Designs, Codes and Cryptography (2024) 92:1761–1802
https://doi.org/10.1007/s10623-024-01366-1

Efficient computation of $(2^n, 2^n)$-isogenies

S. Kunzweiler[1]

Received: 7 February 2023 / Revised: 1 November 2023 / Accepted: 24 January 2024 /
Published online: 12 March 2024

Abstract
Elliptic curves are abelian varieties of dimension one; the two-dimensional analogues are abelian surfaces. In this work we present an algorithm to compute $(2^n, 2^n)$-isogenies between abelian surfaces defined over finite fields. These isogenies are the natural generalization of 2^n-isogenies of elliptic curves. The efficient computation of such isogeny chains gained a lot of attention as the runtime of the attacks on SIDH (Castryck–Decru, Maino–Martindale, Robert) depends on this computation. Different results deduced in the development of our algorithm are also interesting beyond these applications. For instance, we derive a formula for the evaluation of $(2, 2)$-isogenies. Given an element in Mumford coordinates, this formula outputs the (unreduced) Mumford coordinates of its image under the $(2, 2)$-isogeny. Furthermore, we study 4-torsion points on Jacobians of hyperelliptic curves and explain how to extract square roots of coefficients of 2-torsion points from these points.

Keywords Isogeny-based cryptography · Richelot isogenies · Hyperelliptic curves · Computer algebra

Mathematics Subject Classification 11G20 · 11G10 · 14K02 · 14Q10

1 Introduction

In the past years, a lot of progress has been made in the efficiency with regard to computing elliptic curve isogenies. The popularity of this research topic originates in the introduction of the isogeny-based cryptographic primitives SIDH [11] and CSIDH [4] which are two candidates proposed for post-quantum cryptography. Both protocols describe a Diffie-Hellman key exchange, where the public keys are elliptic curves and the secret keys describe isogenies. For the generation of their public keys as well as for the computation of the shared key, both parties need to compute an isogeny of exponential (but smooth) degree. A major difference between the two protocols is that CSIDH relies on a commutative group action similar to the

Communicated by L. Mérai.

✉ S. Kunzweiler
sabrina.kunzweiler@math.u-bordeaux.fr

[1] Ruhr University Bochum, Bochum, Germany

previously developed but less efficient CRS scheme [10, 22], whereas SIDH is structurally more similar to the isogeny-based CGL hash function [5]. In 2022, a new line of attacks on the SIDH protocol appeared [2, 18, 21]. These attacks break SIDH, but have no effect on other isogeny-based protocols such as CSIDH or the CGL hash function.

A general advantage of isogeny-based protocols is their structural similarity to group-based Diffie-Hellman key exchange, which allows us to translate existing schemes into the post-quantum world more easily. However in terms of running time, other candidates are currently in the lead. To improve the efficiency of isogeny-based protocols, it is essential to further optimize isogeny computations.

Generalization of elliptic curve isogenies A generalization of pre-quantum Elliptic Curve Cryptography (ECC) is Hyperelliptic Curve Cryptography (HECC), where the group law on the Jacobian of a hyperelliptic curve is considered. While the group law computation on such Jacobians is more involved than on elliptic curves, it allows us to use a smaller prime field than in the elliptic-curve case. It is natural to ask, whether cryptographic protocols based on isogenies of elliptic curves can also be generalized to hyperelliptic curves. For instance, there exist proposals to generalize the SIDH protocol [12][1] and the CGL hash function [3, 25] to Jacobians of hyperelliptic curves of genus 2. While this allows for faster prime field arithmetic, the computation of isogenies is more involved.

$(2^n, 2^n)$-**isogenies** In this work, we focus on the computation of $(2^n, 2^n)$-isogenies, which are the natural analogues of 2^n-isogenies of elliptic curves. Let $\mathcal{J} = \mathcal{J}(\mathcal{C})$ be the Jacobian of a hyperelliptic curve $\mathcal{C}$ of genus 2. Further, let $\mathcal{J}[2^n]$ denote the 2^n-torsion of $\mathcal{J}$, which is a free $\mathbb{Z}/2^n\mathbb{Z}$-module of rank 4. Similar to the elliptic-curve case, we will consider isogenies that are defined by subgroups of $\mathcal{J}[2^n]$. However, these subgroups will not be cyclic and to describe them it is necessary to consider the *Weil pairing*, which is an alternating, bilinear pairing $e_{2^n} : \mathcal{J}[2^n] \times \mathcal{J}[2^n] \to \mu_{2^n}$. Here μ_{2^n} is the group of 2^n-th roots of unity.

A $(2^n, 2^n)$-*isogeny* is an isogeny $\phi : \mathcal{J} \to \mathcal{J}'$, where $G := \ker(\phi) \subset \mathcal{J}[2^n]$ satisfies $G \simeq \mathbb{Z}/2^n\mathbb{Z} \times \mathbb{Z}/2^n\mathbb{Z}$, and the Weil-paring restricts trivially to G, that is $e_{2^n|_G} \equiv 1$. In this case, we say that G is a $(2^n, 2^n)$-*group*. The codomain $\mathcal{J}'$ is uniquely determined up to isomorphism and it is an abelian surface. Generically, this means that it is the Jacobian of another genus-2 curve $\mathcal{C}'$.[2] Vice versa any $(2^n, 2^n)$-subgroup of $\mathcal{J}$ defines a $(2^n, 2^n)$-isogeny. In total, the Jacobian of a genus-2 curve has roughly 2^{3n} different $(2^n, 2^n)$-subgroups. This compares very favorably to the case of elliptic curves, where we have about 2^n different 2^n-groups for any elliptic curve.

From a $(2^n, 2^n)$-isogeny algorithm, we request that it takes as input a genus-2 curve $\mathcal{C}$, a description of a given $(2^n, 2^n)$-group $G \subset \mathcal{J}(\mathcal{C})$ and possibly some further elements $T_1, \ldots, T_k \in \mathcal{J}$. And it should output $\mathcal{J}(\mathcal{C}')$ together with elements $T_1', \ldots, T_k'$, such that there is an isogeny $\phi : \mathcal{J} \to \mathcal{J}(\mathcal{C}')$ with kernel G and $T_i' = \phi(T_i)$ for $i \in \{1, \ldots, k\}$.

Contributions

Our main contribution is an efficient algorithm for computing $(2^n, 2^n)$-isogenies. The computation of a $(2^n, 2^n)$-isogeny may be decomposed into n computations of $(2, 2)$-isogenies. Consequently one of the main ingredients to our algorithm, is a formula for the computation of $(2, 2)$-isogenies (Theorem 15). By this, we mean a formula that inputs data on a

[1] While there does not exist an explicit implementation of an attack on G2SIDH, it is known that the recent attacks on SIDH generalize to higher genus rendering the G2SIDH protocol insecure as well.

[2] In special cases it is also possible that $\mathcal{J}'$ is not the Jacobian of a genus-2 curve, but the product of two elliptic curves. We postpone this technicality to Sect. 3.

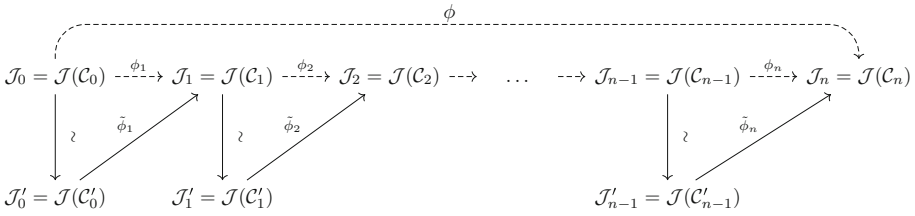


Fig. 1 Sketch of our method to compute $(2^n, 2^n)$-isogenies

$(2, 2)$-group $G \subset \mathcal{J}$ and some element $T \in \mathcal{J}$, and outputs not only the codomain of the isogeny ϕ corresponding to G, but also the image point $T' = \phi(T)$. For efficiency purposes, our formula is specialized to a specific curve form, as well as a specific form of the kernel G. The second important ingredient to our algorithm is a way to efficiently combine these specialized $(2, 2)$-isogenies in order to obtain the desired $(2^n, 2^n)$-isogeny. This is achieved by introducing a *special* symplectic basis for $\mathcal{J}[2^n]$ (Definition 3) and extracting certain square roots from the coordinates of 4-torsion elements of $\mathcal{J}$ (Corollary 5). To make this more precise, we now explain the main steps of the algorithm.

Setup Let K be some finite field of characteristic greater than 3. We start with the Jacobian $\mathcal{J}_0$ of a genus-2 curve $\mathcal{C}_0$, and a $(2^n, 2^n)$-group $G = \langle G_1, G_2 \rangle \subset \mathcal{J}_0$ with K-rational generators, and $n > 1$. Our goal is to compute the isogeny $\phi : \mathcal{J}_0 \to \mathcal{J}_n$ with kernel G. This is the top dashed arrow in Fig. 1. In this setting, it is no restriction to assume that $\mathcal{C}_0$ is defined by an equation of the form

$$\mathcal{C}_0 : y^2 = (x^2 - 1)(x^2 - A_0)(E_0 x^2 - B_0 x + C_0) \quad \text{with } A_0, B_0, C_0, E_0 \in K,$$

and the generators of G are given as

$$G_1 = T_1 + aT_3 + bT_4, \quad G_2 = T_2 + bT_3 + cT_4, \quad \text{with } a, b, c \in \mathbb{Z}/2^n\mathbb{Z}$$

for some special symplectic basis $\mathcal{B} = (T_1, T_2, T_3, T_4)$ of $\mathcal{J}_0[2^n]$.

Isogeny computation The isogeny ϕ is computed as $\phi = \phi_n \circ \cdots \circ \phi_1$, where each $\phi_i : \mathcal{J}_{i-1} \to \mathcal{J}_i$ is a $(2, 2)$-isogeny.

In the first step, we compute the $(2, 2)$-isogeny $\phi_1 : \mathcal{J}_0 \to \mathcal{J}_1$ with kernel $G_{\phi_1} = \langle 2^{n-1} G_1, \ 2^{n-1} G_2 \rangle$. To this end, we first apply a coordinate transformation so that the resulting equation is of the form

$$\mathcal{C}'_0 : y^2 = E'_0 \cdot x\,(x^2 - A'_0 x + 1)(x^2 - B'_0 x + C'_0) \quad \text{with } A'_0, B'_0, C'_0, E'_0 \in K,$$

and the kernel transforms into $G'_{\phi_1} = \langle J(x, 0), \ J(x^2 - A'_0 x + 1, 0) \rangle$. Here $J(a, b)$ denotes an element of the Jacobian with Mumford representation (a, b), see Definition 2. Such a transformation always exists due to the special setup chosen in the algorithm and can be computed efficiently by extracting square roots from the 4-torsion element $2^{n-2} G_1 \in \mathcal{J}_0$. Now, it is possible to apply the formula from Theorem 15 to explicitly compute the isogeny $\tilde{\phi}_1 : \mathcal{J}'_0 \to \mathcal{J}_1$ with kernel G'_{ϕ_1}. When composed with the transformation $\mathcal{J}_0 \to \mathcal{J}'_0$, this yields the isogeny $\phi_1 : \mathcal{J}_0 \to \mathcal{J}_1$. Via these maps, we compute the images $\phi_1(G_1), \phi_1(G_2)$ which generate a $(2^{n-1}, 2^{n-1})$-subgroup of $\mathcal{J}_1$. This completes Step 1 of the algorithm.

The isogenies $\phi_2, \dots, \phi_{n-1}$ are computed in a completely analogous way. Only the very last step $\phi_n : \mathcal{J}_{n-1} \to \mathcal{J}_n$, needs to be treated separately, since in this case, one cannot extract the square root from a 4-torsion element. More details are given in Sect. 5.3.

Note that apart from the images of the group generators, our algorithm also allows the computation of image points $\phi(T)$ for arbitrary elements $T \in \mathcal{J}_0$.

Applications

Our new algorithm for computing $(2^n, 2^n)$-isogenies has various applications. For instance, our algorithm can be used in the Castryck-Decru attack on SIDH. Indeed, our methods can be used as a drop-in replacement for the isogeny chains in their implementation and result in a speed-up of the original attack.

Since the preparation of this work predates the publications of the attacks on SIDH, the main application we had in mind was the G2SIDH protocol [12], where our algorithm can be applied for Alice's part of the key exchange. This speeds up the protocol by several orders of magnitudes.

Apart from these applications, we believe that the efficient computation of $(2^n, 2^n)$-isogenies will also be useful in the development of novel genus-2 isogeny-based cryptographic protocols. For instance, there already exists a suggestion to build a verifiable delay function using $(2^n, 2^n)$-isogenies [6]. However due to the lack of efficient methods, the authors had to restrict to a special case of $(2^n, 2^n)$-isogenies which did not make use of the rich structure of the genus-2 isogeny graph.

Comparison to previous work

Given a $(2, 2)$-group $G \subset \mathcal{J}(\mathcal{C})$ for some genus-2 curve $\mathcal{C}$, there exists a very compact formula for computing the codomain curve $\mathcal{C}'$ of the $(2, 2)$-isogeny due to Richelot [20]. Moreover the so-called *Richelot correspondence* provides a way to compute images of elements $T \in \mathcal{J}$ under this isogeny. However this method includes several steps (cf. Algorithm 1). In particular, it necessitates to compute the support of a divisor $\sum_{i=1}^{k} P_i \in \mathrm{Div}(\mathcal{C})$ representing T. This not only involves square root computations, but also requires to pass to a degree-2 extension of the base field in about half of the cases. While our method for computing $(2, 2)$-isogenies also relies on the Richelot correspondence, our formula (Theorem 15) completely replaces Algorithm 1 and only requires standard additions and multiplications in the base field.

