z}/2\mathbb{Z})^g$.

Hence, to find a Hamiltonian path from η_0, we try the $\Delta_{\sigma,\mathbf{v}} \circ \eta_0$ for all $\sigma \in \mathfrak{S}(g)$ and $\mathbf{v} \in (\mathbb{Z}/2\mathbb{Z})^g$. This amounts to $2^g g!$ Hypercube automorphisms to try (*e.g.* 384 in dimension $g = 4$).

Remark 3.6. In the case $S = \{\mathbf{e}_1, \cdots, \mathbf{e}_g\}$ and in low dimension (*e.g.* $g \leq 4$), we have such a limited amount of graph automorphisms to try ($2^g g!$) that we can actually try them all for every isogeny codomain computation. Since Algorithm 1 is constant time by design, we obtain a constant

time solution to the HIIP with low computational overhead. In practice, we can restrict ourselves to just a handful of automorphisms of the hypercube (*e.g.* the $\Delta_{\sigma,\mathbf{0}}$ for $\sigma \in \mathfrak{S}(g)$).

Remark 3.7. Even though $S = \{\mathbf{e}_1, \cdots, \mathbf{e}_g\}$ is a big enough support for a Hamiltonian path to exist for generic isogeny computations, this may not always be the case. In these cases, we can either add vectors to S (*e.g.* of Hamming weight 2) or solve the HIIP with other subgraphs than Hamiltonian paths with a higher cost than Algorithm 1 (or do both). Note that when S grows, we may also consider more graph automorphisms than in Lemma 3.5. This is a trade-off between sums of points to compute (following Equation (3)) and graph automorphisms to consider on the one hand, and higher cost of the HIIP solution on the other hand.

In qt-Pegasis, the second 2-isogeny is a generic isogeny, however, a fifth vector in the support S (either $\mathbf{e}_1 + \mathbf{e}_2$ or $\mathbf{e}_3 + \mathbf{e}_4$) is sometimes necessary (see [7, § 4.3, p. 24]), together with another Hamiltonian path not derived from the Gray-cycle, in order to find a valid Hamiltonian path and solve the relevant HIIP in dimension $g = 4$. We detail this case in Appendix D.

We shall see in Section 5.3 that computing a gluing isogeny from a product of 4 elliptic curves also requires to solve a HIIP in dimension $g = 4$ with S containing 5 elements while also considering non-Hamiltonian connected graphs with a *Rabbit* structure (as defined in Definition 5.2).

4. Superglue in 4D: improving gluing formulae on a product of 4 elliptic curves

This section is motivated by the computation of gluing 2-isogenies starting from products of four elliptic curves which appear in qt-Pegasis. Even though we work in a more general setting, part of this section will be specific to qt-Pegasis. We generalise to dimension 4 the approach introduced by Max Duparc in Superglue [10] for 2-dimensional gluing 2-isogeny computations, aiming at efficient formulae easy to implement in constant time.

Throughout this section, we fix a gluing 2-isogeny $f : A \to B$, where $A := \prod_{m=1}^{4} E_m$ is a product of Montgomery elliptic curves and B is a 4-dimensional non-product principally polarised abelian variety. Let $\mathscr{B} := (S_1, \cdots, S_4, T_1, \cdots, T_4)$ be a symplectic basis of $A[8]$ adapted to f (so that $\ker(f) = \langle [4]T_1, \cdots, [4]T_4 \rangle$). Let Θ_A be the level-2 theta structure on A induced by $[2]\mathscr{B}$ and Θ_B be the level-2 theta structure on B induced by $f_*(\mathscr{B}) = ([2]f(S_1), \cdots, [2]f(S_4), f(T_1), \cdots, f(T_4))$.

4.1. Evaluation algorithm using barycentric coordinates.

The approach proposed in [10] to evaluate f at a point $P \in A$ is to use a fixed auxiliary point $T \in A$ and Mumford's duplication formula [14] (see also [16, Lemma 6.1.12]):

$$U^B(f(P)) \odot U^B(f(T)) = H\left(\theta^A(P+T) \odot \theta^A(P-T)\right), \tag{8}$$

In practice (*e.g.* in qt-Pegasis), the auxiliary point is a 16-torsion point lying over $\ker(f)$. We shall explain in Section 4.2 how to compute the needed inverse dual image theta point $U^B(f(T))^{\odot -1}$ by solving an instance of the HIIP.

We now explain how to compute $\theta^A(P+T)\odot\theta^A(P-T)$. Let $m\in\{1,\cdots,4\}$ and $(x_m:y_m:z_m)$ be the projective coordinates of E_m. Assume we are given for all $m\in\{1,\cdots,4\}$ a level-2 theta structure Θ_{E_m} on E_m induced by a basis $\mathscr{B}_m:=(V_m,W_m)$ of $E_m[4]$ with $x_m(W_m)=-z_m(W_m)$, so that associated theta coordinates are, following [16, Proposition 5.3.47], of the form

$$\theta^{E_m}(x_m:y_m:z_m)=(a_m(x_m-z_m):b_m(x_m+z_m)), \tag{9}$$

and with $(a_m:b_m):=(x_m(V_m)+z_m(V_m):x_m(V_m)-z_m(V_m))$ the theta null point.

Lemma 4.1. *The product theta coordinates of a point $X:=((x_m:y_m:z_m))_{1\leq m\leq 4}\in A$ are given by:*

$$\otimes_{m=1}^4\theta^{E_m}(X)=\left(\otimes_{m=1}^4(a_m,b_m)\right)\odot\left(\overline{H}(\otimes_{m=1}^4(x_m,z_m))\right)$$

where $\overline{H}:(x_{\mathbf{i}})_{\mathbf{i}\in(\mathbb{Z}/2\mathbb{Z})^4}\longmapsto\left(\sum_{\mathbf{j}\in(\mathbb{Z}/2\mathbb{Z})^4}(-1)^{\langle\mathbf{1}-\mathbf{i}|\mathbf{j}\rangle}x_{\mathbf{j}}\right)_{\mathbf{i}\in(\mathbb{Z}/2\mathbb{Z})^4}$ (with $\mathbf{1}:=(1,1,1,1)$) is the bit-flipped Hadamard transform.

Proof. Let us write $x_{0,m}:=x_m$ and $x_{1,m}:=z_m$ for all $m\in\{1,\cdots,4\}$. Then Equation (9) ensures that

$$\begin{aligned}
\otimes_{m=1}^4\theta^{E_m}(X)&=\otimes_{m=1}^4(a_m(x_m-z_m):b_m(x_m+z_m))\\
&=\left(\otimes_{m=1}^4(a_m,b_m)\right)\odot\left(\otimes_{m=1}^4(x_m-z_m:x_m+z_m)\right)\\
&=\left(\otimes_{m=1}^4(a_m,b_m)\right)\odot\left(\prod_{m=1}^4(x_{0,m}+(-1)^{1-i_m}x_{1,m})\right)_{\mathbf{i}\in(\mathbb{Z}/2\mathbb{Z})^4}\\
&=\left(\otimes_{m=1}^4(a_m,b_m)\right)\odot\left(\sum_{\mathbf{j}\in(\mathbb{Z}/2\mathbb{Z})^4}\prod_{m=1}^4(-1)^{j_m(1-i_m)}x_{j_m,m}\right)_{\mathbf{i}\in(\mathbb{Z}/2\mathbb{Z})^4}\\
&=\left(\otimes_{m=1}^4(a_m,b_m)\right)\odot\overline{H}(\otimes_{m=1}^4(x_m,z_m)),
\end{aligned}$$

as desired. □

Let $\mathbf{N}_1$ be the change of coordinates matrix from product $(x:z)$ coordinates to product theta coordinates $\bigotimes_{m=1}^4\theta^{E_m}$ on A. By Lemma 4.1, it acts on vectors as $\mathbf{x}\longmapsto\left(\otimes_{m=1}^4(a_m,b_m)\right)\odot\overline{H}(\mathbf{x})$. Let $\mathbf{N}_2$ be the change of theta coordinates matrix from the product theta structure $\bigotimes_{m=1}^4\Theta_{E_m}$ to Θ_A. By [9, Theorem 12], $\mathbf{N}_2$ can be computed from the symplectic change of basis from $\bigotimes_{m=1}^4\mathscr{B}_m$ to $[2]\mathscr{B}$. Let $\mathbf{N}:=\mathbf{N}_2\cdot\mathbf{N}_1$. Then, we have:

$$\begin{aligned}
\theta^A(P+T)\odot\theta^A(P-T)=&\left(\mathbf{N}\cdot\otimes_{m=1}^4\left(x_m(P_m+T^{(m)}),z_m(P_m+T^{(m)})\right)\right)\\
&\odot\left(\mathbf{N}\cdot\otimes_{m=1}^4\left(x_m(P_m-T^{(m)}),z_m(P_m-T^{(m)})\right)\right),
\end{aligned} \tag{10}$$

with component-wise decompositions $P:=(P_1,\cdots,P_4)$ and $T:=(T^{(1)},\cdots,T^{(4)})$. To compute $\theta^A(P+T)\odot\theta^A(P-T)$, we could use Equation (10) directly. However, [10] already proposes an alternate method in dimension 2 exploiting the symmetries of the formula, also making it easier to

use points on the quadratic twist of A in order to work on a smaller base field k. The idea is to express $P_m + T^{(m)}$ and $P_m - T^{(m)}$ simultaneously with *barycentric coordinates*, whose definition is recalled below.