G2SIDH implementation The computation of $(2^n, 2^n)$-isogenies in G2SIDH relies on Algorithm 1 mentioned above for which the authors provide a proof-of-concept implementation. To compare the efficiency of this algorithm with our new methods, we use the setup from [12, Appendix B]. In their example $p = 2^{51}3^{32} - 1$ and a G2SIDH key exchange in the superspecial isogeny graph over $\mathbb{F}_{p^2}$ is performed. In this protocol, Alice has to compute a $(2^{51}, 2^{51})$-isogeny $\phi_A : \mathcal{J} \to \mathcal{J}_A$ and the images $\phi_A(T_1), \ldots, \phi_A(T_4)$ of four elements $T_1, \ldots, T_4 \in \mathcal{J}$ to generate her public key. Then for the generation of the shared key, she has to perform another $(2^{51}, 2^{51})$-isogeny computation. This time without computing any image points. The authors report on timings of 145.7 and 74.8 seconds for the generation of the public key and the shared key, respectively [12]. For comparison we ran their code on our platform, a laptop with an Intel i7-8565U processor and 16 GB of RAM with Linux 5.13.0 and Magma V2.27.5.[3] The obtained timings were very much dependent on the choice of the secret key; on average the computation of the public key took around 75 seconds and the generation of the shared key around 36 seconds. The variance observed in the running

[3] Note that arithmetic in prime fields of the form $\mathbb{F}_{p^2}$ has been considerably improved in a recent update and requires Magma version at least 2.27.4.

time can be explained by the fact that the algorithm requires to compute several square roots. Depending on the input, this requires computations in a field extension at several steps.

In comparison, with our new algorithm the public key generation takes approximately 0.08 seconds and the computation of the shared key approximately 0.05 seconds.

Richelot chains in the SIDH attack Concurrently with the preparation of our manuscript, another algorithm for the efficient computation of $(2^n, 2^n)$-isogenies was developed in [2] with the goal of attacking the SIDH protocol. In essence, their approach is to replace Algorithm 1 by a (small) Gröbner basis computation. This results in a significant speed-up of the original algorithm from the G2SIDH implementation.

To compare the efficiency with our methods, we use the same setting as above. That is we compute a $(2^{51}, 2^{51})$-isogeny over $\mathbb{F}_{p^2}$ with $p = 2^{51}3^{32} - 1$. Here, the first round takes 0.24 s and the second round 0.15 s (as opposed to 0.08 and 0.05 s with our algorithm).

Moreover, there exists a new SageMath implementation of the attack [19]. The computation of $(2^n, 2^n)$-isogenies is based on the algorithm from [2] with the Gröbner basis computations replaced by explicit computations. Further the code contains algorithmic improvements similar to the ones that we describe in Remark 5. In that way, the authors achieved timings comparable to those of the Magma implementation. In our running example, the computation of a $(2^{51}, 2^{51})$-isogeny including image points takes 0.58 seconds and without image points 0.39 seconds on our laptop using SageMath version 9.7.

Genus-2 hash functions One of the first protocols based on elliptic curve isogenies is the Charles–Goren–Lauter (CGL) hash function [5]. In [25], Takashima suggests a generalization to Jacobians of genus-2 curves. Necessary improvements concerning the security have been implemented by Castryck, Decru and Smith in [3]. The genus-2 hash function relies on the computation of (2, 2)-isogenies. However the setup is different and the methods developed therein cannot be applied for computing $(2^n, 2^n)$-isogenies as are needed for a G2SIDH key exchange or the Castryck–Decru attack. In particular, for the hash function it is not necessary to compute images of elements of $\mathcal{J}$ under the isogeny, but it suffices to compute the codomains of isogenies.

Computing elliptic curve isogenies on Kummer surfaces In [7], the author develops a method to compute 2^n-isogenies of elliptic curves defined over $\mathbb{F}_{p^2}$ as isogenies of Jacobians of hyperelliptic curves defined over $\mathbb{F}_p$. To be more precise, isogenies of the Kummer surface of the Jacobians are considered. Indeed our methods partially resemble the findings in that work. In particular the methods in [7] involve a formula for pushing points through (2, 2)-isogenies which is similar to Theorem 15. However the formulas in [7] rely on the fact that the Jacobian $\mathcal{J}$ is constructed as a cover of an elliptic curve and cannot be used to compute general (2, 2)-isogenies.

The preprint [6] suggests generalizations of some of the formulas from [7] to arbitrary Kummer surfaces. However the consideration is restricted to a set of three (2, 2)-isogenies (out of 15 possible (2, 2)-isogenies) and it seems not applicable to the general case.

Similarities with elliptic curve based methods Our general strategy for the computation of isogeny chains is inspired by the methods that have been developed in the framework of elliptic curve isogeny based cryptography. For instance, to efficiently compute a chain of 2-isogenies of elliptic curves, it is suggested to use Montgomery curves $\mathcal{E} : By^2 = x^3 + Ax^2 + x$ and an isogeny formula specified to kernels of the form $G = \langle (0, 0) \rangle$. Moreover 4-torsion points can be used to avoid square root computations [11, Sect. 4.3]. Our contributions can hence be interpreted as a generalization of these methods to the 2-dimensional setting.

Outline

We start by recalling some basic facts about the arithmetic of genus-2 curves and their Jacobians in Sect. 2. In that section we also introduce two types of hyperelliptic equations that will be used throughout the paper. Further the section contains an analysis of the 4-torsion group of the Jacobian variety. Section 3 is dedicated to the theory of Richelot isogenies. In particular, we explain in detail how to use the *Richelot correspondence* to compute the image of elements of the Jacobian under an isogeny. In Sect. 4, we proceed to study Richelot isogenies in the setting of *Type-1 equations*. For this specialized setting, we derive a compact formula to compute the image of points under a Richelot isogeny. Finally in Sect. 5, we introduce $(2^n, 2^n)$-isogenies and develop an algorithm for their computation. Moreover we compare our algorithm to other implementations of $(2^n, 2^n)$-isogenies from the literature.

Appendix A provides formulas for the special cases that were excluded in Sect. 4. While these only occur with negligible probability and are not overly important from a computational perspective, some theoretically interesting configurations occur. Appendix B contains SageMath code that can be used to verify the formulas deduced in this work. Note that this code as well as implementations of our algorithm in Magma and SageMath are available at [14].

2 Arithmetic of genus-2 curves

Let K be a finite field with characteristic $p > 3$. Any algebraic curve $\mathcal{C}$ of genus 2 is a hyperelliptic curve. It admits an affine equation of the form $y^2 = f(x)$, where $f \in K[x]$ is a square-free polynomial of degree 5 or 6. We call this equation a *hyperelliptic equation* for $\mathcal{C}$. The set of points on $\mathcal{C}$ is given by

$$\mathcal{C}(\bar{K}) = \{(u, v) \in \bar{K}^2 \mid v^2 = f(u)\} \cup \begin{cases} \{\infty\} & \text{if } \deg(f) = 5, \\ \{\infty_+, \infty_-\} & \text{if } \deg(f) = 6. \end{cases}$$

We refer to points of the form $(u, v) \in \mathcal{C}(\bar{K})$ as affine points and to ∞, respectively $\infty_\pm$ as the point(s) at infinity.

The *hyperelliptic involution* $\tau : \mathcal{C} \to \mathcal{C}$ is defined by mapping a point $P = (u, v) \in \mathcal{C}(\bar{K})$ to the point $\tau(P) = (u, -v) \in \mathcal{C}(\bar{K})$. The point $P = \infty$ in the degree-5 case is fixed, while in the degree-6 case the points ∞_+, ∞_- are swapped by the involution. The *Weierstrass points* of $\mathcal{C}$ are the points that are fixed under the hyperelliptic involution. Writing $f = c_f \prod_i (x - r_i)$ for the factorization of f over $\bar{K}$, the Weierstrass points of $\mathcal{C}$ are

$$\{(r_1, 0), \ldots, (r_5, 0), \infty\} \text{ if } \deg(f) = 5, \text{ and } \{(r_1, 0), \ldots, (r_6, 0)\} \text{ if } \deg(f) = 6.$$

2.1 Equations for genus-2 curves

Given a hyperelliptic curve $\mathcal{C}$, there exist various different hyperelliptic equations for $\mathcal{C}$. Coordinate transformations as described in the well-known proposition below allow to move from one equation to the other.

Proposition 1 *(Corollary 7.4.33 in [17]) Let $\mathcal{C}$ be a hyperelliptic curve of genus 2 over K and let*

$$y^2 = f(x), \quad y'^2 = g(x')$$

be two hyperelliptic equations of $\mathcal{C}$. *Then there exist* $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{GL}_2(K)$ *and* $e \in K \setminus \{0\}$ *such that*

$$x' = \frac{ax+b}{cx+d}, \quad y' = \frac{ey}{(cx+d)^3}.$$

Remark 1 For instance, the *Rosenhain form* is a type of hyperelliptic equation. It is an equation for $\mathcal{C}$ of the form

$$y^2 = x(x-1)(x-\lambda_1)(x-\lambda_2)(x-\lambda_3).$$

This can be viewed as the analogue of the Legendre form for elliptic curves given by $y^2 = x(x-1)(x-\lambda)$.

Note that in the elliptic-curve case, we have a one-parameter family which stems from the fact that the moduli space of elliptic curves has dimension 1. On the other hand in the genus-2 case, we obtain a three-parameter family, since the moduli space has dimension 3.

We will work with two different types of hyperelliptic equations that resemble the Montgomery form for elliptic curves. These two types of equations are defined as follows.

Definition 1 Let $\mathcal{C}$ be a hyperelliptic curve of genus 2 defined over K. We say that a hyperelliptic equation has *Type* 1 if it is of the form

$$y^2 = Ex(x^2 - Ax + 1) \quad (x^2 - Bx + C) \qquad \textbf{[Type1]}$$

and *Type* 2 if it is of the form

$$y^2 = (x^2-1)(x^2-A)(Ex^2 - Bx + C) \qquad \textbf{[Type2]}$$

for some $A, B, C, E \in K$.[4]

Clearly not every genus-2 curve admits an equation of Type 1 or 2 defined over the base field, but it might be necessary to pass to a field extension. Further we note that the existence of a Type-1 equation is equivalent to the existence of a Type-2 equation over the same field. To see this, apply the coordinate change $(x', y') = \left(\frac{x-1}{x+1}, \frac{y}{(x+1)^3}\right)$ to an equation of Type 2 and redefine the constants appropriately.

A sufficient criterion for the existence of Type-1 and Type-2 equations is provided by the following proposition.

Proposition 2 *Let* $\mathcal{C}$ *be a hyperelliptic curve of genus* 2 *defined by a hyperelliptic equation* $y^2 = g(x)$ *over a finite field* K. *Assume that all Weierstrass points are* K*-rational. Then there exist hyperelliptic equations of Type* 1 *and* 2 *for* $\mathcal{C}$.

Proof Let $g = c_g \prod_i (x - r_i)$. We are going to construct a coordinate transformation t that for some $\alpha \in K$ maps four of the Weierstrass points to $(0,0)$, ∞, $(\alpha, 0)$ and $(1/\alpha, 0)$, respectively, hence generates a Type-1 equation. First note that the transformation

$$t_a : x \mapsto a \cdot \frac{x - r_1}{x - r_2} \quad \text{with } a \in K \setminus \{0\}$$

satisfies $t_a(r_1) = 0$ and $t_a(r_2) = \infty$. It remains to choose a. For that purpose consider the quantities

$$\lambda_i = \frac{r_i - r_2}{r_i - r_1} \in K \quad \text{for } i \in \{3, 4, 5\}$$

[4] The letter D is omitted on purpose since it is reserved for representing divisors.

and choose a pair $i \neq j$ such that $\lambda_i \cdot \lambda_j$ is a square in K. Note that such a pair exists since K is finite. Finally let $a \in K$ such that $a^2 = \lambda_i \cdot \lambda_j$. Then

$$t_a(r_j) = \frac{a}{\lambda_j} = \frac{\lambda_i}{a} = \frac{1}{t_a(r_i)}$$

and the resulting hyperelliptic equation with coordinates $(x', y') = \left(t_a(x),\ y/(x - r_2)^3\right)$ has Type 1. As we noted before the existence of a Type-1 equation is equivalent to the existence of a Type-2 equation. □

2.2 The Jacobian variety

Let $\mathcal{J} = \mathcal{J}(\mathcal{C})$ be the Jacobian variety of a genus-2 curve $\mathcal{C}$ defined by $y^2 = f(x)$. This is an abelian surface, in particular there exists a group structure on $\mathcal{J}$. Recall that for any field extension L/K, the group of L-rational points $\mathcal{J}(L)$ is isomorphic to the Picard group $Pic^0_{\mathcal{C}}(L)$. This means that elements of $\mathcal{J}$ can be represented as equivalence classes of degree-0 divisors on $\mathcal{C}$.

An effective divisor $D \in \mathrm{Div}(\mathcal{C})$ is in *general position* if it is of the form

$$D = P_1 + \cdots + P_d, \quad \text{for some } P_i \in \mathcal{C}(\bar{K}) \setminus \{\infty, \infty_\pm\} \text{ with } P_i \neq \tau(P_j) \text{ for } i \neq j.$$

In this case $d = \deg(D)$ is the degree of D.

Definition 2 Let $D = P_1 + \cdots + P_d$ be a divisor in general position on $\mathcal{C}$ and let $a, b \in K[x]$ with the following properties:

1. a is monic of degree d,
2. $\deg(b) < d$,
3. $f \equiv b^2 \pmod{a}$.
4. $a(u_i) = 0$, $b(u_i) = v_i$, where $P_i = (u_i, v_i)$ for $1 \leq i \leq d$. If a point $P_i = (u_i, v_i)$ appears with multiplicity in D, then a has a root of the same multiplicity in u_i.

Then we say that (a, b) is the *Mumford representation* for D.

Each divisor in general position admits a Mumford representation ([24, Lemma 4.16]). Moreover it is shown in [13, Proposition 1] that every element $[D] \in \mathcal{J}$ has a unique representative of the form $[P_1 + P_2 - D_\infty]$, where

$$D_\infty = \begin{cases} 2 \cdot \infty & \text{if } \deg(f) = 5, \\ \infty_+ + \infty_- & \text{if } \deg(f) = 6, \end{cases}$$

and $P_1 + P_2$ is an effective divisor with affine part in general position. In the generic case this means that $P_1 + P_2$ is an affine divisor in general position. But it also includes cases where one or both of P_1, P_2 are points at infinity. This allows us to represent elements of $\mathcal{J}$ using the Mumford representation of the affine part of the effective divisor $P_1 + P_2$. To avoid ambiguity, we introduce the following notation for a Mumford pair (a, b) as in Definition 2.

- $D(a, b) := P_1 + \cdots + P_d \in \mathrm{Div}(\mathcal{C})$.
- $J(a, b) := [P_1 + P_2 - D_\infty] \in \mathcal{J}(\mathcal{C})$.

The first notation $D(a, b)$ is defined for arbitrary pairs (a, b) satisfying the properties from Definition 2, while in the second notation $J(a, b)$, we implicitly assume that $\deg(a) \leq 2$. The case where $\deg(a) = 2$ is clear. If $\deg(a) = 1$, then $J(a, b) = [P_1 + P_2 - D_\infty]$,

with $P_1 = D(a, b)$ and P_2 a point at infinity. For $\deg(f) = 5$, this is well-defined, but for $\deg(f) = 6$, there are two options $P_2 \in \{\infty_\pm\}$. To simplify the exposition, we will ignore this special case here. It only becomes relevant in one special instance of our isogeny formulas which we treat in Appendix A.2. There, we also introduce the necessary notation.

2.3 Torsion points

We now proceed to study the torsion points of $\mathcal{J}$. Recall that for any positive integer m, the m-torsion of $\mathcal{J}$ is defined as

$$\mathcal{J}[m] = \{J \in \mathcal{J} \mid m \cdot J = 0\}.$$

For any point $P_0 \in \mathcal{C}(K)$, the map

$$\iota_{P_0} : \mathcal{C} \hookrightarrow \mathcal{J}, \qquad P \mapsto [P - P_0]$$

defines an embedding of $\mathcal{C}$ into $\mathcal{J}$. In this section, we only consider odd-degree models of $\mathcal{C}$, so that $\infty \in \mathcal{C}(K)$. In this setting, we choose $P_0 = \infty$ and simply write $\iota = \iota_\infty$. This means that for a point $P = (u, v) \in \mathcal{C}(\bar{K})$, we have $\iota(P) = (x - u, v)$ in Mumford representation. And $\iota(\infty) = (1, 0)$ is the identity element in $\mathcal{J}$. Note that via this embedding the hyperelliptic involution τ on $\mathcal{C}$ induces a map on $\mathcal{J}$ which corresponds to multiplication by -1 and in particular $-J(a, b) = J(a, -b)$ for any element $J(a, b) \in \mathcal{J}$.

2.3.1 Two-torsion points

The 2-torsion of the Jacobian of a hyperelliptic curve is well-studied and explicit representations are well known. We apply these results to curves with Type-1 equation, i.e. we assume

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

As described above, we fix the embedding

$$\iota : \mathcal{C} \hookrightarrow \mathcal{J}, \qquad P \mapsto [P - \infty].$$

The 2-torsion points on $\mathcal{J}$ are the divisors fixed under the action of the hyperelliptic involution τ. These are the images of the affine Weierstrass points, as well as their sums and the identity element $J(1, 0)$. Consequently, the number of 2-torsion points on $\mathcal{J}$ is $6 + \binom{5}{2} = 16$.

Let α be a root of the polynomial $x^2 - Ax + 1$ and β, γ the roots of $x^2 - Bx + C$. Then the set of Weierstrass points of $\mathcal{C}$ is given by

$$\{(0, 0), (\alpha, 0), (1/\alpha, 0), (\beta, 0), (\gamma, 0), \infty\} \subset \mathcal{C}(\bar{K}).$$

Let $P_r = (r, 0)$ be a Weierstrass point. Consequently, the Mumford representations of the 2-torsion points are

$$\begin{aligned} \iota(\infty) &= J(1, 0), \\ \iota(P_r) &= J(x - r, 0) && \text{for } r \in \{0, \alpha,\ 1/\alpha,\ \beta,\ \gamma\}, \\ \iota(P_r) + \iota(P_{r'}) &= J((x - r)(x - r'), 0) && \text{for } r \neq r' \in \{0, \alpha,\ 1/\alpha,\ \beta,\ \gamma\}. \end{aligned}$$

In general not all of these points will be defined over K. But due to the structure of Type-1 equations, the following elements are always contained in $\mathcal{J}(K)$:

$$J(1, 0), \quad J(x, 0), \quad J(x^2 - Ax + 1, 0), \quad J(x^2 - Bx + C, 0).$$

In fact, these four points form a subgroup of $\mathcal{J}[2]$, that is maximal 2-isotropic (cf. Subsection 3.1).

2.3.2 Four-torsion points

In [27], Zarhin provides explicit formulas for division by 2 on the Jacobian of a genus-2 curve [27, Theorem 3.2]. We will apply this result in order to obtain explicit representations for 4-torsion points on the Jacobian and use these to extract certain square roots. The following statement is tailored to our situation.

Proposition 3 *Let $\mathcal{C} : y^2 = g(x)$ be a degree-5 hyperelliptic equation defined over K. Let $P = (r, 0) \in \mathcal{C}(\bar{K})$ be a Weierstrass point of $\mathcal{C}$, and denote by $\{r_1, \dots, r_4\}$ the remaining roots of g.*

Then any choice of square roots

$$\mathfrak{r} = (\mathfrak{r}_1, \dots, \mathfrak{r}_4) \in \bar{K}^4 \quad \text{with } \mathfrak{r}_i^2 = r - r_i \quad \text{for } i \in \{1, 2, 3, 4\}$$

defines a 4-torsion point $[D_\mathfrak{r}] \in \mathcal{J}(\mathcal{C})$ with the property $2 \cdot [D_\mathfrak{r}] = \iota(P)$. Here $[D_\mathfrak{r}] = J(a_\mathfrak{r}, b_\mathfrak{r})$, where

$$a_\mathfrak{r} = (x - r)^2 - s_2(\mathfrak{r})(x - r) + s_4(\mathfrak{r}),$$
$$\frac{1}{\sqrt{c_g}} \cdot b_\mathfrak{r} = (s_1(\mathfrak{r})s_2(\mathfrak{r}) - s_3(\mathfrak{r}))(x - r) - s_1(\mathfrak{r})s_4(\mathfrak{r})$$

with s_i the i-th elementary symmetric polynomial in $\mathfrak{r} = (\mathfrak{r}_1, \dots, \mathfrak{r}_4)$ and c_g is the leading coefficient of g.

Proof The case $c_g = 1$ is is a direct consequence of Theorem 3.2 in [27], see also Example 3.7 in loc.cit.

Let $\mathcal{C}_1$ be the hyperelliptic curve defined by setting $c_g = 1$, i.e. $\mathcal{C}_1 : y^2 = \frac{1}{c_g} \cdot g(x)$ and let $[D] = J(a, b)$ be a 4-torsion point on $\mathcal{J}(\mathcal{C}_1)$ satisfying $2 \cdot [D] = J(x, 0) \in \mathcal{J}(\mathcal{C}_1)$. Then $[D'] = J(a, \sqrt{c_g}\, b) \in J(\mathcal{C})$ and a direct calculation shows that $2 \cdot [D'] = J(x, 0) \in \mathcal{J}(\mathcal{C})$. □

Below, we provide an example for the application of Proposition 3 to a curve given by a Type-1 equation. Together with Corollary 4 it illustrates an easy way to extract a square root from a four-torsion point. This result motivates the extraction from Corollary 5, which is obtained in a more general setting and is essential for our algorithm in Sect. 5.

Example 1 In this example, we consider a Type-1 hyperelliptic equation $y^2 = Ex(x^2 - Ax + 1)(x^2 - Bx + C)$ and apply Proposition 3 to compute the 4-torsion points $T \in \mathcal{J}$ satisfying $2 \cdot T = J(x, 0)$. In this case $r = 0$ in the above proposition and $\mathfrak{r}_1, \mathfrak{r}_2, \mathfrak{r}_3, \mathfrak{r}_4$ are square roots of the negative x-coordinates of the remaining Weierstrass points respectively. We denote

$$\mathfrak{r}_1 = \sqrt{-\alpha},\ \mathfrak{r}_2 = \sqrt{-1/\alpha},\ \mathfrak{r}_3 = \sqrt{-\beta},\ \mathfrak{r}_4 = \sqrt{-\gamma},$$

having in mind that there are in total 2^4 choices for these 4 square roots. Note that $(\mathfrak{r}_1\mathfrak{r}_2)^2 = 1$ and $(\mathfrak{r}_3\mathfrak{r}_4)^2 = C$, hence we denote $\mathfrak{r}_1\mathfrak{r}_2 = \sqrt{1}$ and $\mathfrak{r}_3\mathfrak{r}_4 = \sqrt{C}$. The elementary symmetric polynomials $s_i(\mathfrak{r})$ are

$$s_1(\mathfrak{r}) = \sqrt{-\alpha} + \sqrt{-1/\alpha} + \sqrt{-\beta} + \sqrt{-\gamma},$$
$$s_2(\mathfrak{r}) = (\sqrt{-\alpha} + \sqrt{-1/\alpha})(\sqrt{-\beta} + \sqrt{-\gamma}) + \sqrt{1} + \sqrt{C},$$

$$s_3(\mathfrak{r}) = (\sqrt{-\alpha}+\sqrt{-1/\alpha})\sqrt{C}+(\sqrt{-\beta}+\sqrt{-\gamma})\sqrt{1},$$
$$s_4(\mathfrak{r}) = \sqrt{1}\cdot\sqrt{C}.$$

It follows that the 4-torsion points satisfying $2\cdot T = J(x,0)$, have Mumford representation $T = J(a,b)$, where

$$a = x^2 - ((\sqrt{-\alpha}+\sqrt{-1/\alpha})(\sqrt{-\beta}+\sqrt{-\gamma})+\sqrt{1}+\sqrt{C})\cdot x+\sqrt{1}\cdot\sqrt{C},$$
$$b = \big((2\sqrt{1}-A+\sqrt{C})(\sqrt{-\beta}+\sqrt{-\gamma})+(2\sqrt{C}+\sqrt{1}-B)(\sqrt{-\alpha}+\sqrt{-1/\alpha})\big)\cdot\sqrt{E}\,x$$
$$+\sqrt{1}\cdot\sqrt{C}\cdot(\sqrt{-\alpha}+\sqrt{-1/\alpha}+\sqrt{-\beta}+\sqrt{-\gamma})\cdot\sqrt{E}.$$

Corollary 4 *Let $\mathcal{C}: y^2 = Ex(x^2-Ax+1)(x^2-Bx+C)$ be defined over K. Assume that $T = J(x^2+a_1x+a_0, b_1x+b_0) \in \mathcal{J}(\mathcal{C})(K)$ is a K-rational 4-torsion point satisfying $2\cdot T = J(x,0)$. Then C is a square in K and in particular $a_0^2 = C$.*

Proof This follows directly from the discussion in Example 1. □

Similarly, we obtain the following corollary in a slightly more general setting. This result is used in Proposition 9 which provides the coordinate transformation needed for the isogeny chain computations in Subsection 5.3.

Corollary 5 *Let $\mathcal{C}: y^2 = c_g\,x\,(x-\beta_1)(x-\beta_2)(x-\gamma_1)(x-\gamma_2)$ be a hyperelliptic equation. If $T = J(x^2+a_1x+a_0, b_1x+b_0) \in \mathcal{J}(K)$ satisfies $2\cdot T = J(x,0)$, then*

$$\sqrt{\beta_1\beta_2} = \frac{(a_0b_0b_1 - a_1b_0^2)\beta_1\beta_2 + c_ga_0^2(a_0-\beta_1\beta_2)^2}{b_0^2\beta_1\beta_2 + c_ga_0^2(a_0-\beta_1\beta_2)(-a_1-\beta_1-\beta_2)}$$

for some choice of $\sqrt{\beta_1\beta_2}$.

Proof Let

$$\mathfrak{r}_1 = \sqrt{-\beta_1},\ \mathfrak{r}_2 = \sqrt{-\beta_2},\ \mathfrak{r}_3 = \sqrt{-\gamma_1},\ \mathfrak{r}_4 = \sqrt{-\gamma_2}$$

be the choice of square roots corresponding to the 4-torsion element T, and let $s_1(\mathfrak{r})\ldots, s_4(\mathfrak{r})$ be the symmetric polynomials in $\mathfrak{r}_1,\ldots,\mathfrak{r}_4$. One can verify algebraically (cf. Appendix B.1) that

$$\sqrt{-\beta_1}\sqrt{-\beta_2} = \frac{s_1(\mathfrak{r})s_3(\mathfrak{r})\beta_1\beta_2 + (s_4(\mathfrak{r})-\beta_1\beta_2)^2}{\beta_1\beta_2 s_1(\mathfrak{r})^2 + (s_4(\mathfrak{r})-\beta_1\beta_2)(s_2(\mathfrak{r})-\beta_1-\beta_2)}.$$

Using Proposition 3, we extract the values of s_i from the Mumford coordinates of T:

$$s_1(\mathfrak{r}) = \frac{-b_0}{a_0\sqrt{c_g}},\quad s_2(\mathfrak{r}) = -a_1,\quad s_3(\mathfrak{r}) = \frac{b_0a_1-b_1a_0}{a_0\sqrt{c_g}},\quad s_4(\mathfrak{r}) = a_0.$$

Substituting these expressions into the equation for $\sqrt{-\beta_1}\sqrt{-\beta_2}$ above, yields the formula in the statement of the corollary. □

3 Richelot isogenies

Let $\mathcal{C}$ be a genus-2 curve with hyperelliptic equation $y^2 = g(x)$, where $g(x) = c_g\prod_{i=1}^{d}(x-r_i)$ and $\mathcal{J}(\mathcal{C})$ its Jacobian. Given a group $G \subset \mathcal{J}(\mathcal{C})[2]$ that is maximal 2-isotropic, there exists a morphism

$$\phi: \mathcal{J}(\mathcal{C}) \to \mathcal{A} \quad \text{with } \ker(\phi) = G.$$

The map ϕ is a $(2,2)$-*isogeny* and $\mathcal{A}$ is an abelian surface. The abelian surface $\mathcal{A}$ is either the Jacobian of a hyperelliptic curve or the product of two elliptic curves.

Isogenies of this form have been extensively studied in the literature. In particular there exist very compact formulas to compute the codomain of a given isogeny and a correspondence that can be used to compute the image of divisors under the isogeny. These findings are attributed to Richelot, therefore $(2,2)$-isogenies are usually called *Richelot isogenies*. In this section, we recall the necessary background for the next section. For proofs we refer to [12, 23].