Definition 4.2. [10, Definition 6 and Proposition 2] Let E be a Montgomery elliptic curve of equation $y^2 = x^3 + \alpha x^2 + x$ and $E^t : \beta y^2 = x^3 + \alpha x^2 + x$ be a quadratic twist, where $\alpha \in k$ and $\beta \in k^*$ is not a square. Let $P, Q \in E(k) \cup E^t(k)$ such that $P \neq \pm Q$. The *barycentric coordinates* of (P, Q) form a projective point $(u : v : w) \in \mathbb{P}^2(k)$ such that $P+Q = (x_\oplus : z_\oplus)$ and $P-Q = (x_\ominus : z_\ominus)$ on the Kummer line $E/\pm$, where:

$$x_\oplus = u - \delta_P \delta_Q v, \quad x_\ominus = u + \delta_P \delta_Q v \quad \text{and} \quad z_\oplus = z_\ominus = w,$$

$\delta_P := 1$ if $P \in E$, $\delta_P := \sqrt{\beta}$ otherwise and δ_Q is defined similarly.

We can apply [10, Algorithm 2] to compute barycentric coordinates from $P \neq \pm Q$. However, we also need to treat the case $P = \pm Q$ in our applications. We slightly modify the previous definition to that end. In the case $P = \pm Q$, the v coefficient sees its position swapped, meaning that

$$x_\oplus = x_\ominus = u, \quad z_\oplus = w - \delta_P \delta_Q v \quad \text{and} \quad z_\ominus = w + \delta_P \delta_Q v.$$

In Appendix E, we adapt [10, Algorithm 2] into a general algorithm compatible with both twists and zero points. For all $m \in \{1, \cdots, 4\}$, we can then compute the barycentric coordinates $(u_m : v_m : w_m)$ of $(P_m, T^{(m)})$ with $14\mathbf{M} + 5\mathbf{S} + 2\mathbf{m} + 13\mathbf{a}$.

Remark 4.3. In order to simplify already heavy notations, we assume in the following that $P_m \neq \pm T_m$ for all $m \in \{1, \cdots, 4\}$ and illustrate the general case in Appendix E.

Lemma 4.4. *Assume that either $P_m \in E_m$ for all $m \in [\![1, 4]\!]$ or $P_m \in E_m^t$ for all $m \in [\![1, 4]\!]$, and similarly for the $T^{(m)}$. Also assume that $P_m \neq \pm T^{(m)}$ for all $m \in [\![1, 4]\!]$. Let $\beta \in k^*$ be a common twisting coefficient of $E_m^t : \beta y^2 = x^3 + \alpha_m x^2 + x$ for all $m \in \{1, \cdots, 4\}$. Then*

$$\theta^A(P + T) \odot \theta^A(P - T) = (\mathbf{N} \cdot \mathbf{u})^{\odot 2} - \Delta_P \Delta_T (\mathbf{N} \cdot \mathbf{v})^{\odot 2},$$

where

$$\Delta_P := \begin{cases} 1 & \text{if } \forall m \in [\![1, 4]\!],\ P_m \in E_m \\ \beta & \text{if } \forall m \in [\![1, 4]\!],\ P_m \in E_m^t \end{cases}, \quad \Delta_T := \begin{cases} 1 & \text{if } \forall m \in [\![1, 4]\!],\ T^{(m)} \in E_m \\ \beta & \text{if } \forall m \in [\![1, 4]\!],\ T^{(m)} \in E_m^t \end{cases},$$

$$\mathbf{u} := \mathbf{u}_{12} \otimes \mathbf{u}_{34} + \Delta_P \Delta_T \mathbf{v}_{12} \otimes \mathbf{v}_{34}, \quad \mathbf{v} := \mathbf{u}_{12} \otimes \mathbf{v}_{34} + \mathbf{v}_{12} \otimes \mathbf{u}_{34},$$

and for $i, j \in [\![1, 4]\!]$,

$$\mathbf{u}_{ij} := \begin{pmatrix} u_i u_j + \Delta_P \Delta_T v_i v_j \\ w_i u_j \\ u_i w_j \\ w_i w_j \end{pmatrix} \quad \text{and} \quad \mathbf{v}_{ij} := \begin{pmatrix} u_j v_i + u_i v_j \\ w_i v_j \\ v_i w_j \\ 0 \end{pmatrix}. \tag{11}$$

Proof. Since $(u_m : v_m : w_m)$ are the barycentric coordinates of $(P_m, T^{(m)})$, we have

$$\begin{aligned}
\otimes_{m=1}^4 \left(x_m(P_m \pm T^{(m)}), z_m(P_m \pm T^{(m)})\right) &= \otimes_{m=1}^4 \left((u_m, w_m) \mp \delta_{P_m}\delta_{T^{(m)}}(v_m, 0)\right) \\
&= \otimes_{m=1}^4 \left((u_m, w_m) \mp \sqrt{\Delta_P\Delta_T}(v_m, 0)\right) \\
&= \left(\mathbf{u}_{12} \mp \sqrt{\Delta_P\Delta_T}\mathbf{v}_{12}\right) \otimes \left(\mathbf{u}_{34} \mp \sqrt{\Delta_P\Delta_T}\mathbf{v}_{34}\right) \\
&= \mathbf{u}_{12} \otimes \mathbf{u}_{34} + \Delta_P\Delta_T\mathbf{v}_{12} \otimes \mathbf{v}_{34} \\
&\quad \mp \sqrt{\Delta_P\Delta_T}\left(\mathbf{u}_{12} \otimes \mathbf{v}_{34} + \mathbf{v}_{12} \otimes \mathbf{u}_{34}\right) \\
&= \mathbf{u} \mp \sqrt{\Delta_P\Delta_T}\mathbf{v}
\end{aligned}$$

Then, we get by Equation (10):

$$\begin{aligned}
\theta^A(P+T) \odot \theta^A(P-T) &= (\mathbf{N} \cdot (\mathbf{u} - \sqrt{\Delta_P\Delta_T}\mathbf{v})) \odot (\mathbf{N} \cdot (\mathbf{u} + \sqrt{\Delta_P\Delta_T}\mathbf{v})) \\
&= (\mathbf{N} \cdot \mathbf{u})^{\odot 2} - \Delta_P\Delta_T(\mathbf{N} \cdot \mathbf{v})^{\odot 2}
\end{aligned}$$

This completes the proof. □

We summarise the new evaluation algorithm we propose in Algorithm 2, with a detailed account of its computational cost.

Remark 4.5. Let us compare Algorithm 2 with the inversion free version [16, Algorithm 6.6] of the state of the art algorithm for gluing isogeny evaluation introduced in [9, Algorithm 5]. In order to evaluate $P \in A$, [16, Algorithm 6.6] uses n auxiliary points $T' \in A[4]$ such that $[2]T' \in \ker(f)$ and claims a computational cost of $(6n+29)\mathbf{M}+16(n+1)\mathbf{S}+64(n+2)\mathbf{a}$ in dimension 4 when $\theta^A(P)$ and the $\theta^A(P+T')$ are given. To obtain a fair comparison, we account for the cost when P and the T' are given on entry as tuples of elliptic curve points. Using Jacobian coordinates, the computation of the $P+T'$ costs $4n(17\mathbf{M}+6\mathbf{S}+13\mathbf{a})$ [20, Algorithm 8.12]. We can then compute $\otimes_{m=1}^4(x_m(P), z_m(P))$ in $24\mathbf{M}$ by computing $\otimes_{m=1}^2(x_m(P), z_m(P))$ and $\otimes_{m=3}^4(x_m(P), z_m(P))$ (in $2 \times 4\mathbf{M}$), and then their tensor product (in $16\mathbf{M}$). To obtain $\theta^A(P)$, we can then apply the twisted Hadamard $\overline{H}$ to $\otimes_{m=1}^4(x_m(P), z_m(P))$, multiply by $\otimes_{m=1}^4(a_m, b_m)$ and then by $\mathbf{N}_2$, which costs $16\mathbf{M}+256\mathbf{m}+304\mathbf{a}$. The same can be done to obtain the $\theta^A(P+T')$ from $P+T'$.

On the whole, the total cost of this gluing evaluation method is $(114n+69)\mathbf{M} + (40n+16)\mathbf{S} + 256(n+1)\mathbf{m} + (420n+432)\mathbf{a}$. In practice (*e.g.* in qt-Pegasis), we expect to need at least $n = 2$ auxiliary points T', so the state of the art cost is at least $281\mathbf{M} + 96\mathbf{S} + 768\mathbf{m} + 1272\mathbf{a}$, to be compared with $171\mathbf{M} + 52\mathbf{S} + 541\mathbf{m} + 829\mathbf{a}$.

In addition to this clear computational cost gain (by almost a factor 2, at least), Algorithm 2 is equipped to use points on the twist, allowing further cost gains by working over a smaller base field k. Also note that unlike [16, Algorithm 6.6], Algorithm 2 does not contain conditional branching[2], which facilitates constant time implementations.