3.1 (2, 2)-Subgroups

A group $G \subset \mathcal{J}[2]$ is called a $(2,2)$-*subgroup* of $\mathcal{J}$ if $G \simeq \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ and G is isotropic with respect to the 2-Weil paring meaning that e_2 restricts trivially to G, where $e_2 : \mathcal{J}[2] \times \mathcal{J}[2] \to \{\pm 1\}$.

Recall that $\mathcal{J}[2]$ is a $\mathbb{Z}/2\mathbb{Z}$-module of rank 4, therefore there are 15 non-trivial 2-torsion elements in $\mathcal{J}$. The $(2,2)$-subgroups of $\mathcal{J}$ can be described very explicitly. Let $[D] = J(a,b) \in \mathcal{J}[2]$ and $[D'] = J(a',b') \in \mathcal{J}[2]$ be elements of order 2. Then $b = b' = 0$ and the roots of a and a' are x-coordinates of the Weierstrass points of $\mathcal{C}$. One can check that $e_2(D, D') = 1$ if and only if $a \cdot a'$ divides g and $\frac{g}{a \cdot a'}$ is a polynomial of degree 1 or 2 (see [23, Lemma 8.1.4]). Moreover in this case $[D] + [D'] = (\frac{g}{c_g aa'}, 0)$. This property already characterizes $(2,2)$-subgroups. To simplify the exposition, we define $r_6 = \infty$ and $x - r_6 = 0 \cdot x + 1$ if $\deg(g) = 5$.[5]

Lemma 6 *With the notation above, a group $G \subset \mathcal{J}[2]$ is a $(2,2)$-subgroup if and only if*

$$G = \langle J(g_1, 0), J(g_2, 0) \rangle,$$

where $g_1 = (x - r_{\sigma(1)})(x - r_{\sigma(2)})$ and $g_2 = (x - r_{\sigma(3)})(x - r_{\sigma(4)})$ for some permutation $\sigma \in S_6$. In that case,

$$G = \big\{ J((x - r_{\sigma(1)})(x - r_{\sigma(2)}), 0),\ J((x - r_{\sigma(3)})(x - r_{\sigma(4)}), 0),\\ J((x - r_{\sigma(5)})(x - r_{\sigma(6)}), 0),\ J(1, 0) \big\}.$$

It follows that the $(2,2)$-groups correspond precisely to the partitions of the set of Weierstrass points into subsets of size 2, hence there are precisely 15 such groups. In [23], this relation is formalized by introducing *quadratic splittings*.

3.2 Richelot correspondence

The next proposition provides information on the codomain of an isogeny defined by a $(2,2)$-subgroup.

Proposition 7 *Let $G = \langle J(g_1, 0), J(g_2, 0) \rangle$ be a $(2,2)$-subgroup and $g_3 = \frac{g}{g_1 g_2}$, so that $g = g_1 \cdot g_2 \cdot g_3$. Denote $g_i = g_{i,2}x^2 + g_{i,1}x + g_{i,0}$ for $i \in \{1,2,3\}$. Further let $\phi : \mathcal{J} \to \mathcal{A}$ be the isogeny with kernel G and*

$$\delta = \det \begin{pmatrix} g_{1,0} & g_{1,1} & g_{1,2} \\ g_{2,0} & g_{2,1} & g_{2,2} \\ g_{3,0} & g_{3,1} & g_{3,2} \end{pmatrix}.$$

[5] In other words r_6 is the x-coordinate of the Weierstrass point at infinity and 1 is the polynomial with a root at ∞.

1. *If $\delta \neq 0$, then $\mathcal{A}$ is isomorphic to the Jacobian of the genus-2 curve*
$$\mathcal{C}' : y^2 = h_1 h_2 h_3,$$
where
$$h_1 = \delta^{-1} \cdot (g_2' g_3 - g_2 g_3'), \quad h_2 = \delta^{-1} \cdot (g_3' g_1 - g_3 g_1'), \quad h_3 = \delta^{-1} \cdot (g_1' g_2 - g_1 g_2'),$$
and g_i' denotes the derivative of g_i with respect to x.
2. *If $\delta = 0$, then $\mathcal{A}$ is isomorphic to a product of elliptic curves $\mathcal{E}_1 \times \mathcal{E}_2$ with defining equations*
$$\mathcal{E}_1 : y^2 = \prod_{i=1}^{3} (a_{i,1} x + a_{i,2}), \qquad \mathcal{E}_2 : y^2 = \prod_{i=1}^{3} (a_{i,1} + a_{i,2} x),$$
where $a_{i,0}, a_{i,1}$ are such that $g_i = a_{i,1}(x - s_1)^2 + a_{i,2}(x - s_2)^2$ for some $s_1, s_2 \in K$.

Proof The first part is [23, Theorem 8.4.11]. The second part follows from the discussion in [23, Section 8.3]. □

Note that the element δ defined in the proposition is only well defined up to multiplication by ± 1, since it depends on the ordering of the polynomials g_1, g_2, g_3. A different choice of the sign corresponds to computing an isogeny to the Jacobian of a quadratic twist of $\mathcal{C}'$.

In order to compute the image of an element in $\mathcal{J}(\mathcal{C})$ under an isogeny ϕ, we restrict to the first case of the above proposition, i.e. we assume that $\delta \neq 0$.

Proposition 8 *Let $\mathcal{C}$ and $\mathcal{C}'$ be as defined in Part 1 of Proposition 7. Then the $(2,2)$-isogeny $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ from Proposition 7 is defined by the correspondence $\mathcal{R} \subset \mathcal{C} \times \mathcal{C}'$ with*
$$\begin{aligned} \mathcal{R} : \quad 0 &= g_1(u) h_1(u') + g_2(u) h_2(u') \\ v v' &= g_1(u) h_1(u')(u - u') \end{aligned} \tag{1}$$
for points $(P, P') = \big((u, v), (u', v')\big) \in \mathcal{C} \times \mathcal{C}'$.

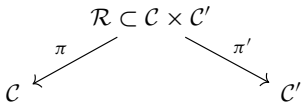
Proof This is [23, Theorem 8.4.11]. □

The correspondence defined in Proposition 8 is called *Richelot correspondence*. Given a point $P = (u, v) \in \mathcal{C}$, the first equation in (1) has two solutions for u' and the second equation has precisely one solution for v' (depending on u'). This means that one point on $\mathcal{C}$ *corresponds* to two points on $\mathcal{C}'$. The correspondence extends uniquely to a homomorphism of the Jacobians. In the following, we will describe this map more explicitly. To simplify the exposition, we make the following assumptions:

- $\mathcal{C}$ is defined by a degree-5 equation, hence $D_\infty = 2\infty \in \mathrm{Div}(\mathcal{C})$.
- $\mathcal{C}'$ contains a rational Weierstrass point P_0'.

Note that we will be in this situation for the formulas developed in the next section. In most cases, $\mathcal{C}'$ will be defined by a degree-6 extension, hence $D_\infty' = \infty_+ + \infty_- \in \mathrm{Div}(\mathcal{C}')$.

Let us consider the following diagram.



Here π and π' are the projections from $\mathcal{R}$ to $\mathcal{C}$ and $\mathcal{C}'$ respectively. This gives rise to a morphism $\psi : \mathcal{C} \to \mathcal{J}(\mathcal{C}')$, where for a point $P \in \mathcal{C}$, we first consider its pullback along π to obtain a divisor $R = \pi^{-1}(P)$. Here, this divisor is of the form $R = (P, P_1) + (P, P_2) \in \mathrm{Div}(\mathcal{R})$. The pushforward along π' yields $P_1 + P_2 \in \mathrm{Div}(\mathcal{C}')$. Finally this divisor is mapped to the Jacobian via the embedding $\iota' : \mathcal{C}' \to \mathcal{J}(\mathcal{C}')$, $P' \mapsto [P' - P'_0]$ for some K-rational Weierstrass point P'_0 of $\mathcal{C}'$. Choosing a Weierstrass point has the advantage that the hyperelliptic involution induces multiplication by $[-1]$. The map ψ is summarized below.

$$\begin{aligned} \psi : \mathcal{C} &\xrightarrow{\pi^*} \mathrm{Div}(\mathcal{R}) \xrightarrow{\pi'_*} \mathrm{Div}(\mathcal{C}') \xrightarrow{\iota'} \mathcal{J}(\mathcal{C}'), \\ P &\longmapsto (P, P_1) + (P, P_2) \longmapsto P_1 + P_2 \longmapsto [P_1 + P_2 - 2P'_0]. \end{aligned}$$

Finally ψ induces a homomorphism of the Jacobians of $\mathcal{C}$ and $\mathcal{C}'$,

$$\begin{aligned} \phi : \quad \mathcal{J}(\mathcal{C}) &\to \mathcal{J}(\mathcal{C}'), \\ [P + Q - D_\infty] &\mapsto \psi(P) + \psi(Q) - 2\psi(\infty). \end{aligned}$$

Using the correspondence from Proposition 8. the computation of $\psi(P)$ is straightforward for an affine point $P \in \mathcal{C}(K)\backslash\{\infty\}$. To compute $\psi(\infty)$, we use that one of g_i for $i \in \{1, 2, 3\}$ has degree 1, and write $[P^* - \infty] \in G$ for the corresponding element in the kernel of ϕ. Then $\psi(\infty) = \psi(P^*)$ can be computed using the coordinates of the affine point P^*.

Note that ϕ does not depend on the embedding $\iota' : \mathcal{C}' \to \mathcal{J}(\mathcal{C}')$ that was chosen in the construction of ψ. Moreover, with $P_1 + P_2$ as above and analogously $Q_1 + Q_2 = \pi'_* \circ \pi^*(Q)$, we have that

$$\psi(P) + \psi(Q) - 2\psi(\infty) = [P_1 + P_2 - D'_\infty] + [Q_1 + Q_2 - D'_\infty] \in \mathcal{J}(\mathcal{C}'),$$

where we used that $2 \cdot \psi(\infty) - 2 \cdot [D'_\infty] = 0$.

The above discussion contains all ingredients to explicitly compute the image of elements $J(a, b) \in \mathcal{J}(\mathcal{C})$ under the isogeny ϕ. For future reference, the overall procedure is summarized in Algorithm 1. We restrict this description to the case where $\deg(a) = 2$. The case $\deg(a) = 1$ is easier since in this case $J(a, b) = [P - \infty]$ for a point $P \in \mathcal{C}(K)$ and in particular $\psi(P) \in \mathcal{J}(\mathcal{C})[K]$.

We would like to point out that Algorithm 1 is not new, but it is a standard procedure to compute the image of elements in $\mathcal{J}(\mathcal{C})$ under a $(2, 2)$-isogeny, see for example [7, 12]. The algorithm consists of four main steps.

Step 1 concerns the computation of the codomain of ϕ, more precisely an equation for the curve $\mathcal{C}'$ such that $\mathcal{J}(\mathcal{C}')$ is the codomain of ϕ. This is done as outlined in Proposition 7. The remaining steps are needed to compute $\phi(J(a, b))$.

In **Step 2** the support of the divisor $D(a, b)$ is computed, that is we compute $P, Q \in \mathcal{C}(\bar{K})$ with $D(a, b) = P + Q$. This requires the computation of the roots of the polynomial $a \in K[x]$. In about half of the cases this also requires passing to a degree-2 field extension of K.

In **Step 3** the Richelot correspondence (Eq. 1) is used to compute the divisors $D_P = \pi'_* \circ \pi^*(P)$ and $D_Q = \pi'_* \circ \pi^*(Q) \in \mathrm{Div}(\mathcal{C}')$.

In **Step 4** we compute $J(a', b') = \phi(J(a, b))$ as the sum of $[D_P - D'_\infty]$ and $[D_Q - D'_\infty]$. This summation is done using Cantor's algorithm. It consists of a composition step and a reduction step. For more details on Cantor's algorithm, the reader is referred to [16, 24] in the odd-degree case and [13] in the even-degree case.

The procedure is illustrated in Example 2 in the next section.

Algorithm 1 Computing (2, 2)-isogenies

Input: A curve $\mathcal{C} : y^2 = g_1(x)g_2(x)g_3(x)$, the group $G = \langle J(g_1, 0),\ J(g_2, 0)\rangle$, and an element $J(a, b) \in \mathcal{J}(\mathcal{C})$, where $\deg(a) = 2$.

Output: A curve $\mathcal{C}'$ and an element $J(a', b') \in \mathcal{J}(\mathcal{C}')$ such that there is an isogeny $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ with kernel G and $\phi(J(a, b)) = J(a', b')$.

Step 1 *Compute* $\mathcal{C}'$.
$\delta = \det\left((g_{ij})_{1\le i\le 3,\ 0\le j\le 2}\right)$
for $i = 1$ **to** 3 **do**
  $h_i = \delta^{-1}(g'_{i+1}g_{i+2} - g_{i+1}g'_{i+2})$, indices are viewed mod 3.
Set $\mathcal{C}' : y^2 = h_1h_2h_3$.

Step 2 *Compute* $P, Q \in \mathcal{C}(\bar{K})$ *with* $J(a, b) = [P + Q - D_\infty]$.
Compute the roots $u, s \in \bar{K}$ of $a \in K[x]$.
Evaluate $v = b(u), t = b(s) \in \bar{K}$.
$\Rightarrow P = (u, v)$ and $Q = (s, t)$.

Step 3 *Compute* $D_P, D_Q \in \mathrm{Div}(\mathcal{C}')$.
Set $D_P = D(a_P, b_P)$, where
$a_P = \mathsf{monic}(g_1(u)h_1(x) + g_2(u)h_2(x))$,
$b_P = g_1(u)h_1(x)(u - x) \cdot v^{-1} \pmod{a_P}$.
Set $D_Q = D(a_Q, b_Q)$, where
$a_Q = \mathsf{monic}(g_1(s)h_1(x) + g_2(s)h_2(x))$,
$b_Q = g_1(s)h_1(x)(s - x) \cdot t^{-1} \pmod{a_Q}$.

Step 4 *Compute* $[D'] = [D_P + D_Q - 2D'_\infty]$ *using Cantor's algorithm.*
(a) Composition:
  $\Rightarrow D(a_{comp}, b_{comp}) = D_P + D_Q \in \mathrm{Div}(\mathcal{C}')$.
(b) Reduction:
  $\Rightarrow J(a', b') = [D(a_{comp}, b_{comp})] \in \mathcal{J}(\mathcal{C}')$.

4 Richelot isogenies for type-1 equations

In this section, we consider a genus-2 curve $\mathcal{C}$ defined by a Type-1 equation

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

Moreover, we fix the (2, 2)-group

$$G = \langle J(x, 0),\, J(x^2 - Ax + 1, 0)\rangle \subset \mathcal{J}(\mathcal{C})[2]$$

and restrict our considerations to the isogeny $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{A}$ with $\ker(\phi) = G$. First, we show that under some mild conditions any (2, 2)-group may be transformed into a group of this form (Proposition 9) and then translate the results from the previous section into our setting. In the second part, we develop formulas that completely replace Algorithm 1. Our main result is Theorem 15.

4.1 Richelot correspondence for type-1 equations

In order to apply the formulas that will be developed in this section for an arbitrary (2, 2)-isogeny ϕ, it is necessary to perform a coordinate transformation to obtain a kernel of the desired form. In general, this might require to extend the field of definition. The next proposition shows that a coordinate transformation is possible over the base field K if there exists

a K-rational point T of order 4 such that $2 \cdot T$ is in the kernel of ϕ. Since the goal of this work is to compute $(2^n, 2^n)$-isogenies (see Sect. 5), this is not a serious restriction.

Proposition 9 *Let $g_1, g_2, g_3 \in K[x]$ be quadratic polynomials, $\mathcal{C}$ a genus-2 curve, defined an equation of the from $y^2 = g_1(x)g_2(x)g_3(x)$ and $G = \langle J(g_1, 0), J(g_2, 0)\rangle$ a $(2,2)$-subgroup of $\mathcal{J}(\mathcal{C})$. If the roots of g_1 are K-rational and there exists a K-rational 4-torsion point $T \in \mathcal{J}(\mathcal{C})$ such that $2 \cdot T = J(g_1(x), 0)$, then there exists a rational coordinate transformation $t : (x, y) \mapsto (x', y')$ such that*

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

is a Type-1 equation for $\mathcal{C}$ and $G = \langle J(x', 0), J(x'^2 - Ax' + 1, 0)\rangle$.

Proof The transformation t is constructed as the composition of two transformations, t_1 and t_2. We denote

$$g_1 = (x - \alpha_1)(x - \alpha_2), \quad g_2 = (x - \beta_1)(x - \beta_2), \quad g_3 = c_g \cdot (x - \gamma_1)(x - \gamma_2).$$

Note that $\alpha_1, \alpha_2 \in K$ by assumption, and $\beta_1, \beta_2, \gamma_1, \gamma_2 \in \bar{K}$. The first transformation is defined as

$$t_1 : x \mapsto \frac{x - \alpha_2}{x - \alpha_1}, \quad y \mapsto \frac{y}{(x - \alpha_1)^3}.$$

This leads to an equation of the form

$$y^2 = c_g \cdot x(x - \beta_1')(x - \beta_2')(x - \gamma_1')(x - \gamma_2'),$$

where β_i' and γ_i' are the images of β_i and γ_i respectively.

The final transformation is of the form $t_2 : x \mapsto a \cdot x$, where a satisfies $a^2 = 1/(\beta_1'\beta_2')$. This square root can be extracted from the Mumford coordinates of the 4-torsion element $t_1(T)$ as explained in Corollary 5. □

The next two propositions are translations of Proposition 7 and Proposition 8 to the setting specified in this section.

Proposition 10 *Let $\mathcal{C}$ be a genus-2 curve defined by a Type-1 equation $y^2 = Ex(x^2 - Ax + 1)(x^2 - Bx + C)$ and let $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{A}$ be the isogeny with kernel $\ker(\phi) = \langle J(x, 0), J(x^2 - Ax + 1, 0)\rangle \subset \mathcal{J}(\mathcal{C})[2]$.*

1. *If $C \neq 1$, then $\mathcal{A}$ is isomorphic to the Jacobian of the genus-2 curve with Type-2 equation*

$$\mathcal{C}' : y^2 = (x^2 - 1)(x^2 - A')(E'x^2 - B'x + C'),$$

where

$$A' = C, \quad B' = \frac{2}{E}, \quad C' = \frac{B - AC}{E(1 - C)}, \quad E' = \frac{A - B}{E(1 - C)}.$$

2. *If $C = 1$, then $\mathcal{A}$ is isomorphic to a product of elliptic curves $\mathcal{E}_1 \times \mathcal{E}_2$ with defining equations*

$$\mathcal{E}_1 : y^2 = c_1 \cdot (x - 1)\left(x - \frac{A + 2}{A - 2}\right)\left(x - \frac{B + 2}{B - 2}\right),$$
$$\mathcal{E}_2 : y^2 = c_2 \cdot (x - 1)\left(x - \frac{A - 2}{A + 2}\right)\left(x - \frac{B - 2}{B + 2}\right),$$

where

$$c_1 = E \cdot (A - 2)(B - 2) \quad \text{and} \quad c_2 = -E \cdot (A + 2)(B + 2).$$

Proof The proposition is implied by Proposition 7. To see this, first note that

$$\delta = E \cdot \det \begin{pmatrix} 0 & 1 & 0 \\ 1 & -A & 1 \\ C & -B & 1 \end{pmatrix} = -E \cdot (1 - C),$$

hence $\delta = 0$ if and only if $C = 1$.

The case $C \neq 1$ can be easily verified by a direct computation. Note that we further applied the coordinate change $(x, y) \mapsto (x, (1 - C) \cdot y)$ in order to obtain a simpler form of the equation for $\mathcal{C}'$.

For the case $C = 1$, we use the identities

$$\begin{aligned} x &= \frac{1}{4}(x+1)^2 - \frac{1}{4}(x-1)^2, \\ x^2 - Ax + 1 &= \frac{-A+2}{4}(x+1)^2 + \frac{A+2}{4}(x-1)^2, \\ x^2 - Bx + 1 &= \frac{-B+2}{4}(x+1)^2 + \frac{B+2}{4}(x-1)^2. \end{aligned}$$

Inserting these values into the elliptic curve equations provided in Proposition 7 and scaling x appropriately, yields the desired result. □

The description of the Richelot correspondence simplifies as well when applied in our specific setting.

Proposition 11 *Let $\mathcal{C}$ and $\mathcal{C}'$ be as defined in Part* 1 *of Proposition 10, in particular $C \neq 1$. Then the $(2, 2)$-isogeny $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ from Proposition 7 is defined by the correspondence $\mathcal{R} \subset \mathcal{C} \times \mathcal{C}'$ with*

$$\begin{aligned} \mathcal{R}: \quad 0 &= \left(u^2 - Bu + 1\right) \cdot u'^2 + 2(C-1)u \cdot u' - Cu^2 + Bu - C \\ vv' &= (A-B)u \cdot u'^3 - \left((A-B)u^2 + 2(1-C)u\right) \cdot u'^2 \\ &\quad + \left(2(1-C)u^2 - (AC-B)u\right) \cdot u' + (AC-B)u^2 \end{aligned}$$

for points $(P, P') = \left((u, v), (u', v')\right) \in R \subset \mathcal{C} \times \mathcal{C}'$.

Proof This is a consequence of Proposition 8 with $g_1 = x$, $g_2 = x^2 - Ax + 1$ and $h_1 = E'x^2 - B'x + C'$, $h_2 = x^2 - A'$. Note that we applied the same coordinate change $(u', v') \mapsto (u', (1 - C) \cdot v')$ to points in $\mathcal{C}'$ as in the previous proposition. □

Example 2 Here, we illustrate the general Richelot isogeny procedure (Algorithm 1) in the setting of Type-1 equations. This means, we use Propositions 10, 11 instead of Proposition 7, 8.

Let $\mathcal{C}$ be a genus-2 curve over the finite field $\mathbb{F}_7$ with Type-1 equation $y^2 = x(x^2 - x + 1)(x^2 - 3x + 2)$. We want to compute the isogeny ϕ with kernel $G = \langle x, x^2 - x + 1 \rangle$ and evaluate it at the element $J(x^2 + 4, x + 4) \in \mathcal{J}(\mathcal{C})$.

First note that $C = 2 \neq 1$, hence the codomain of the isogeny is again the Jacobian of a hyperelliptic curve. More precisely, we have $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ with

$$\mathcal{C}' : y^2 = (x^2 - 1)(x^2 - 2)(2x^2 - 2x - 1)$$

as per Proposition 10. For the computation of the image point, we first determine the points in the support of $J(a, b)$. For this purpose, we need to factor the polynomial $a = x^2 + 4 = (x + 2i)(x - 2i)$ which requires working in the field $\mathbb{F}_{7^2} = \mathbb{F}_7(i)$. We find

$$J(x^2 + 4, x + 4) = [P + Q - D_\infty], \quad \text{with } P = (2i, 4 + 2i),\ Q = (-2i, 4 - 2i).$$

Next, we use the correspondence from Proposition 11 in order to compute the divisors D_P and D_Q on $\mathcal{C}'$. Evaluating the correspondence at P and Q respectively, we find that

$$D_P = (x^2 + (3i+6)x + 6i + 3, (3i+3)x + 5i + 1),$$
$$D_Q = (x^2 + (4i+6)x + i + 3, (4i+3)x + 2i + 1).$$

It remains to compute the sum $J(a', b') = [D_P + D_Q - 2D_\infty] \in \mathcal{J}(\mathcal{C}')$. Following Cantor's algorithm, the composition step yields

$$D_{\text{comp}} = D_P + D_Q = (x^4 + 5x^3 + 2x^2 + 2x + 3, x^3 + 2x^2 - 2x),$$

and finally

$$J(a', b') = J(x^2 + 3x - 3, 3x + 3).$$

An important observation in the last example is that while the input and output of the algorithm are $\mathbb{F}_7$-rational, it was necessary to compute over the quadratic extension $\mathbb{F}_{7^2}$ for the intermediate steps. Our goal is to avoid these intermediate computations. This motivates the derivation of explicit formulas.

4.2 Explicit formulas

In this section, we present compact formulas for the Richelot isogeny ϕ. By this we mean formulas for the image $\phi(J(a, b))$ for any element $J(a, b) \in \mathcal{J}(\mathcal{C})$.

First, we consider the easier case, where $J(a, b) = [P - \infty]$. Here it is necessary to distinguish between Weierstrass points (Proposition 12) and general points $P \in \mathcal{C}(K)$ (Proposition 14). Note that in these cases, our formulas do not provide a major advantage over Algorithm 1.

A significant speed-up occurs in the general case $J(a, b)$, where a is a degree-2 polynomial. In that case, Algorithm 1 necessitates to factor the polynomial a and possibly pass to a degree-2 extension of the ground field, whereas our formula completely avoids these computations. It works only with the Mumford coordinates and consists of additions, multiplications and inversions in the ground field. This formula is provided in Theorem 15. It presents the main result of this section.

In the main theorem, we have to exclude some edge cases which are treated in Appendix A. The first of these cases is when $D(a, b)$ is supported at a Weierstrass point of $\mathcal{C}$. This situation is very similar to the case where $J(a, b) = [P - \infty]$ and is explained in Section A.1. The second special case is when $\gcd(a, x^2 - Bx + 1) \neq 1$. In this case, it is necessary to consider elements of the form $[P + \infty_\pm - D'_\infty] \in \mathcal{J}(\mathcal{C}')$ to describe the image $\phi(J(a, b))$. These were the elements that we excluded in the notation introduced in Sect. 2.2. The necessary notation and formulas for this case are provided in Appendix A.2. The last special case concerns divisors where the polynomial a is a square or $a = (x - u)(x - 1/u)$ for some $u \in \bar{K}$. This case is treated in Section A.3. All possible cases and criteria to decide which formula to apply are summarized in Table 1. To keep this overview compact, we did not include precise references to the intersection of cases (i.e. when two different criteria apply). But this information is of course included in the statements. The last column of the table also provides an overview concerning the frequency of these cases, where $q = \#K$ is assumed to be large. Apart from the general case in Theorem 15, all other cases appear with negligible probability for randomly chosen elements $J(a, b) \in \mathcal{J}(\mathcal{C})$.

We would like to point out that this case distinction is inherent in describing elements of the Jacobian by their Mumford coordinates. Indeed, a similar case distinction is necessary

Table 1 Formulas for the image of $J(a, b)$ under the $(2, 2)$-isogeny ϕ

Criteria	Formula	Number of cases
$\mathbf{a = x + a_0},\ \mathbf{b = b_0}$		
$b_0 = 0$	Proposition 12	$O(1)$
$a_0^2 + Ba_0 + 1 = 0$	Proposition 21	$O(1)$
$b_0(a_0^2 + Ba_0 + 1) \neq 0$	Proposition 14	$O(q)$
$\mathbf{a = x^2 + a_1 x + a_0},\ \mathbf{b = b_1 x + b_0}$		
$b_1(a_1 b_0 - a_0 b_1) + b_0^2 = 0$	Corollary 19	$O(q)$
$a_0 B^2 + (a_0 + 1)a_1 B + (a_0 - 1)^2 + a_1^2 = 0$	Propositions 23, 24	$O(q)$
$(a_0 - 1)(a_1^2 - 4a_0) = 0$	Propositions 26, 27	$O(q)$
general case	Theorem 15	$O(q^2)$

in concurrently developed methods for the computation of Richelot isogeny chains [2, 19]. These works currently only describe the general case that we treat in Theorem 15.