[2]Aside from the computation of Δ_P and Δ_T which trivially depend on input points.

Remark 4.6. By [9, Theorem 12], the coefficients of the change of theta coordinates matrix $\mathbf{N}_2$ are sums of fourth roots of unity, so we expect multiplications by these coefficients to be cheap to compute. Also note that the accounted cost of matrix vector multiplication by $\mathbf{N}_2$ $(256\mathbf{m} + 240\mathbf{a})$ is an upper bound and can be much cheaper in practice (see Section 5.2).

Algorithm 2: Superglue4D: gluing evaluation algorithm on a product of 4 elliptic curves.

Data: A point $P \in A = \prod_{i=1}^{4} E_m$ or $P \in A^t = \prod_{i=1}^{4} E_m^t$ to evaluate, an auxiliary point $T \in A$ or $T \in A^t$, its inverse image $U^B(f(T))^{\odot -1}$, the product theta null point $\otimes_{m=1}^{4}(a_m, b_m)$ and the change of theta coordinates matrix $\mathbf{N}_2 \in \mathrm{GL}_{16}(k)$.
Result: The image theta point $\theta^B(f(P))$.

for $m = 1$ **to** 4 **do**
  Call [10, Algorithm 2] to compute the barycentric coordinates $(u_m : v_m : w_m)$ of $(P_m, T^{(m)})$; // $4(14\mathbf{M} + 5\mathbf{S} + 2\mathbf{m} + 13\mathbf{a})$
end
Compute $\mathbf{u}_{12}$ and $\mathbf{u}_{34}$ from Equation (11), (17), (18) or (19); // $2(5\mathbf{M} + \mathbf{m} + \mathbf{a})$
Compute $\mathbf{v}_{12}$ and $\mathbf{v}_{34}$ from Equation (11), (17), (18) or (19); // $2(4\mathbf{M} + \mathbf{a})$
$\mathbf{u} \leftarrow \mathbf{u}_{12} \otimes \mathbf{u}_{34}$; // $16\mathbf{M}$
$\mathbf{u} \leftarrow \mathbf{u} + \Delta_P\Delta_T\mathbf{v}_{12} \otimes \mathbf{v}_{34}$; // $9\mathbf{M} + 3\mathbf{m} + 9\mathbf{a}$
$\mathbf{v} \leftarrow \mathbf{u}_{12} \otimes \mathbf{v}_{34} + \mathbf{v}_{12} \otimes \mathbf{u}_{34}$; // $2 \times 12\mathbf{M} + 12\mathbf{a}$
$\mathbf{w} \leftarrow \overline{H}(\mathbf{u})$ (recursive Hadamard [9, Algorithm 19] and swapping coordinates); // $64\mathbf{a}$
$\mathbf{x} \leftarrow \overline{H}(\mathbf{v})$; // $64\mathbf{a}$
$\mathbf{w} \leftarrow \left(\otimes_{m=1}^{4}(a_m, b_m)\right) \odot \mathbf{w}$, $\mathbf{x} \leftarrow \left(\otimes_{m=1}^{4}(a_m, b_m)\right) \odot \mathbf{x}$; // $2 \times 16\mathbf{M}$
$\mathbf{w} \leftarrow \mathbf{N}_2 \cdot \mathbf{w}$, $\mathbf{x} \leftarrow \mathbf{N}_2 \cdot \mathbf{x}$; // $2(256\mathbf{m} + 240\mathbf{a})$
$\mathbf{w} \leftarrow \mathbf{w}^{\odot 2}$, $\mathbf{x} \leftarrow \mathbf{x}^{\odot 2}$; // $2 \times 16\mathbf{S}$
$\theta^A(P+T) \odot \theta^A(P-T) \leftarrow \mathbf{w} - \Delta_P\Delta_T\mathbf{x}$; // $16\mathbf{m} + 16\mathbf{a}$
$U^B(f(P)) \leftarrow H(\theta^A(P+T) \odot \theta^A(P-T))$; // $64\mathbf{a}$
$U^B(f(P)) \leftarrow U^B(f(P)) \odot U^B(f(T))^{\odot -1}$; // $16\mathbf{M}$
$\theta^B(f(P)) \leftarrow H(U^B(f(P)))$; // $64\mathbf{a}$
return $\theta^B(f(P))$; // Total cost: $171\mathbf{M} + 52\mathbf{S} + 541\mathbf{m} + 829\mathbf{a}$

4.2. **Computing an inverse codomain theta point.** In this paragraph, we address the computation of the inverse image dual theta auxiliary point $U^B(f(T))^{\odot -1}$ needed for the gluing evaluation (as input of Algorithm 2). As in Section 3.1, assume we are given $T_1, \cdots, T_4 \in A[8]$ such that $\ker(f) = \langle [4]T_1, \cdots, [4]T_4 \rangle$. Then, we can apply Equation (8) to T and $T_{\mathbf{j}} := \sum_{m=1}^{4}[j_m]T_m$ for some $\mathbf{j} \in (\mathbb{Z}/2\mathbb{Z})^4$,

$$U^B(f(T)) \odot U^B(f(T_{\mathbf{j}})) = H(\theta^A(T + T_{\mathbf{j}}) \odot \theta^A(T - T_{\mathbf{j}})). \tag{12}$$

Given the $\pi_{\mathbf{i},\mathbf{j}} := H(\theta^A(T + T_{\mathbf{j}}) \odot \theta^A(T - T_{\mathbf{j}}))_{\mathbf{i}}$ for all $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^4$ and $\mathbf{j} \in S$, for some support $S \subseteq (\mathbb{Z}/2\mathbb{Z})^4$, we define an instance of the HIIP (Problem 3.1), whose solution is $U^B(f(T))^{\odot -1}$.

We choose T so that $U_{\mathbf{i}}^B(f(T))$ does not vanish and the associated HIIP instance admits a solution, and choose S big enough so that the codomain graph G is connected, and we have enough equations to determine a unique solution to the HIIP by Theorem 3.2. In the qt-Pegasis context, for efficiency reasons, T is a 16-torsion point such that $[8]T \in \ker(f)$ and S contains 5 points. We also use a slightly modified version of Algorithm 1, specialised to the graph we encounter, which does not admit a Hamiltonian path due to its *Rabbit* structure (see Definition 5.2). A detailed account of this method in constant time is provided in Section 5.3 and Appendix C.

5. Application to qt-Pegasis

5.1. **Summary of qt-Pegasis.** The goal of qt-Pegasis [7] is to compute the action by any ideal on oriented supersingular elliptic curves. In the following, we focus on the implemented version of qt-Pegasis in the CSURF [32] context. We consider a prime of the form $p = c2^f - 1$ (with c small), and the set of supersingular elliptic curves E defined over $\mathbb{F}_p$ in Montgomery form, which are oriented by the quadratic imaginary order $\mathfrak{O} := \mathbb{Z}[(\sqrt{-p}+1)/2]$ *i.e.* that admit a maximal embedding $\mathfrak{O} \hookrightarrow \mathrm{End}(E)$. Here, $\sqrt{-p}$ corresponds to the Frobenius endomorphism $\pi_p : (x, y) \in E \longmapsto (x^p, y^p) \in E$ and $\mathfrak{O} \simeq \mathrm{End}_{\mathbb{F}_p}(E)$.

To compute $E_{\mathfrak{a}}$, resulting from the action of an ideal $\mathfrak{a} \subseteq \mathfrak{O}$ on an $\mathfrak{O}$-oriented curve E, we first solve a norm equation

$$\sum_{m=1}^{4} N(\mathfrak{b}_m) = 2^e, \tag{13}$$

with $e = f - 3$ and ideals $\mathfrak{b}_m \sim \mathfrak{a}$ satisfying additional conditions (*e.g.* [7, Lemmas 4-5]). We then use the solution $\mathfrak{b}_1, \cdots, \mathfrak{b}_4$ to compute a 4-dimensional 2^e-isogeny $F : E^4 \to E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times A$, where A is an abelian surface. Finally, $E_{\mathfrak{a}}$ is extracted from the codomain of F. We focus on the computation of F in the following.

Using [6, Lemma D.2], we can easily find a basis (P, Q) of $E[2^{e+2}]$ such that $P \in E(\mathbb{F}_p)$ and $Q \in E^t(\mathbb{F}_p)$, so that the basis can be defined with $\mathbb{F}_p$-rational coordinates only (either on the curve E or its twist E^t) instead of $\mathbb{F}_{p^2}$-rational coordinates. From (P, Q) and a solution to Equation (13), we can compute 2^{e+2}-torsion points $T_1, \cdots, T_4$ such that $\ker(F) = \langle [4]T_1, \cdots, [4]T_g \rangle$, where $T_1, T_2 \in E^4(\mathbb{F}_p)$ and $T_3, T_4 \in (E^t)^4(\mathbb{F}_p)$ [7, Equation (11)]. As explained in Section 2.6, by using such torsion points above $\ker(F)$, we can compute F as a chain of 2-isogenies:

$$A_0 = E^4 \xrightarrow{f_1} A_1 \xrightarrow{f_2} A_2 \quad \cdots \quad A_{e-2} \xrightarrow{f_{e-1}} A_{e-1} \xrightarrow{f_e} A_e = E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times A,$$

with operations over $\mathbb{F}_p$ only. Conditions are enforced to ensure that $f_1 : E^4 \to A_1$ is a gluing isogeny and the $f_i : A_{i-1} \to A_i$ for $i \geq 2$ are generic isogenies [7, Lemmas 4-5]. In [7], it was observed experimentally that the 2-isogenies $f_i : A_{i-1} \to A_i$ for $i \geq 2$ can be computed over $\mathbb{F}_p$. However, no algorithm was proposed to compute the first isogeny $f_1 : E^4 \to A_1$ over $\mathbb{F}_p$ instead

of $\mathbb{F}_{p^2}$.[3] Neither was it explained how f_1 could be computed in constant time, regardless of the solution of Equation (13) depending on the secret value $\mathfrak{a}$. These issues are fully addressed in Sections 5.2 and 5.3.