Throughout, we assume that $\mathcal{C}$ is a genus-2 curve defined by a Type-1 equation

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

with $C \neq 1$. Further $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ is the isogeny with kernel $\ker(\phi) = \langle J(x, 0), J(x^2 - Ax + 1, 0)\rangle \subset \mathcal{J}(\mathcal{C})[2]$ from Proposition 10. In particular, $\mathcal{C}'$ is of the form

$$\mathcal{C}' : y^2 = (x^2 - 1)(x^2 - A')(E'x^2 - B'x + C'),$$

with

$$A' = C, \quad B' = \frac{2}{E}, \quad C' = \frac{B - AC}{E(1 - C)}, \quad E' = \frac{A - B}{E(1 - C)}.$$

Proposition 12 *Let $P \in \mathcal{C}(K)$ be a Weierstrass point, then $\phi([P - \infty])$ is as described below.*

1. *If $P \in \{(0, 0), \infty\}$, then $\phi([P - \infty]) = 0$.*
2. *If $P = (\alpha, 0)$, where $\alpha^2 - A\alpha + 1 = 0$, then $\phi([P - \infty]) = J(x^2 - 1, 0)$.*
3. *If $P = (\beta, 0)$, where $\beta^2 - B\beta + C = 0$, then $\phi([P - \infty]) = J(x^2 - A', 0)$.*

Proof In Case 1, if $P = \infty$, then $[P - \infty] = 0 \in \mathcal{J}(\mathcal{C})$, so there is nothing to show. For $P = (0, 0)$, we have $[(0, 0) - \infty] = [(0, 0) + \infty - 2\infty] \in \ker(\phi)$ by definition.

For the next cases, we fix a Weierstrass point $P_0' \in \mathcal{C}'(K)$ and use the map $\psi : \mathcal{C} \to \mathcal{J}(\mathcal{C}')$ subject to the embedding $\iota : \mathcal{C}' \to \mathcal{J}(\mathcal{C}')$, $Q \mapsto [Q - P_0']$ as defined in Sect. 3.2. Moreover, we note that the Richelot correspondence (Proposition 11) implies $\psi(\infty) = \psi((0, 0)) = [D(x^2 - A', 0) - 2P_0']$.

In Case 2, we find $\psi(P) = [D(E'x^2 - B'x + C', 0) - 2P_0']$. It follows that

$$\begin{aligned}\phi([P - \infty]) &= [D(E'x^2 - B'x + C', 0) - D(x^2 - A', 0)] \\ &= [D(E'x^2 - B'x + C', 0) + D(x^2 - A', 0) - 2D_\infty'] \\ &= J(x^2 - 1, 0).\end{aligned}$$

Here, we did not normalize the Mumford representation of $D(E'x^2 - B'x + C', 0)$ so that the case $E' = 0$ is included.

For Case 3, denote $D(x^2 - Bx + C, 0) = P + Q$ with $P = (\beta, 0)$ and $Q = (\gamma, 0)$. The first relation in the Richelot correspondence shows that $\psi(P) = [P_1 + P_2 - 2P_0']$, where $x(P_1) = x(P_2) = \beta$. Similarly $\psi(Q) = [Q_1 + Q_2 - 2P_0']$, where $x(Q_1) = x(Q_2) = \gamma$. The second relation vanishes for all possible y-coordinates for P_1, P_2 and Q_1, Q_2. Indeed, we find that $\tau(P_1) = P_2$ and $\tau(Q_1) = Q_2$, where τ is the hyperelliptic involution $\tau : \mathcal{C}' \to \mathcal{C}'$. To see this, note that necessarily

$$\begin{aligned}[P_1 + P_2 - D(x^2 - A', 0)] &= \phi([P - \infty]) = -\phi([Q - \infty]) \\ &= -[Q_1 + Q_2 - D(x^2 - A', 0)].\end{aligned}$$

Adding $J(x^2 - A', 0)$ on both sides yields

$$[P_1 + P_2 - D'_\infty] = [\tau(Q_1) + \tau(Q_2) - D'_\infty].$$

Since $x(P_1) = x(P_2) \neq x(Q_1) = x(Q_2)$, this implies that both sides of the equation are zero, hence $P_1 = \tau(P_2)$ and $Q_1 = \tau(Q_2)$ as claimed above. Consequently,

$$\phi([P - \infty]) = \phi([Q - \infty]) = J(x^2 - A', 0).$$

□

The following Lemma makes Step 3 in Algorithm 1 more explicit. Given a point $P \in \mathcal{C}(K)$, it provides a formula for the corresponding divisor $D_P \in \mathrm{Div}(\mathcal{C}')$ under the Richelot correspondence. The provided formula will be used in many of the subsequent propositions.

Lemma 13 *Let $\mathcal{R} \subset \mathcal{C} \times \mathcal{C}'$ be the Richelot correspondence defined in Proposition 11 and denote by $\pi : \mathcal{R} \to \mathcal{C}$, $\pi' : \mathcal{R} \to \mathcal{C}'$ the natural projections from this correspondence. If $P = (u, v) \in \mathcal{C}(\bar{K})$ with $(u^2 - Bu + 1) \cdot v \neq 0$, then $D_P := \pi'_* \circ \pi^*(P)$ is equal to $D(a_P, b_P)$, where $a_P = x^2 + a_{P,1}x + a_{P,0}$, $b_P = b_{P,1}x + b_{P,0}$ with*

$$\begin{aligned}a_{P,1} &= \frac{2(C-1)u}{u^2 - Bu + 1}, \qquad a_{P,0} = \frac{-Cu^2 + Bu - C}{u^2 - Bu + 1}, \\ b_{P,1} &= \frac{u(1-C)(u^2 - Au + 1)}{(u^2 - Bu + 1)^2 \cdot v} \cdot (2u^3 - Bu^2 + (-B^2 + 4C - 2)u + B), \\ b_{P,0} &= \frac{-u(1-C)(u^2 - Au + 1)}{(u^2 - Bu + 1)^2 \cdot v} \cdot (Bu^3 + (-B^2 + 2C)u^2 - Bu + 2C).\end{aligned}$$

Proof The statement is deduced from the description of the Richelot correspondence provided in Proposition 11. Normalizing the first equation from the correspondence, yields a_P. Then b_P is obtained by dividing the right hand side of the second equation in the proposition by v and reducing this modulo a_P. □

The next proposition explains the computation of the image $\phi(J(a, b))$, where $J(a, b) = [P - \infty]$ and P is not a Weierstrass point.

Proposition 14 *Let $\mathcal{C}$ be a genus-2 curve defined by a Type-1 equation $y^2 = Ex\,(x^2 - Ax + 1)(x^2 - Bx + C)$ and assume $C \neq 1$. Further let $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ be the isogeny with kernel $\ker(\phi) = \langle J(x, 0), J(x^2 - Ax + 1, 0)\rangle \subset \mathcal{J}(\mathcal{C})[2]$ from Proposition 10. Then for an element $J(a, b) = J(x + a_0,\ b_0) \in \mathcal{J}(\mathcal{C})$ with $b_0(a_0^2 + Ba_0 + 1) \neq 0$, its image under the isogeny ϕ is given by*

$$\phi(J(a, b)) = \left[D_P + D_Q - 2D'_\infty\right] \in \mathcal{J}(\mathcal{C}'),$$

where $D_Q = D(x^2 - A', 0)$ and $D_P = D(a_P, b_P)$ as in Lemma 13 for $(u, v) = (-a_0, b_0)$.

Proof We have that $\mathcal{J}(x + a_0, b_0) = [(-a_0, b_0) - \infty]$. This means that

$$\phi(J(a, b)) = \psi((-a_0, b_0)) - \psi(\infty),$$

where $\psi : \mathcal{C} \to \mathcal{J}(\mathcal{C}')$ is the map induced by the Richelot correspondence $\mathcal{R}$ in Proposition 11 with respect to the embedding $\iota : \mathcal{C}' \to \mathcal{J}(\mathcal{C}'),\ P \mapsto [P - P'_0]$ (see Sect. 3.2).

As in the proof of the previous proposition, we use that $\psi(\infty) = [D(x^2 - A', 0) - 2P'_0]$. Further $\psi((-a_0, b_0)) = [D_P - 2P'_0]$, where D_P is as in Lemma 13 (with $(u, v) = (-a_0, b_0)$).

In conclusion

$$\begin{aligned}\phi(J(x + a_0, b_0)) &= [D(a_P, b_P) - D(x^2 - A', 0)] \\ &= [D(a_P, b_P) + D(x^2 - A', 0) - 2D'_\infty],\end{aligned}$$

where we used that $2 \cdot \left[D(x^2 - A', 0) - D'_\infty\right] = 0$. □

The remainder of this section is dedicated to Theorem 15 and its proof. This theorem provides a formula for the image of a general element $J(x^2 + a_1x + a_0,\ b_1x + b_0) \in \mathcal{J}(\mathcal{C})$ under ϕ.

Theorem 15 *Let $\mathcal{C}$ be a genus-2 curve defined by a Type-1 equation $y^2 = Ex\,(x^2 - Ax + 1)(x^2 - Bx + C)$ and assume $C \neq 1$. Further let $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ be the isogeny with kernel $\ker(\phi) = \langle J(x, 0), J(x^2 - Ax + 1, 0)\rangle \subset \mathcal{J}(\mathcal{C})[2]$ from Proposition 10. We assume that $J(a, b) = J(x^2 + a_1x + a_0,\ b_1x + b_0) \in \mathcal{J}(\mathcal{C})$ satisfies*

$$\begin{aligned}0 &\neq -b_1(a_1b_0 - a_0b_1) + b_0^2, \\ 0 &\neq a_0B^2 + (a_0 + 1)a_1B + (a_0 - 1)^2 + a_1^2, \\ 0 &\neq (a_0 - 1)(a_1^2 - 4a_0).\end{aligned} \tag{2}$$

Then

$$\phi(J(a, b)) = \left[D\left(\frac{a'_4x^4 + a'_3x^3 + a'_2x^2 + a'_1 + a'_0}{a'_4}, \frac{b'_3x^3 + b'_2x^2 + b'_1x + b'_0}{b'_{den}} \right) - 2D'_\infty \right],$$

where

$$\begin{aligned}a'_0 &= ((a_0 - 1)^2 + a_1^2)C^2 + (a_0 + 1)a_1BC + a_0B^2 \\ a'_1 &= 2 \cdot (C - 1) \cdot ((a_0 + 1)a_1C + 2a_0B) \\ a'_2 &= -(a_0 + 1)a_1B(C + 1) - 2a_0B^2 + 4a_0C^2 - 2((a_0 + 1)^2 + a_1^2)C + 4a_0 \\ a'_3 &= -2 \cdot (C - 1) \cdot (2a_0B + (a_0 + 1)a_1) \\ a'_4 &= a_0B^2 + (a_0 + 1)a_1B + (a_0 - 1)^2 + a_1^2\end{aligned}$$

and

$$\begin{aligned}\mu &= a_1b_0 - a_0b_1 \\ b'_0 &= a_0\mu AB + (a_0b_0(a_0 - 1) + a_1\mu)\,AC + a_0\,(a_1\mu - b_0(a_0 - 1))\,B \\ &\quad + \mu((a_0 - 1)^2 + a_1^2)C \\ b'_1 &= a_0b_0AB + (a_0a_1b_0 + \mu(a_0 + 1))\,AC - 2a_0\mu A + a_0(\mu + b_1)B \\ &\quad + \left(2a_0a_1\mu + b_0(-a_0^2 + a_1^2 + 1)\right)C - 2a_0a_1\mu + 2a_0b_0(a_0 + 1) \\ b'_2 &= -a_0\mu AB + 2a_0b_0AC + (-a_0b_0(a_0 + 1) - a_1\mu)\,A + a_0\,(-a_1\mu + b_0(a_0 - 1))\,B \\ &\quad + 2a_0\,(\mu + b_1)\,C - (a_0^2 + a_1^2 + 1)\mu\end{aligned}$$

$$b_3' = -a_0 b_0 AB + \left(-a_0^2 b_1 - \mu\right) A - a_0(\mu + b_1)B - b_0((a_0-1)^2 + a_1^2)$$
$$b_{den}' = (a_0 - 1) \cdot (-\mu b_1 + b_0^2).$$

Note that the formulas as presented in Theorem 15 are not completely optimized. Instead, we decided for a presentation that achieves a better readability. For an optimized version, where the number of multiplications and additions is reduced, the reader is referred to our implementation [14].

Proof The proof involves several symbolic computations that were performed using the Computer Algebra System SageMath [26]. Here, we explain the overall strategy and give some details on the computations. The formulas that we obtained may be verified using the Code provided in Appendix B and in our GitHub repository [14].

Let $\mathcal{C}$ be a genus-2 curve defined by a Type-1 equation $y^2 = Ex\,(x^2 - Ax + 1)(x^2 - Bx + C)$ and $J(a, b) = (x^2 + a_1 x + a_0,\ b_1 x + b_0) \in \mathcal{J}(\mathcal{C})$. In order to verify that the formula for the image $\phi(J(a, b))$ in the theorem is correct, we follow Algorithm 1 symbolically and compare the output with our formula. The computations are preformed over the ring

$$R = \mathbb{Q}[A, B, C, a_0, a_1, b_0, b_1, u],$$

where we assume that u is a root of the polynomial $x^2 + a_1 x + a_0$. This provides us with a first relation among the variables. Moreover, it must hold that $Ex\,(x^2 - Ax + 1)(x^2 - Bx + C) - (b_1 x + b_0)^2 \equiv 0 \pmod{x^2 + a_1 x + a_0}$, since the pair (a, b) represents a divisor on the curve. This provides us with two more relations. We let $I \triangleleft R$ be the ideal defined by these three relations.

The first step of the algorithm consists in computing the codomain curve. This is already covered by Proposition 10, so we directly proceed to the second step. This step requires to compute the support $P = (u, v)$, $Q = (s, t)$ of the divisor $D(a, b)$. Recall that u is a variable in our ring R. The remaining coordinates can be represented as follows.

$$v = b_1 u + b_0, \quad s = -a_1 - u, \quad t = b_1 s + b_0.$$

In the third step, we compute the divisors D_P and D_Q that correspond to P and Q under the Richelot correspondence. Here, we can use the explicit description from Lemma 13. We denote $D_P = D(a_P, b_P)$ and $D_Q = D(a_Q, b_Q) \in \mathrm{Div}(\mathcal{C}')$, where a_P, b_P are just as in the statement of the lemma and a_Q, b_Q are obtained by replacing (u, v) by (s, t). Note that the first two inequalities in (2) guarantee that we do not divide by zero in this step. To make this more precise, $0 \neq a_0 B^2 + (a_0 + 1)a_1 B + (a_0 - 1)^2 + a_1^2$ is equivalent to requiring that $u^2 - Bu + 1$ and $s^2 - Bs + 1$ are non-zero (cf. Lemma 22); and $0 \neq -b_1(a_1 b_0 - a_0 b_1) + b_0^2$, means that none of P and Q are Weierstrass points, hence $v, t \neq 0$ (cf. Lemma 18).

Finally, we perform Step 4(a), the composition step of Cantor's Algorithm with output the Mumford representation $D(a', b') = D_P + D_Q$. Here we make use of the third inequality $0 \neq (a_0 - 1)(a_1^2 - 4a_0)$ which implies that $\gcd(a_P, a_Q) = 1$ (cf. Lemma 25). In that case $a' = a_P \cdot a_Q$. This results in a symbolic expression for a' with coefficients in (the fraction field of) R. This expression is considerably more complicated than the one provided in the theorem. In particular, it still contains the variable u representing the x-coordinate of one of the points in the support of the divisor. To verify that the simpler expression from our theorem is correct, we check that the coefficients coincide modulo the ideal I.

It remains to verify that b' is correct. We know that it is the unique polynomial in $\mathrm{Frac}(R)[x]$ with $\deg(b') \leq 3$ that satisfies

$$b' \equiv b_P \pmod{a_P},$$
$$b' \equiv b_Q \pmod{a_Q},$$
$$b'^2 \equiv f \pmod{a'},$$

where $f = (x^2 - 1)(x^2 - A')(E'x^2 - B'x + C')$ is the defining polynomial for $\mathcal{C}'$. One can verify that these three properties are satisfied for the polynomial b' provided in the theorem. Again, it is necessary to take into account the relations contained in the ideal I, when checking the identities. □

Remark 2 The formula provided in Theorem 15 replaces steps 1, 2, 3, 4(a) in Algorithm 1. This results in a major speed-up in the isogeny computation, since all of the square root computations as well as the computation of a field extension are avoided. In order to find the (reduced) Mumford representation (a'', b'') for the divisor class $\mathcal{J}(\mathcal{C})$, it remains to carry out Step 4(b). Here, this last step consists of two computations:

- Computing the quotient $\left(f - b'^2\right)/a'$, where $f = (x^2 - 1)(x^2 - A')(E'x^2 - B'x + C')$ is the defining polynomial for $\mathcal{C}'$. The normalization of that quotient is then a monic polynomial a'' of degree at most 2.
- Computing the residue of $-b'$ modulo a''. This residue is the polynomial b'' with $\deg(b'') < \deg(a'')$.

Both of these computations can be executed very efficiently using the methods developed for HECC.

It is also possible to extract a formula for the reduced Mumford representation directly. However the formula that we obtained is not very compact, hence it is computationally preferable to use the formula from Theorem 15 and then perform Step 4(b) of Algorithm 1 when computing a (2, 2)-isogeny of the given form.

Example 3 To make the description more explicit, let us go back to Example 2. That is we consider the genus-2 curve $\mathcal{C}$ over $\mathbb{F}_7$ with Type-1 equation $y^2 = x(x^2 - x + 1)(x^2 - 3x + 2)$. And we want to evaluate the isogeny ϕ with kernel $G = \langle x, x^2 - x + 1\rangle$ at the element $J(x^2 + 4, x + 4) \in \mathcal{J}(\mathcal{C})$.

Instead of following Algorithm 1, we may now simply evaluate the formula from Theorem 15 at

$$A = 1, B = 3, C = 2, E = 1, \quad a_0 = 4, a_1 = 0, b_0 = 4, b_1 = 1.$$

This yields

$$a' = x^4 + 5x^3 + 2x^2 + 2x + 3, \; b' = x^3 + 2x^2 + 5x.$$

The divisor $D(a', b')$ coincides with the unreduced Mumford presentation computed in Example 2. We see that the new formula replaces almost all steps in the algorithm and allows us to perform all computations over $\mathbb{F}_7$.

5 Efficiently computing $(2^n, 2^n)$-isogenies

In this section, we first introduce $(2^n, 2^n)$-isogenies and analyze these in more detail for the case of Type-2 equations. In particular, we define the term *special symplectic basis*. Then, we present our algorithm for computing $(2^n, 2^n)$-isogenies as chains of (2, 2)-isogenies and compare its efficiency to other algorithms in the literature.

5.1 $(2^n, 2^n)$-Isogenies

Let $\mathcal{A}$ be a principally polarized abelian surface. For any $n \in \mathbb{N}$, the 2^n-torsion group $\mathcal{A}[2^n]$ is a $\mathbb{Z}/2^n\mathbb{Z}$-module of rank 4. Let $\boldsymbol{\mu}_{2^n}$ be the multiplicative group of 2^n-th roots of unity. The Weil pairing

$$e_{2^n} : \mathcal{A}[2^n] \times \mathcal{A}[2^n] \to \boldsymbol{\mu}_{2^n}$$

is an alternating, bilinear pairing on this module. We say that a basis (T_1, T_2, T_3, T_4) for $\mathcal{A}[2^n]$ is *symplectic* (w.r.t. the Weil pairing) if for some primitive 2^n-th root $\mu \in \boldsymbol{\mu}_{2^n}$,

$$e_{2^n}(T_i, T_j) = \mu^{\pm 1} \text{ if } j = i \pm 2$$

and the Weil pairing is trivial on all other combinations of basis elements, more precisely

$$e_{2^n}(T_i, T_j) = 1 \text{ if } j \notin \{i \pm 2\}.$$

Phrased differently, the pairing matrix of the basis is of the form

$$\begin{pmatrix} \log(e_{2^n}(T_1, T_1)) & \dots & \log(e_{2^n}(T_1, T_4)) \\ \vdots & & \vdots \\ \log(e_{2^n}(T_4, T_1)) & \dots & \log(e_{2^n}(T_4, T_4)) \end{pmatrix} = \begin{pmatrix} 0 & \mathrm{Id}_2 \\ -\mathrm{Id}_2 & 0 \end{pmatrix},$$

where the logarithm is taken with respect to μ and Id_2 is the identity matrix.

We are interested in isogenies $\phi : \mathcal{A} \to \mathcal{A}'$ that can be evaluated as a non-backtracking chain of n $(2, 2)$-isogenies. The kernels of such isogenies are maximal 2^n-isotropic subgroups of $\mathcal{A}$. The group structure of such groups is analyzed in [12]. In particular, the authors show that for any maximal 2^n-isotropic subgroup G, there exists a $k \in \{0, \dots, \lfloor \frac{n}{2} \rfloor\}$ such that

$$G \simeq \mathbb{Z}/2^n\mathbb{Z} \times \mathbb{Z}/2^{n-k}\mathbb{Z} \times \mathbb{Z}/2^k\mathbb{Z}.$$

We restrict our considerations to the case of rank-2 subgroups (i.e. the case $k = 0$). These constitute roughly two thirds of all 2^n-isotropic groups of $\mathcal{A}$. For short, we say that an isogeny $\phi : \mathcal{A} \to \mathcal{A}'$ is a $(2^n, 2^n)$-*isogeny* if $G = \ker \phi \simeq \mathbb{Z}/2^n\mathbb{Z} \times \mathbb{Z}/2^n\mathbb{Z}$ and call G a $(2^n, 2^n)$-*group*.

Given a $(2^n, 2^n)$-group $G = \langle G_1, G_2 \rangle \subset \mathcal{A}[2^n]$, we consider the isogeny chain

$$\mathcal{A} = \mathcal{A}_0 \xrightarrow{\phi_1} \mathcal{A}_1 \longrightarrow \dots \xrightarrow{\phi_i} \mathcal{A}_i \longrightarrow \dots \xrightarrow{\phi_n} \mathcal{A}_n = \mathcal{A}', \qquad \mathcal{A}_0 \xrightarrow{\psi_i} \mathcal{A}_i$$

where $\phi_i : \mathcal{A}_{i-1} \to \mathcal{A}_i$ is the isogeny with kernel $2^{n-i}\langle \psi_{i-1}(G_1), \psi_{i-1}(G_2) \rangle$ and $\psi_i = \phi_i \circ \dots \circ \phi_1$.

5.2 $(2^n, 2^n)$-Groups and type-2 equations

The set of $(2^n, 2^n)$-groups has been analyzed in [15]. In particular the authors provide a method for the random sampling of such groups when provided with a symplectic basis (T_1, T_2, T_3, T_4) of $\mathcal{A}[2^n]$. As suggested in that article, we restrict to the subset

$$\mathcal{G} = \left\{ \langle T_1 + aT_3 + bT_4,\ T_2 + bT_3 + cT_4 \rangle \mid a, b, c \in \mathbb{Z}/2^n\mathbb{Z} \right\} \tag{3}$$

of $(2^n, 2^n)$-subgroups. Each tuple $(a, b, c) \in (\mathbb{Z}/2^n\mathbb{Z})^3$ defines a different $(2^n, 2^n)$-group, hence groups can be sampled by choosing (a, b, c) at random. Of course, this sampling

method depends on the choice of the symplectic basis for $\mathcal{A}[2^n]$. In the following, we will discuss a choice that is particularly favorable for our setting.

From now on, we consider a genus-2 curve $\mathcal{C}$ given by a Type-2 equation

$$\mathcal{C} : y^2 = (x^2 - 1)(x^2 - A)(Ex^2 - Bx + C)$$

for some $A, B, C, E \in K$, and the abelian variety $\mathcal{J} = \mathcal{J}(\mathcal{C})$. We denote the Weierstrass points of $\mathcal{C}$ by

$$\{(1, 0), (-1, 0), (\alpha, 0), (-\alpha, 0), (\beta, 0), (\gamma, 0)\},$$

where α is a square root of A and β, γ are the roots of $(Ex^2 - Bx + C)$. As before, we assign $\gamma = \infty$ if $E = 0$, and in this case treat the polynomial $x - \gamma$ as a constant.

Lemma 16 *Let $\mathcal{C}$ be defined by a Type-2 equation. Then $\mathcal{B} = (T_1, T_2, T_3, T_4)$ with*

$$T_1 = J\left((x-1)(x-\alpha), 0\right), \qquad T_3 = J\left((x-1)(x+1), 0\right),$$
$$T_2 = J\left((x+\alpha)(x-\beta), 0\right), \qquad T_4 = J\left((x-\beta)(x-\gamma), 0\right),$$

is a symplectic basis for $\mathcal{J} = \mathcal{J}(\mathcal{C})[2]$, where α, β, γ are as defined above.

Proof This is easily verified by a direct computation. □

Lemma 17 *Let $\mathcal{B} = (T_1, T_2, T_3, T_4)$ and $\mathcal{C}$ as in Lemma 16. Then the set $\mathcal{G}$ of $(2, 2)$-groups from Eq. 3 comprises the 8 groups of the form*

$$\left\langle J\left((x - (-1)^i)(x - (-1)^j\alpha), 0\right),\ J\left((x - (-1)^{j+1}\alpha)(x - r), 0\right)\right\rangle,$$

where $i, j \in \{0, 1\}$ and $r \in \{\beta, \gamma\}$.

Proof For $i \in \{0, \ldots, 7\}$, define

$$G_i = \langle T_1 + a_i T_3 + b_i T_4,\ T_2 + b_i T_3 + c_i T_4\rangle,$$

where (a_i, b_i, c_i) is the 2-adic representation of i, meaning $i = 4a_i + 2b_i + c_i$ with $(a_i, b_i, c_i) \in \{0, 1\}^3$. Then

$$\begin{aligned}
G_0 &= \langle J\left((x-1)(x-\alpha), 0\right),\ J\left((x+\alpha)(x-\beta), 0\right)\rangle,\\
G_1 &= \langle J\left((x-1)(x-\alpha), 0\right),\ J\left((x+\alpha)(x-\gamma), 0\right)\rangle,\\
G_2 &= \langle J\left((x+1)(x+\alpha), 0\right),\ J\left((x-\alpha)(x-\gamma), 0\right)\rangle,\\
G_3 &= \langle J\left((x+1)(x+\alpha), 0\right),\ J\left((x-\alpha)(x-\beta), 0\right)\rangle,\\
G_4 &= \langle J\left((x+1)(x-\alpha), 0\right),\ J\left((x+\alpha)(x-\beta), 0\right)\rangle,\\
G_5 &= \langle J\left((x+1)(x-\alpha), 0\right),\ J\left((x+\alpha)(x-\gamma), 0\right)\rangle,\\
G_6 &= \langle J\left((x-1)(x+\alpha), 0\right),\ J\left((x-\alpha)(x-\gamma), 0\right)\rangle,\\
G_7 &= \langle J\left((x-1)(x+\alpha), 0\right),\ J\left((x-\alpha)(x-\beta), 0\right)\rangle.
\end{aligned}$$

These are precisely the 8 groups from the statement of the lemma. □

In our algorithm, we want to use the explicit description of the different $(2, 2)$-groups from Lemma 17. Therefore, on the 2-torsion level, we need to have a symplectic basis as in Lemma 16. This motivates the following definition.

Definition 3 For a genus-2 curve $\mathcal{C}$ defined by a Type-2 equation, we say that a symplectic basis $\mathcal{B} = (T_1, T_2, T_3, T_4)$ of $\mathcal{J}(\mathcal{C})[2^n]$ is a *special symplectic basis* if $2^{n-1} \cdot \mathcal{B} = (2^{n-1}T_1, 2^{n-1}T_2, 2^{n-1}T_3, 2^{n-1}T_4)$ is the basis from Lemma 16.

Note that a special symplectic basic exists for any genus-2 curve $\mathcal{C}$ defined by a Type-2 equation. However, it is in general not unique. For the case $n = 1$ the basis from Lemma 16 is the only special symplectic basis. For $n > 1$, a special symplectic basis can be constructed as follows. Starting with some symplectic basis $\mathcal{B}$ for $\mathcal{J}(\mathcal{C})[2^n]$, one first computes a base change from the 2-torsion basis $2^{n-1}\mathcal{B}$ to the basis from Lemma 16. The base change matrix M is a symplectic matrix over $\mathbb{Z}/2\mathbb{Z}$, hence it can be lifted to a symplectic matrix M' over $\mathbb{Z}/2^n\mathbb{Z}$. Applying M' to the original basis $\mathcal{B}$ then yields a basis with the desired properties.