5.2. **On the gluing evaluation algorithm.** In this paragraph, we illustrate how Algorithm 2 can be used to evaluate the gluing isogeny $f_1 : E^4 \to A_1$ by working only over $\mathbb{F}_p$ and in constant time, while further optimising the number of needed $\mathbb{F}_p$-arithmetic operations.

First, let us remark that $p \equiv 3 \mod 4$ so that -1 is not a square in $\mathbb{F}_p$ and we can set the twisting parameter to $\beta = -1$. If E has equation $y^2 = x^3 + \alpha x^2 + x$, its quadratic twist E^t should have equation $-y^2 = x^3 + \alpha x^2 + x$. Consequently, in Algorithm 2, we have $\Delta_P, \Delta_T \in \{-1, 1\}$ and the multiplications by $\Delta_P \Delta_T$ can be done with (constant time) conditional negations.

Assume that we only evaluate points $P \in E^4(\mathbb{F}_p)$ or $P \in (E^t)^4(\mathbb{F}_p)$. Also assume that the auxiliary point T either belongs to $E^4(\mathbb{F}_p)$ or $(E^t)^4(\mathbb{F}_p)$. As we shall see in Section 5.3, T is a 16-torsion point either equal to $[2^{e-2}](T_1 + T_2) \in E^4(\mathbb{F}_p)$ or $[2^{e-2}](T_3 + T_4) \in (E^t)^4(\mathbb{F}_p)$, where the T_m have been introduced in Section 5.1. By [10, Proposition 2], the barycentric coordinates of the components of P and T are then defined over $\mathbb{F}_p$ and hence, all computations in Algorithm 2 take place over $\mathbb{F}_p$ and provided $U^{A_1}(f_1(T))^{\odot -1}$, the product theta null point $(a, b)^{\otimes 4}$ of E^4 and the change of theta coordinates matrix $\mathbf{N}_2$ are defined over $\mathbb{F}_p$, then all the gluing are defined over $\mathbb{F}_p$.

We shall see in Section 5.3 that $U^{A_1}(f_1(T))^{\odot -1}$ is solution of a HIIP defined over $\mathbb{F}_p$ so it is defined over $\mathbb{F}_p$. In [7], it is also assumed that the Montgomery coefficient α of E is such that $\alpha - 2$ is not a square, so that 4-torsion points of $E^t(\mathbb{F}_p)$ are of the form $(\pm 1 : * : 1)$ and we must have $[2^e]Q = (\pm 1 : * : 1)$ (where the 2^{e+2}-torsion basis (P, Q) has been defined in Section 5.1). Hence, the 4-torsion basis $([2^e]P, [2^e]Q)$ defines a theta null point $(a : b) = (x([2^e]P) + z([2^e]P) : x([2^e]P) - z([2^e]P))$, up to a switch of coordinates (see [7, p. 37]). This is defined over $\mathbb{F}_p$, so the product theta null point $(a, b)^{\otimes 4}$ is defined over $\mathbb{F}_p$.

Finally, it remains to explain why the change of theta coordinates matrix $\mathbf{N}_2$ is defined over $\mathbb{F}_p$ and how it can be applied at cheap cost and in constant time. This matrix can be computed by applying [9, Theorem 12] to the symplectic change of basis matrix of $E^4[4]$ given by Equation (14) (Appendix A) expressed with coefficients $n_1 \equiv 1, 3 \mod 4$ and $\sigma_1, \sigma_2, \sigma_3, \sigma_4 \in \mathbb{Z}/4\mathbb{Z}$. These values depend on the solution of Equation (13) and satisfy the conditions from [7, Lemmas 4-5], which ensure that $f_1 : E^4 \to A_1$ is a gluing isogeny and that $f_2 : A_1 \to A_2$ is a generic isogeny. There are only 32 possible values for $n_1, \sigma_1, \sigma_2, \sigma_3, \sigma_4$ to be enumerated. By enumerating all of them, we obtain only 4 possible matrices $\mathbf{N}_2$ whose coefficients are always $-1, 0, 1 \in \mathbb{F}_p$. These matrices and the corresponding values $n_1, \sigma_1, \sigma_2, \sigma_3, \sigma_4$ are presented in Appendix A. These matrices $\mathbf{N}_2$ are, up to index reordering, 4 separate dimension-2 Hadamard transforms and are therefore computable in only $32\mathbf{a}$ with [9, Algorithm 19] (to be compared with naive Hadamard matrix vector multiplication using

[3] In particular, it was not explained how we can work with sums of points $P + Q$ with $P \in E(\mathbb{F}_p)$ and $Q \in E^t(\mathbb{F}_p)$ that are necessary for computing and evaluating f_1 and defined over $\mathbb{F}_{p^2}$ in general.

$256\mathbf{m} + 240\mathbf{a}$). Given the structure of $\mathbf{N}_2$, it can be applied in constant time, independently of the values $n_1, \sigma_1, \sigma_2, \sigma_3, \sigma_4$. This explains how Algorithm 2 can run in constant time independently of inputs, with arithmetic operations over $\mathbb{F}_p$ only and at the reduced cost of $171\mathbf{M}+52\mathbf{S}+21\mathbf{m}+421\mathbf{a}$.

5.3. **A tale of rabbits: on the gluing image inverse dual theta point computation.** In this paragraph, we explain the choice of auxiliary point T used to evaluate f_1 with Algorithm 2 and how to compute $U^{A_1}(f_1(T))^{\odot -1}$ with the method suggested in Section 4.2. The choices of T and of the method to find $U^{A_1}(f_1(T))^{\odot -1}$ are made to ensure we only work over $\mathbb{F}_p$. Depending on the values of the coefficients $n_1, \sigma_1, \sigma_2, \sigma_3, \sigma_4$ introduced in Section 5.2, we either have $T := [2^{e-2}](T_1 + T_2) \in E^4(\mathbb{F}_p)$ or $T := [2^{e-2}](T_3 + T_4) \in (E^t)^4(\mathbb{F}_p)$. When $T := [2^{e-2}](T_1 + T_2)$, we use Equation (12) with indices $\mathbf{j}$ in the support $S := \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_3 + \mathbf{e}_4\}$, *i.e.* we use equations:

$$U^{A_1}\big(f_1(T)\big) \odot U^{A_1}\big(f_1([2^{e-1}]T_m)\big) = H\big(\theta^{E^4}(T + [2^{e-1}]T_m) \odot \theta^{E^4}(T - [2^{e-1}]T_m])\big)$$
$$(\mathbf{j} = \mathbf{e}_m,\ m \in [\![1, 4]\!])$$
$$U^{A_1}\big(f_1(T)\big) \odot U^{A_1}\big(f_1([2^{e-1}](T_3 + T_4))\big) = H\big(\theta^{E^4}\big(T + [2^{e-1}](T_3 + T_4)\big) \odot \theta^{E^4}\big(T - [2^{e-1}](T_3 + T_4)\big)\big)$$
$$(\mathbf{j} = \mathbf{e}_3 + \mathbf{e}_4)$$

Since $T + [2^{e-1}]T_1 = [3 \cdot 2^{e-2}]T_1 + [2^{e-2}]T_2 \in E^4(\mathbb{F}_p)$ and $T - [2^{e-1}]T_2 = [2^{e-2}]T_1 - [2^{e-2}]T_2 \in E^4(\mathbb{F}_p)$, the right side of the first equation above ($\mathbf{j} = \mathbf{e}_1$) can be computed over $\mathbb{F}_p$. The same applies for the second equation above ($\mathbf{j} = \mathbf{e}_2$). For the last three equations, the right terms can also be computed over $\mathbb{F}_p$ by using the formula from Lemma 4.4. Hence, we obtain an instance of the HIIP defined over $\mathbb{F}_p$. Similarly, when $T := [2^{e-2}](T_3 + T_4)$, we use Equation (12) with indices $\mathbf{j}$ in the support $S := \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_1 + \mathbf{e}_2\}$ and also obtain an instance of the HIIP defined over $\mathbb{F}_p$.

Remark 5.1. This choice of 16-torsion auxiliary point can be used for the computation of the second 2-isogeny of the chain $f_2 : A_1 \to A_2$, where a fifth point is needed on the support to compute it (see [7, p. 24] and Appendix D. This allows another optimisation.

As we claimed in Remark 3.7 and Algorithm 2, the codomain graphs underlying the gluing isogenies in qt-Pegasis are not Hamiltonian and we therefore cannot solve the HIIP using Algorithm 1. Obtaining Hamiltonian paths would require considering bigger supports S, which would make it impossible to work solely over $\mathbb{F}_p$. Thankfully, using S as detailed above is sufficient to ensure that our codomain graph G is connected. It has in fact a *Rabbit structure* (see Figure 1), as defined below.

Definition 5.2. A *rabbit structure* is a connected graph G with $(\mathbb{Z}/2\mathbb{Z})^g$ as set of verticies and two vertices $\mathbf{r}_1, \mathbf{r}_2 \in (\mathbb{Z}/2\mathbb{Z})^g$ (called the *ears*) connected to the same vertex and only that one. The subgraph $G \setminus \{\mathbf{r}_1, \mathbf{r}_2\}$ is called the Rabbit's *body*.

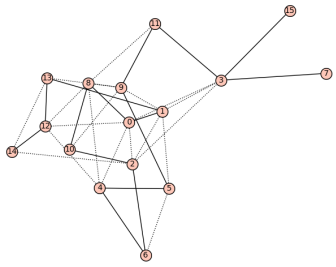

FIGURE 1. Rabbit skeleton (plain edges) included in a rabbit structure (dotted edges) in dimension $g = 4$ from an instance of the HIIP. This corresponds to the rabbit of type 7 in Table 5. Vectors $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^4$ are represented as integers decomposed in base 2 as $i_1 + 2i_2 + 4i_3 + 8i_4$.

To compute $U^{A_1}(f_1(T))^{\odot -1}$, with a 5-elements support S as above, we solve the relevant HIIP instance with an approach close to Algorithm 1. Instead of using a Hamiltonian path in G, we use a *rabbit skeleton* made of the two ears $\mathbf{r}_1, \mathbf{r}_2$ together with a Hamiltonian path in $G \setminus \{\mathbf{r}_1, \mathbf{r}_2\}$. We refer to Appendix C for algorithmic details. Note that the relevant support and rabbit skeleton can be selected very fast and in constant time from the precomputed tables of Appendix B.

5.4. **Experimental results.** Using our new formulae, we constructed the first C-implementation of 4-dimensional 2-isogenies that computes the 4-dimensional isogeny chain of qt-Pegasis in constant time[4]. It can be found in the following repository:

https://github.com/Pierrick-Dartois/qt-pegasis-Fp.

The performance of our C-implementation was evaluated on both ARM-64bits architecture[5] and x86-64bits architecture[6]. The results are shown in Table 1, together with practical timing collected using a more recent Apple CPU[7].

TABLE 1. Performance of the C-implementation of 4-dimensional chain of qt-Pegasis in CPU Mcycles. Results are the average of 100 benchmark runs.

$\lceil \log_2(p) \rceil$	p	ARM[5] (Mcycles)	X86[6] (Mcycles)	Timing[7] (ms)
505	$27 \cdot 2^{500} - 1$	146.95	274.38	23.94
1008	$15 \cdot 2^{1004} - 1$	871.34	1627.05	146.19
1525	$5 \cdot 2^{1522} - 1$	3163.05	5208.81	543.01
2017	$5 \cdot 2^{2014} - 1$	7041.82	11192.45	1222.96
4030	$45 \cdot 2^{4024} - 1$	55228.89	83757.68	10654.93

[4]However, the norm equation solver (Step 1 in Table 2), which is not the focus of this paper, is not implemented in constant time (and not yet part of the C library, as we use precomputed SageMath solutions).
[5]Apple M1 Pro 8 cores CPU with nominal clock speed 3.2 GHz, running macOS 15.6.1 and using Clang v16.0.0.
[6]Intel Xeon(R) CPU E5-1650 v3 6 cores with nominal clock speed 3.5 GHz, running Manjaro 5.10.237, and using gcc 15.1.1.
[7]Apple M4 Pro CPU with nominal clock speed of 4.5 GHz, running macOS 26.2 and using Clang v17.0.0.

Additionally, to compare our new formulae to the previous methods detailed in [9, § 4.2] and [16, § 6.1], we also implemented our new formulae into the SageMath implementation of qt-Pegasis [33]. The comparison is highlighted in Table 2. Our new formulae achieve a total speed-up between 5.7 and 19% of qt-Pegasis, depending on the prime size, with an acceleration of 3.2-17% of the isogeny chain computation (Step 3) and an acceleration of 35-58% of the elliptic curve arithmetic needed to compute kernel points, by working over $\mathbb{F}_p$ instead of $\mathbb{F}_{p^2}$ (Step 2).

TABLE 2. Comparison between new and old formulae in the SageMath implementation of qt-Pegasis. Steps 1, 2 and 3 correspond respectively to the norm equation solving, kernel points computation and isogeny chain computation. Results are the average of 100 runs on an AMD Ryzen 77840 U 8 cores CPU with 3.3 GHz clock speed, running Ubuntu 24.04.02 LTS, using SageMath 10.6 and Python 3.12.11.

	Timings with old formulae (s)				**Timings with new formulae (s)**			
$\lceil \log_2(p) \rceil$	Step 1	Step 2	Step 3	**Total**	Step 1	Step 2	Step 3	**Total**
505	0.012	0.43	0.702	**0.757**	0.012	0.018	0.583	**0.613**
1008	0.028	0.15	1.976	**2.155**	0.028	0.069	1.792	**1.889**
1525	0.028	0.355	4.133	**4.516**	0.034	0.175	3.804	**4.013**
2017	0.053	0.652	7.446	**8.152**	0.062	0.367	7.002	**7.431**
4030	0.691	3.261	37.474	**41.426**	0.680	2.110	36.283	**39.073**

REFERENCES

[1] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 423–447, Cham, 2023. Springer Nature Switzerland.

[2] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 448–471, Cham, 2023. Springer Nature Switzerland.

[3] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 472–503, Cham, 2023. Springer Nature Switzerland.

[4] Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQIsignHD: New Dimensions in Cryptography. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 3–32, Cham, 2024. Springer Nature Switzerland.

[5] Max Duparc and Tako Boris Fouotsa. SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, pages 396–429, Singapore, 2025. Springer Nature Singapore.

[6] Pierrick Dartois, Jonathan Komada Eriksen, Tako Boris Fouotsa, Arthur Herlédan Le Merdy, Riccardo Invernizzi, Damien Robert, Ryan Rueger, Frederik Vercauteren, and Benjamin Wesolowski. Pegasis: Practical effective class group action using 4-dimensional isogenies. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, pages 67–99, Cham, 2025. Springer Nature Switzerland.

[7] Pierrick Dartois, Jonathan Komada Eriksen, Riccardo Invernizzi, and Frederik Vercauteren. qt-Pegasis: Simpler and faster effective class group actions. Cryptology ePrint Archive, Paper 2025/1859, 2025.

[8] Damien Robert. The module action for isogeny based cryptography. Cryptology ePrint Archive, Paper 2024/1556, 2024.

[9] Pierrick Dartois. Fast computation of 2-isogenies in dimension 4 and cryptographic applications. *Journal of Algebra*, 683:449–514, 2025.

[10] Max Duparc. Superglue: Fast Formulae for (2,2)-Gluing Isogenies. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology – ASIACRYPT 2025*, pages 372–400, Singapore, 2026. Springer Nature Singapore.

[11] David Lubicz and Damien Robert. Computing isogenies between abelian varieties. *Compositio Mathematica*, 148(5):1483–1515, 9 2012.

[12] David Lubicz and Damien Robert. Computing separable isogenies in quasi-optimal time. *LMS Journal of Computation and Mathematics*, 18(1):198–216, 2015.

[13] David Lubicz and Damien Robert. Fast change of level and applications to isogenies. *Research in Number Theory*, 9(7), 2022.

[14] David Mumford. On the equations defining abelian varieties 1. *Inventiones mathematicae*, 1(4):287–354, 1966.

[15] Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert. An Algorithmic Approach to (2, 2)-Isogenies in the Theta Model and Applications to Isogeny-Based Cryptography. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, pages 304–338, Singapore, 2025. Springer Nature Singapore.

[16] Pierrick Dartois. *Fast computation of higher dimensional isogenies for cryptographic applications*. PhD thesis, Université de Bordeaux, France, 7 2025.

[17] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 64–93, Cham, 2020. Springer International Publishing.

[18] Kohei Nakagawa, Hiroshi Onuki, Wouter Castryck, Mingjie Chen, Riccardo Invernizzi, Gioella Lorenzon, and Frederik Vercauteren. SQIsign2D-East: A new signature scheme using 2-dimensional isogenies. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, pages 272–303, Singapore, 2025. Springer Nature Singapore.

[19] Andrea Basso, Pierrick Dartois, Luca De Feo, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQIsign2D–West. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, pages 339–370, Singapore, 2025. Springer Nature Singapore.

[20] Marius A. Aardal, Gora Adj, Diego F. Aranha, Andrea Basso, Isaac Andrés Canales Martínez, Jorge Chávez-Saab, Maria Corte-Real Santos, Pierrick Dartois, Luca De Feo, Max Duparc, Jonathan Komada Eriksen, Tako Boris Fouotsa, Décio Luiz Gazzoni Filho, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Luciano Maino, Michael Meyer, Kohei Nakagawa, Hiroshi Onuki, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Giacomo Pope, Krijn Reijnders, Damien Robert, Francisco Rodríguez Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign: Algorithm specifications and supporting documentation. Technical report, National Institute of Standards and Technology, 2025.

[21] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 411–439, Cham, 2020. Springer International Publishing.

[22] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography – PKC 2020*, pages 187–212, Cham, 2020. Springer International Publishing.

[23] Marius A. Aardal, Shahla Atapoor, Karim Baghery, Andrea Basso, Xavier Bonnetain, Giacomo Borin, Daniele Cozzo, Pierrick Dartois, Luca De Feo, Max Duparc, Jonathan K. Eriksen, Tako Boris Fouotsa, Arthur Herlédan Le Merdy, Riccardo Invernizzi, Samuel Jaques, Yi-Fu Lai, Dania Lazzarini, Jason T. LeGrow, Chloe Martindale, Luciano Maino, Jonas Meers, Michael Meyer, Sikhar Patranabis, Robi Pedersen, Giacomo Pope, Doreen Riepel, Damien Robert, Ryan Rueger, Sina Schaeffler, André Schrottenloher, and Frederik Vercauteren. PQarrots: Macaw, Kea and Kakapo, 2026. "Preview Writeup:" In anticipation of a package submission to the NIST Threshold Call.

[24] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Cham, December 2018.

[25] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005.

[26] Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, 2014.

[27] Sabrina Kunzweiler and Damien Robert. Computing modular polynomials by deformation. *Research in Number Theory*, 11(1):10, 2024.

[28] J. S. Milne. *Abelian Varieties*, pages 103–150. Springer New York, New York, NY, 1986.

[29] David Mumford. *Abelian varieties*. Oxford University Press, London, 1974. Second Edition. Tata Institute of fundamental research studies in mathematics.

[30] Christina Birkenhake and Herbert Lange. *Complex Abelian Varieties*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[31] Petr Gregor. Hypercube problems, 2012. Scribed by Václav Vlček. In *Lecture notes on Hypercube structures*.

[32] Wouter Castryck and Thomas Decru. CSIDH on the Surface. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography*, pages 111–129, Cham, 2020. Springer International Publishing.

[33] Pierrick Dartois, Jonathan Komada Eriksen, Riccardo Invernizzi, and Frederik Vercauteren. qt-Pegasis. https://github.com/KULeuven-COSIC/qt-pegasis, 2025.

[34] Lu Lu and Qiongxiang Huang. Automorphisms and isomorphisms of enhanced hypercubes. *Filomat*, 34:2805–2812, 01 2020.

Appendix A. Gluing change of theta coordinates matrices in qt-Pegasis

We present below the values of the change of theta coordinates matrices $\mathbf{N}_2$ depending on the values of the coefficients $n_1, \sigma_1, \sigma_2, \sigma_3, \sigma_4$. These are obtained by applying [9, Theorem 12] to the symplectic change of basis matrix below:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & n_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & n_1 & 0 & 0 \\ 0 & 0 & -n_1\sigma_3 & -n_1\sigma_2 & \sigma_3 & \sigma_2 & 0 & 0 \\ 0 & 0 & n_1\sigma_1 & -n_1\sigma_4 & -\sigma_1 & \sigma_4 & 0 & 0 \\ -n_1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -n_1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & n_1\sigma_4 & n_1\sigma_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -n_1\sigma_2 & n_1\sigma_3 \end{pmatrix}, \tag{14}$$

σ_1	σ_2	σ_3	σ_4	Type of $\mathbf{N}_2$
0	1	1	3	1
0	1	3	1	1
0	3	1	3	1
0	3	3	1	1
1	0	1	3	2
1	0	3	1	2
1	3	0	1	3
1	3	0	3	3

σ_1	σ_2	σ_3	σ_4	Type of $\mathbf{N}_2$
1	3	1	0	4
1	3	3	0	4
3	0	1	3	2
3	0	3	1	2
3	1	0	1	3
3	1	0	3	3
3	1	1	0	4
3	1	3	0	4

Table 3. Exhaustive list of the valid values of coefficients $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ modulo 4, with n_1 any odd integer modulo 4.

For all cases listed in Table 3, there exists a unique index $i \in \{1, 2, 3, 4\}$ such that $\sigma_i \equiv 0 \pmod 4$. This index is the *type* of the matrix $\mathbf{N}_2$. All values of $\mathbf{N}_2$ corresponding to each type are listed below.

Matrix of type 1:

$$\mathbf{N}_2 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \\
0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0
\end{pmatrix}$$

Matrix of type 2:

$$\mathbf{N}_2 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0
\end{pmatrix}$$

Matrix of type 3:

$$\mathbf{N}_2 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0
\end{pmatrix}$$

Matrix of type 4:

$$\mathbf{N}_2 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0
\end{pmatrix}$$

Appendix B. Support and Rabbit skeletons for the gluing image inverse dual theta point computation

Lemma B.1 (Rabbit classification lemma)**.** *In the context of qt-Pegasis:*

1. *There exists exactly four rabbits, each uniquely characterized by the index of its ear of Hamming weight* 3*. Concretely,* $\mathbf{r}_1 \in \{7, 11, 13, 14\}$*, and* $\mathbf{r}_2 = 15$*.*
2. *All rabbits are mutually isomorphic via coordinate permutations* $\Delta_\sigma : (i_m)_{1\le m\le 4} \longmapsto (i_{\sigma(m)})_{1\le m\le 4}$*, with* $\sigma \in V_4 \subset \mathfrak{S}_4$ *Klein's four-group.*
3. *Every rabbit admits a* rabbit skeleton*: a spanning tree determined by a triple* $(\eta, \mathbf{r}_1, \mathbf{r}_2)$*, where* $\eta : [\![0, 2^g - 3]\!] \xrightarrow{\sim} (\mathbb{Z}/2\mathbb{Z})^g \setminus \{\mathbf{r}_1, \mathbf{r}_2\}$ *is a Hamiltonian path in the rabbit body, satisfying* $\{\eta(i), \eta(i+1)\} \in E$ *for all* $i \in [\![0, 2^g - 4]\!]$*, and such that both* $\{\eta(0), \mathbf{r}_1\} \in E$ *and* $\{\eta(0), \mathbf{r}_2\} \in E$*. (See Figure 1.)*

Proof. The auxiliary points T and rabbits structures obtained for all values $n_1, \sigma_1, \sigma_2, \sigma_3, \sigma_4$ are presented in Equation (16) and in Tables 4 and 5 below. Following the approach of [7, pp. 36-38], these have been computed symbolically with scripts available in the code by identifying the zero values of the $U_{\mathbf{i}}^{A_1}(f_1([2^{e-1}]T_{\mathbf{j}}))$ for $\mathbf{i}, \mathbf{j} \in (\mathbb{Z}/2\mathbb{Z})^4$. Applying Equation (8), to $[2^{e-1}]T_{\mathbf{j}}$, we obtain

$$U^{A_1}(f_1([2^{e-1}]T_{\mathbf{j}}))^{\odot 2} = H(\theta^{E^4}([2^e]T_{\mathbf{j}}) \odot \theta^{E^4}(0_{E^4})), \tag{15}$$

where $\theta^{E^4}(0_{E^4})$ can be computed by applying the change of theta coordinates matrix $\mathbf{N}_2$ to the theta product null point $\otimes_{m=1}^4(a, b)$ of E^4 ((a, b) being the symbolic theta null point of E), and where $[2^e]T_{\mathbf{j}}$ is a 4-torsion point that can be expressed as a linear combination of $([2^e]P, [2^e]Q)$ involving the $n_1, \sigma_1, \sigma_2, \sigma_3, \sigma_4$. Using the theta coordinates $\theta^E([2^e u]P + [2^e v]Q)$ for $u, v \in \mathbb{Z}/4\mathbb{Z}$ from [7, Table 3] expressed with symbolic theta constants a and b, we can express the product theta coordinates $\otimes_{m=1}^4 \theta^E([2^e]T_{\mathbf{j}})$ and apply the change of theta coordinates matrix $\mathbf{N}_2$ to express $\theta^{E^4}([2^e]T_{\mathbf{j}})$, as desired. From these expressions and Equation (15), we can find the zero values of $U_{\mathbf{i}}^{A_1}(f_1([2^{e-1}]T_{\mathbf{j}}))$ and infer rabbit skeletons for all values $n_1, \sigma_1, \sigma_2, \sigma_3, \sigma_4 \in \mathbb{Z}/4\mathbb{Z}$ listed in Table 3. □

The type of rabbit is characterised by the following conditions on $\sigma_1, \sigma_2, \sigma_3, \sigma_4 \in \mathbb{Z}/4\mathbb{Z}$.

$$\mathbf{r}_1 = \begin{cases} 7 & \iff ((\sigma_1 = 0) \wedge (\sigma_2\sigma_4 = 1)) \vee ((\sigma_3 = 0) \wedge (\sigma_2\sigma_4 = 3)) \\ 14 & \iff ((\sigma_1 = 0) \wedge (\sigma_2\sigma_4 = 3)) \vee ((\sigma_3 = 0) \wedge (\sigma_2\sigma_4 = 1)) \\ 11 & \iff ((\sigma_2 = 0) \wedge (\sigma_1\sigma_3 = 1)) \vee ((\sigma_4 = 0) \wedge (\sigma_1\sigma_3 = 3)) \\ 13 & \iff ((\sigma_2 = 0) \wedge (\sigma_1\sigma_3 = 3)) \vee ((\sigma_4 = 0) \wedge (\sigma_1\sigma_3 = 1)) \end{cases} \tag{16}$$

Rabbit type	Auxiliary point T	Support S
7	$[2^{e-2}](T_1+T_2)$	$\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3,\mathbf{e}_4,\mathbf{e}_3+\mathbf{e}_4\}$
11	$[2^{e-2}](T_1+T_2)$	$\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3,\mathbf{e}_4,\mathbf{e}_3+\mathbf{e}_4\}$
13	$[2^{e-2}](T_3+T_4)$	$\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3,\mathbf{e}_4,\mathbf{e}_1+\mathbf{e}_2\}$
14	$[2^{e-2}](T_3+T_4)$	$\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3,\mathbf{e}_4,\mathbf{e}_1+\mathbf{e}_2\}$

TABLE 4. Auxiliary points and support corresponding to all rabbit types.

Rabbit type	Ear $\mathbf{r}_1$	Ear $\mathbf{r}_2$	Body $\eta(i)$ for $i \in [\![0,13]\!]$
7	7	15	3, 11, 9, 5, 4, 6, 2, 10, 8, 0, 1, 13, 12, 14
11	11	15	3, 7, 6, 10, 8, 9, 1, 5, 4, 0, 2, 14, 12, 13
13	13	15	12, 14, 6, 5, 1, 9, 8, 10, 2, 0, 4, 7, 3, 11
14	14	15	12, 13, 9, 10, 2, 6, 4, 5, 1, 0, 8, 11, 3, 7

TABLE 5. Rabbit skeletons of different types. Vectors $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^4$ are represented as integers decomposed in base 2 as $i_1 + 2i_2 + 4i_3 + 8i_4$.

APPENDIX C. COMPUTING AN INSTANCE OF THE HIIP ON A RABBIT

Let $G := ((\mathbb{Z}/2\mathbb{Z})^g, E)$, S, $(\pi_{\mathbf{i},\mathbf{j}})_{\mathbf{i}\in(\mathbb{Z}/2\mathbb{Z})^g,\mathbf{j}\in S}$ and $\mathbf{x} := (x_{\mathbf{i}})_{\mathbf{i}\in(\mathbb{Z}/2\mathbb{Z})^g}$ define an instance of the HIIP (with the notations from Section 3.1) and let $(\eta, \mathbf{r}_1, \mathbf{r}_2)$ be a rabbit skeleton. In order to compute Equation (5) efficiently, we reuse the ideas of Section 3.3 and adapt them to rabbits. We define $\mathbf{s}_i := \eta(i+1) - \eta(i)$ for all $i \in [\![0, 2^g - 3]\!]$ and we also define $\mathbf{t}_i := \mathbf{r}_i - \eta(0) \in S$ for $i \in \{1, 2\}$. We then see that

$$x_{\mathbf{r}_1}^{\odot -1} = \pi_{\eta(0),\mathbf{t}_1} \cdot \pi_{\mathbf{r}_2,\mathbf{t}_2} \cdot \prod_{j=0}^{2^g-4} \pi_{\eta(j+1),\mathbf{s}_j}$$

$$x_{\mathbf{r}_2}^{\odot -1} = \pi_{\mathbf{r}_1,\mathbf{t}_1} \cdot \pi_{\eta(0),\mathbf{t}_2} \cdot \prod_{j=0}^{2^g-4} \pi_{\eta(j+1),\mathbf{s}_j}$$

$$\forall i \in [\![0, 2^g - 3]\!], \quad x_{\eta(i)}^{\odot -1} = \pi_{\mathbf{r}_1,\mathbf{t}_1} \cdot \pi_{\mathbf{r}_2,\mathbf{t}_2} \cdot \left(\prod_{j=0}^{i-1} \pi_{\eta(j),\mathbf{s}_j}\right) \cdot \left(\prod_{j=i}^{2^g-4} \pi_{\eta(j+1),\mathbf{s}_j}\right)$$

We can then adapt the iterative computation of $\rho_L(i)$ and $\rho_R(i)$ from Section 3.3 and compute all their values in $(2 \cdot 2^g - 5)\mathbf{M}$. We can then compute all $x_{\mathbf{i}}^{\odot -1}$ for all $\mathbf{i} \neq \mathbf{r}_1$ using $(2^g - 2)\mathbf{M}$ while computing $x_{\mathbf{r}_1}^{\odot -1}$ can be done using an additional $2\mathbf{M}$. In total, computing the HIIP over rabbits cost $(3 \cdot 2^g - 5)\mathbf{M}$, *i.e.* using exactly one more multiplication than the generic case from Section 3.3.

Appendix D. Computing the second 2-isogeny in qt-Pegasis

It was experimentally observed in [7, § 4.3, p. 24] that the second 2-isogeny of the chain $f_2 : A_1 \to A_2$ in qt-Pegasis, though generic, could not be computed via the usual methods from [9, § 4.2] and [16, § 6.1.2] using only four 8-torsion points ($[2^{e-2}]f_1(T_m)$ for $m \in [\![1, 4]\!]$) to derive its codomain inverse dual theta null point $U^{A_2}(0_{A_2})^{\odot -1}$. Indeed, the codomain graph obtained with a support $S = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4\}$ was not connected in some cases, which made it impossible to determine a unique solution to the relevant HIIP instance by Theorem 3.2.

It is actually sufficient to add a fifth vector to the support to make the codomain graph connected. Depending on the auxiliary point T used for the gluing isogeny computation (see Remark 5.1 and Table 4), we either add $\mathbf{e}_1 + \mathbf{e}_2$ or $\mathbf{e}_3 + \mathbf{e}_4$ to the support in order to reuse $f_1(T)$. Using $[2^{e-2}]f_1(T_1), \cdots, [2^{e-2}]f_1(T_4)$ and $f_1(T)$, we can then find a Hamiltonian path to solve the relevant HIIP with Algorithm 1. However, as we have seen in Remark 3.7, we cannot use the method suggested in Section 2.6 to find a Hamiltonian path, by acting with automorphisms of the Tesseract[8] from Lemma 3.5 on the Gray Hamiltonian cycle. Indeed, the orbit under this action only contains Hamiltonian cycles of the Tesseract, whose adjacent points are always separated by a vector of Hamming weight 1 and never by our fifth support vector of Hamming weight 2. Because of this fifth point in the support ($S = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_i + \mathbf{e}_j\}$ with $(i, j) \in \{(1, 2), (3, 4)\}$), our codomain graph $G = ((\mathbb{Z}/2\mathbb{Z})^4, E)$ is a subgraph of the *extended Tesseract* $G_S := ((\mathbb{Z}/2\mathbb{Z})^4, E_S)$, where $\{\mathbf{i}, \mathbf{i} + \mathbf{j}\} \in E_S$ if and only if $\mathbf{j} \in S$. To find a Hamiltonian path in G, we may then act with the group of automorphisms of G_S, given by the following.

Lemma D.1.

(i) If $S = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_3 + \mathbf{e}_4\}$, all automorphisms of G_S are of the form

$$(i_1, i_2, i_3, i_4) \longmapsto (i_{\sigma_1(1)} + v_1, i_{\sigma_1(2)} + v_2, \sigma_2(i_3, i_4)),$$

with $\sigma_1 \in \mathfrak{S}_2$, $\mathbf{v} \in (\mathbb{Z}/2\mathbb{Z})^2$ and $\sigma_2 \in \mathfrak{S}((\mathbb{Z}/2\mathbb{Z})^2)$, a permutation of $(\mathbb{Z}/2\mathbb{Z})^2$.

(ii) If $S = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_1 + \mathbf{e}_2\}$, all automorphisms of G_S are of the form

$$(i_1, i_2, i_3, i_4) \longmapsto (\sigma_2(i_1, i_2), i_{\sigma_1(1)+2} + v_1, i_{\sigma_1(2)+2} + v_2),$$

with $\sigma_1 \in \mathfrak{S}_2$, $\mathbf{v} \in (\mathbb{Z}/2\mathbb{Z})^2$ and $\sigma_2 \in \mathfrak{S}((\mathbb{Z}/2\mathbb{Z})^2)$.

Proof. (i) is a straightforward consequence of [34, § 3]. We see G_S as the Cartesian product of the square Q_2 (hypercube of dimension 2), corresponding to indices i_1, i_2, and the complete graph with 4 elements K_4, corresponding to indices i_3, i_4. Then [34, Lemma 3.4] ensures that $\mathrm{Aut}(G_S) \simeq \mathrm{Aut}(Q_2) \times \mathrm{Aut}(K_4)$, where we know that

$$\mathrm{Aut}(Q_2) = \{(i_1, i_2) \longmapsto (i_{\sigma_1(1)+2} + v_1, i_{\sigma_1(2)+2} + v_2) \mid \sigma_1 \in \mathfrak{S}_2, \mathbf{v} \in (\mathbb{Z}/2\mathbb{Z})^2\},$$

[8] Hypercube of dimension 4

by Lemma 3.5 and where $\mathrm{Aut}(K_4) \simeq \mathfrak{S}((\mathbb{Z}/2\mathbb{Z})^2)$. (i) follows immediately, and (ii) follows from (i) by permutation of indices. □

In practice, acting with $\mathrm{Aut}(G_S)$ on the Gray Hamiltonian cycle is not always sufficient to find a Hamiltonian path. We may need to compute the orbits under the action of $\mathrm{Aut}(G_S)$ of other Hamiltonian paths containing adjacent points separated by our fifth support vector. In our implementations, besides the Gray Hamiltonian cycle, we act with $\mathrm{Aut}(G_S)$ on the following Hamiltonian path given by Table 6. This approach proved sufficient and was exhaustively validated across all 8192 instances of second isogenies in qt-Pegasis.

Support	Hamiltonian path $\eta(i)$ for $i \in [\![0, 15]\!]$
$S = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_1 + \mathbf{e}_2\}$	6, 5, 4, 0, 1, 3, 2, 10, 11, 9, 8, 12, 13, 14, 15, 7
$S = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_3 + \mathbf{e}_4\}$	6, 5, 4, 0, 8, 12, 13, 9, 1, 2, 10, 14, 15, 11, 3, 7

TABLE 6. Additional starting Hamiltonian path used in addition to the Gray Hamiltonian cycle in order to compute a second orbit under the action of $\mathrm{Aut}(G_S)$. Vectors $\mathbf{i} \in (\mathbb{Z}/2\mathbb{Z})^4$ are represented as integers decomposed in base 2 as $i_1 + 2i_2 + 4i_3 + 8i_4$.

APPENDIX E. COMPUTING BARYCENTRIC COORDINATES IN FULL GENERALITY AND GENERALISING LEMMA 4.4

We present in Algorithm 3 a generalisation of [10, Algorithm 2] for computing barycentric coordinates in full generality. We assume access to a constant-time selection primitive $\mathsf{Select}(x_0, x_1, b)$, which returns x_0 if $b = 0$ and x_1 if $b = 1$. We work with a Montgomery curve $E_\alpha : y^2 = x^3 + \alpha x^2 + x$, and define its quadratic twist as $E^t_\alpha : \beta y^2 = x^3 + \alpha x^2 + x$.

We now explain how to generalise the formula from Lemma 4.4 to evaluate a gluing isogeny $f : A := \prod_{m=1}^4 E_m \to B$ at a point $P \in A(k) \cup A^t(k)$ whose coordinates P_m may satisfy $P_m = \pm T^{(m)}$, $T \in A(k) \cup A^t(k)$ being the auxiliary point of known inverse dual image theta point $U^B(f(T))^{\odot -1}$. First, we remark that the only terms of the expression of $\theta^A(P+T) \odot \theta^A(P-T)$ that may change are the values $\mathbf{u}_{ij}$ and $\mathbf{v}_{ij}$ from Equation (11).

We then obtain the following formulae for $i, j \in [\![1, 4]\!]$:

- When $P_i = \pm T^{(i)}$ and $P_j \neq \pm T^{(j)}$:

$$\mathbf{u}_{ij} := \begin{pmatrix} u_i u_j \\ w_i u_j + \Delta_T \Delta_P v_i v_j \\ u_i w_j \\ w_i w_j \end{pmatrix} \quad \text{and} \quad \mathbf{v}_{ij} := \begin{pmatrix} u_i v_j \\ v_i u_j + w_i v_j \\ 0 \\ v_i w_j \end{pmatrix}. \tag{17}$$

- When $P_i \neq \pm T^{(i)}$ and $P_j = \pm T^{(j)}$:

$$\mathbf{u}_{ij} := \begin{pmatrix} u_i u_j \\ w_i u_j \\ u_i w_j + \Delta_P \Delta_T v_i v_j \\ w_i w_j \end{pmatrix} \quad \text{and} \quad \mathbf{v}_{ij} := \begin{pmatrix} v_i u_j \\ 0 \\ u_i v_j + v_i w_j \\ w_i v_j \end{pmatrix}. \tag{18}$$

- When $P_i = \pm T^{(i)}$ and $P_j = \pm T^{(j)}$:

$$\mathbf{u}_{ij} := \begin{pmatrix} u_i u_j \\ w_i u_j \\ u_i w_j \\ w_i w_j + \Delta_P \Delta_T v_i v_j \end{pmatrix} \quad \text{and} \quad \mathbf{v}_{ij} := \begin{pmatrix} 0 \\ v_i u_j \\ u_i v_j \\ w_i v_j + v_i w_j \end{pmatrix}. \tag{19}$$

We see that these formulae only differ from Equation (11) by coordinates switch, so that they are easy to implement without branching by using constant time swaps and index selections.

Algorithm 3: Generic barycentric coordinates

Data: Points $P = (X_P : Y_P : Z_P)$, $Q = (X_Q : Y_Q : Z_Q)$ defined on the curve E_α or its twist E^t_α. We assume that $(\alpha - 2)/4$ is precomputed.

Result: $(u : v : w) = P \pm Q$ the barycentric coordinates of P and Q, following Definition 4.2, with a boolean indicating the position of v.

$b_0 \leftarrow Z_P \stackrel{?}{=} 0$; // $P = 0$

$b_1 \leftarrow Z_Q \stackrel{?}{=} 0$; // $Q = 0$

$t_1 \leftarrow Z_P \cdot Z_Q$, $t_2 \leftarrow X_P \cdot Z_Q$;

$t_3 \leftarrow Z_P \cdot X_Q$, $t_4 \leftarrow Y_P \cdot Z_Q$;

$t_5 \leftarrow z_P \cdot Y_Q$, $t_6 \leftarrow t_4 - t_5$;

$b_2 \leftarrow t_6 \stackrel{?}{=} 0$; // $y_P = y_Q$

$t_6 \leftarrow t_4 \cdot t_5$, $t_6 \leftarrow t_6 \cdot t_1$;

$v \leftarrow t_6 + t_6$;

$t_7 \leftarrow \alpha \cdot t_1$, $t_2 \leftarrow t_2 + t_3$;

$t_7 \leftarrow t_7 + t_2$, $t_3 \leftarrow t_3 + t_3$;

$t_2 \leftarrow t_2 - t_3$, $t_2 \leftarrow t_2^2$;

$b_3 \leftarrow t_2 \stackrel{?}{=} 0$; // $P = \pm Q$

$t_7 \leftarrow t_7 \cdot t_2$, $t_4 \leftarrow t_4^2$;

$t_5 \leftarrow t_5^2$;

$t_4 \leftarrow \mathsf{Select}(t_4, \beta \cdot t_4, P \stackrel{?}{\in} E^t_\alpha)$;

$t_5 \leftarrow \mathsf{Select}(t_5, \beta \cdot t_5, Q \stackrel{?}{\in} E^t_\alpha)$;

$t_4 \leftarrow t_4 + t_5$, $t_4 \leftarrow t_4 \cdot t_1$;

$u \leftarrow t_4 - t_7$;

$w \leftarrow t_2 \cdot t_1$;

$t_0 \leftarrow X_P + Z_P$, $t_0 \leftarrow t_0^2$;

$t_1 \leftarrow X_P - Z_P$, $t_1 \leftarrow t_1^2$;

$t_2 \leftarrow t_0 - t_1$, $t_5 \leftarrow t_0 \cdot t_1$;

$t_5 \leftarrow t_5 + t_5$, $t_0 \leftarrow ((\alpha - 2)/4) \cdot t_2$;

$t_0 \leftarrow t_0 + t_1$, $t_6 \leftarrow t_0 \cdot t_2$;

$t_4 \leftarrow \mathsf{Select}(t_6, -t_6, b_3 \wedge b_2)$, $u \leftarrow \mathsf{Select}(u, t_5, b_3)$;

$v \leftarrow \mathsf{Select}(v, t_4, b_3)$, $w \leftarrow \mathsf{Select}(w, t_6, b_3)$;

$v \leftarrow \mathsf{Select}(v, 0, b_0 \vee b_1)$, $u \leftarrow \mathsf{Select}(u, X_Q, b_0)$;

$w \leftarrow \mathsf{Select}(w, Z_Q, b_0)$, $u \leftarrow \mathsf{Select}(u, X_P, b_1)$;

$w \leftarrow \mathsf{Select}(w, Z_P, b_1)$, $u \leftarrow \mathsf{Select}(u, 1, b_0 \wedge b_1)$;

return $(u : v : w), (b_3 \wedge \neg b_0 \wedge \neg b_1)$; // Total cost: $14\mathbf{M} + 5\mathbf{S} + 2\mathbf{m} + 13\mathbf{a}$

Univ. Rennes, Inria, CNRS, IRISA, UMR 6074, F-35000, Rennes, France
Email address: pierrick dot dartois at inria dot fr

EPFL, Lausanne, Switzerland
Email address: max dot duparc at epfl dot ch