5.3 Algorithm

We are now ready to describe an algorithm for the efficient computation of $(2^n, 2^n)$-isogenies. This algorithm takes as input any genus-2 curve defined by a Type-2 equation over some finite field K. Moreover it is assumed that the 2^n-torsion of the Jacobian $\mathcal{J}(\mathcal{C})$ is K-rational. A typical example relevant for cryptographic applications is a superspecial hyperelliptic curve $\mathcal{C}$ defined over $K = \mathbb{F}_{p^2}$ with $p \equiv -1 \pmod{2^n}$. In that case Proposition 2 guarantees the existence of a Type-2 equation.

Moreover it is assumed that the $(2^n, 2^n)$-group defining the $(2^n, 2^n)$-isogeny is sampled from the restricted set $\mathcal{G}$ (see Eq. 3) of cardinality 2^{3n} corresponding to a special symplectic basis (T_1, T_2, T_3, T_4) for $\mathcal{J}(\mathcal{C})[2^n]$ as in Definition 3. Note that for cryptographic applications this is not a serious restriction, since $\mathcal{G}$ contains more than half of the $(2^n, 2^n)$-groups of $\mathcal{J}(\mathcal{C})$. Indeed this restriction has already been suggested in the framework of G2SIDH in [15].

Figure 2 provides an overview of the setup and the main steps in the algorithm. Moreover, Fig. 1 in the introduction contains a schematic overview. Using the results and methods developed in this work, all steps in the isogeny chain computation can be performed efficiently. Below we provide some more details on our implementation.

1. The first step only consists of iterative doublings for elements in the Jacobian. There already exist efficient algorithms that were developed in the framework of HECC, see for example [8, 16]. Building on these results, we constructed formulas tailored to Type-2 equations for this computation. Strictly following the algorithm, we need to compute $\frac{n(n-1)}{2}$ such doublings in total. But this number may be decreased by using the alterations described in Remark 5.
   For $i < n$, we also save the 4-torsion element $2^{n-i-1}G_{1,i-1}$ obtained during the computation. This will be used later in Step 3.
2. At a first glance, the second step seems costly since it requires the factorization of two polynomials. Here, we can exploit the properties of the special symplectic basis $\mathcal{B}$. It follows from Lemma 17 that $\alpha_1 \in \{\pm 1\}$. This allows us to find α_1, α_2 by a simple case distinction. Further Lemma 17 implies that $\beta_1 = -\alpha_2$, hence β_2 can be easily computed from the coefficients of the polynomial g_2.
3. For the third step, the case $k = n$ has to be treated separately. If $k < n$, we use the coordinate transformation provided in the proof of Proposition 9.
   In the last step, this proposition cannot be applied since we do not have a 4-torsion point. Therefore the last round necessitates one square root computation to obtain a suitable coordinate transformation. Note that the structure of $\mathcal{J}(\mathcal{C}_n)(K)$ guarantees that this square root is contained in K, so it is not necessary to pass to a field extension. A

Setup We fix a finite field K, an integer n and a genus-2 curve $\mathcal{C}_0$ defined by a Type-2 hyperelliptic equation

$$\mathcal{C}_0 : y^2 = (x^2 - 1)(x^2 - A_0)(E_0x^2 - B_0x + C_0)$$

for some $A_0, B_0, C_0, E_0 \in K$ such that the $\mathcal{J}(\mathcal{C})[2^n]$ is K-rational; and choose a special symplectic basis (T_1, T_2, T_3, T_4) for $\mathcal{J}(\mathcal{C})[2^n]$.

Random Sampling To randomly select a $(2^n, 2^n)$-isogeny, three elements $a, b, c \in \mathbb{Z}/2^n\mathbb{Z}$ are chosen and the elements

$$G_{1,0} = T_1 + aT_3 + bT_4, \quad G_{2,0} = T_2 + bT_3 + cT_4$$

are computed.

Isogeny Chain The following procedure computes an isogeny $\phi : \mathcal{J}(\mathcal{C}_0) \to \mathcal{J}(\mathcal{C}_n)$ with kernel $\langle G_{1,0}, G_{2,0} \rangle$.
For $1 \leq i \leq n$, perform the following steps.

1. Compute $G_1^* = 2^{n-i}G_{1,i-1}$, $G_2^* = 2^{n-i}G_{2,i-1}$ and denote $G_1^* = J(g_1, 0)$, $G_2^* = J(g_2, 0)$.
2. Compute the roots of g_1, g_2 (using the properties of the special symplectic basis) and denote $g_1 = (x - \alpha_1)(x - \alpha_2)$, $g_2 = (x - \beta_1)(x - \beta_2)$.
3. Perform a coordinate change $(x', y') = t(x, y)$ to obtain a Type-1 equation

$$\mathcal{C}'_{i-1} : y'^2 = E'_{i-1}\, x' \,(x'^2 - A'_{i-1}x' + 1)(x'^2 - B'_{i-1}x' + C'_{i-1})$$

   satisfying $t(g_1) = x'$ and $t(g_2) = x'^2 - A'_{i-1}x' + 1$.
4. If $C'_{i-1} = 1$, abort. Otherwise, apply the Richelot isogeny $\tilde{\phi}_i : \mathcal{J}(\mathcal{C}'_{i-1}) \to \mathcal{J}(\mathcal{C}_i)$ from Proposition 10 to obtain a Type-2 equation

$$\mathcal{C}_i : y^2 = (x^2 - 1)(x^2 - A_i)(E_ix^2 - B_ix + C_i)$$

   and the formula from Theorem 15 to compute $G_{1,i} = \phi_i(G_{1,i-1})$, $G_{2,i} = \phi_i(G_{2,i-1})$ with $\phi_i = \tilde{\phi}_i \circ t$.

Then $\mathcal{J}(\mathcal{C}_n)$ is the codomain of the isogeny ϕ.

Fig. 2 Overview of our method for computing $(2^n, 2^n)$-isogenies

possible modification to avoid the square root computation in the last round is discussed in Remark 4.

4. The fourth step consists of applying the formulas from Proposition 10 once to obtain the coefficients for the new Type-2 equation, and the formula from Theorem 15 has to be applied twice to compute the images of the kernel generators.

Remark 3 Note that our formulas can only be applied if the isogeny chain does not contain any split isogenies. In other words, we require $\delta_i = C'_{i-1} - 1 \neq 0$ in each step i (cf. Proposition 10). In the case that $\delta = 0$, the algorithm aborts. In cryptographically relevant settings, where the curve $\mathcal{C}_0$ is superspecial and $n \approx \log(p)/2$, this happens with probability approximately $\log(p)/p$ for a randomly chosen isogeny chain, see for example [9, Section 5].

On the other hand, the splitting case, $\delta = 0$, plays an important role in the SIDH attack described in [2]. However, for the attack to work, one only needs to detect whether the last isogeny ϕ_n splits. For this purpose, it suffices to compute the isogenies $\phi_1, \ldots, \phi_{n-1}$ explicitly and return the value of δ_n. Clearly, this is possible using our methods.

For more efficient versions of the attack as proposed in [18], one also needs to be able to evaluate splittings and gluings. Explicit maps for the splitting and gluing isogeny can be derived from the covering maps implied by the second part of Proposition 10.

Remark 4 For the last $(2, 2)$-isogeny in the isogeny chain, the above algorithm requires one square root computation in the execution of Step 3. This computation can be avoided by slightly changing the setup. For example, one can choose a curve $\mathcal{C}$ such that $\mathcal{J}(\mathcal{C})[2^{n+1}]$ is K-rational and provide the kernel G for a $(2^{n+1}, 2^{n+1})$-isogeny, but consider only the $(2^n, 2^n)$-isogeny defined by $2 \cdot G$. In other words, the last step of the isogeny computation is omitted. In the superspecial case, this necessitates to increase the size of the underlying prime field by two bits.

Remark 5 Running the algorithm as described above, requires to perform $\frac{n(n-1)}{2}$ point doublings in total, since in each step $i \in \{1, \ldots, n\}$, one has to compute the kernel generators of the current isogeny $G_1^* = 2^{n-i} G_{1,i-1}$ and $G_2^* = 2^{n-i} G_{2,i-1}$. Note that

$$G_1^* = \psi_{i-1}(2^{n-i} G_1), \quad G_2^* = \psi_{i-1}(2^{n-i} G_2).$$

This observation provides a different way of computing G_1^* and G_2^* which reduces the total number of doublings. More precisely, in the beginning of the algorithm one computes a list containing

$$H_{1,i} = 2^{n-i} G_1, \quad H_{2,i} = 2^{n-i} G_2 \quad \text{for } i \in \{1, \ldots, n\}.$$

At each step, one additionally computes the image $\phi_i(H_{1,j})$ for all $j \geq i$ so that G_1^* and G_2^* can always be recovered without performing any additional point doublings. While this procedure reduces the number of doublings to n, it increases the number of point image computations by $n(n-1)$.

For optimal performance, one should use a mix of both methods. Using a classical divide-and-conquer method, both the number of point image computations as well as the number of doublings is around $n \log(n)$.[6]

Note that such a strategy was already developed in [11, Section 4.2.2] in the elliptic curve setting. Further, in that setting the authors determine optimal parameters for specific values of n that minimize the number of arithmetic operations.

5.4 Implementation

Implementations of our algorithm in Magma [1] and SageMath [26] are made available in our GitHub repository [14]. Here, we compare its efficiency to related results in the

[6] Thanks to Luca de Feo for pointing this out. In a previous version of the manuscript, we only described a method reducing the number of doublings to $n^{3/2}$.

Table 2 Runtime in seconds for different algorithms computing a $(2^n, 2^n)$-isogeny

	$n = 51$, $\log(p) \approx 100$		$n = 86$, $\log(p) \approx 171$	
	pure isogeny	image points	pure isogeny	image points
Genus-2 SIDH (Magma) [12]	36	75	omitted	omitted
SIDH attack (SageMath) [19]	0.39	0.58	0.80	1.13
SIDH attack (Magma) [2]	0.15	0.24	0.37	0.55
This work (SageMath)	0.16	0.22	0.33	0.43
This work (Magma)	0.05	0.08	0.12	0.16

literature. For that comparison, we use a setup which is typical for applications in isogeny-based cryptography. This means that we consider a prime of the form $p = e \cdot 2^n 3^m - 1$ with $2^n \approx 3^m$ and a small integer e. We choose a superspecial genus-2 curve $\mathcal{C}$ defined over $\mathbb{F}_{p^2}$ so that $\mathcal{J}(\mathcal{C})[2^n] \subset \mathcal{J}(\mathcal{C})(\mathbb{F}_{p^2})$ and compute a $(2^n, 2^n)$-isogeny. Since this is necessary, for some applications, we also compute the image of a set consisting of four elements of $\mathcal{J}(\mathcal{C})$ under this isogeny. The comparison is done on two different instances for typical cryptographic parameters. For instance, in G2SIDH these parameter choices correspond to a (previously assumed) classical security of 75 bits and 128 bits respectively.

All computations were performed on our platform, a laptop with an Intel i7-8565U processor and 16 GB of RAM with Linux 5.13.0, Magma V2.27.5, and SageMath 9.7. We compare our algorithm with the Richelot isogeny computations in the following implementations:

- Genus-2 SIDH by Flynn and Ti: The authors of [12] kindly provided their source code of the implementation of the genus-2 SIDH protocol. While this contains the first available implementation of $(2^n, 2^n)$-isogenies, it is to be understood as a proof-of-concept implementation.
- SIDH attack by Castryck and Decru: The computation of $(2^n, 2^n)$-isogenies is an the important part of the attack on SIDH and an implementation in Magma is provided along with their paper [2]. Here, a significant speedup with respect to the G2SIDH implementation is obtained by replacing Algorithm 1 by a Gröbner basis computation. For more details, the reader is referred to [2, Sect. 6.2]. This work is concurrent to ours.
- SageMath implementation by Oudompheng and Pope: There also exists a version of the SIDH attack implemented in SageMath [19]. The algorithm for the computation of the $(2^n, 2^n)$-isogenies deviates from that in [2]. In particular the Gröbner basis computation is replaced by a more explicit approach. Furthermore, similar as in our implementation, the number of doublings in each chain is reduced by applying the strategy outlined in Remark 5.

The results are summarized in Table 2.

Acknowledgements I would like to thank Tanja Lange for helpful discussions and various suggestions to improve the manuscript, as well as Thomas Decru for helpful comments on an earlier version of the paper. Thanks also go to Yan Bo Ti for sharing his code for the computation of $(2^n, 2^n)$-isogenies in G2SIDH. The author was supported by the DFG under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability Not applicable.

Declarations

Code availability All code relevant to this work is publicly available in our GitHub repository [14].

Appendix A: Special cases of the (2, 2)-isogeny formula

In this section, we treat the special cases that are not covered by Theorem 15 or Propositions 12 and 14. For the entire section, we assume that we are in the setting of Proposition 10, Case 1. This means, we consider the isogeny $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$, where $\mathcal{C}$ and $\mathcal{C}'$ are hyperelliptic curves defined as

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

and

$$\mathcal{C}' : y^2 = (x^2 - 1)(x^2 - A')(E'x^2 - B'x + C'),$$

with

$$A' = C, \quad B' = \frac{2}{E}, \quad C' = \frac{B - AC}{E(1 - C)}, \quad E' = \frac{A - B}{E(1 - C)},$$

and $\ker(\phi) = \langle J(x, 0), J(x^2 - Ax + 1, 0)\rangle \subset \mathcal{J}(\mathcal{C})[2]$. Some of the computations in this section are quite tedious to perform by hand and we recommend to use our code available at [14] or in Section B for verification.

A.1: Divisors supported at Weierstrass points

First, we consider the cases, where the divisor $D(a, b) \in \mathrm{Div}(\mathcal{C})$ is supported on a Weierstrass points of $\mathcal{C}$. This is very similar to the situation where $a = x + a_0$ is a degree-1 polynomial which is treated in Proposition 14. The next lemma provides an easy check for this property.

Lemma 18 *Let $\mathcal{C} : y^2 = f(x)$ be a genus-2 curve and $J(a, b) \in \mathcal{J}(\mathcal{C})$ with $(a, b) = (x^2 + a_1x + a_0, b_1x + b_0)$. Then*

$$-b_1(a_1b_0 - a_0b_1) + b_0^2 = 0$$

if and only if the support of $D(a, b)$ contains a Weierstrass point of $\mathcal{C}$.

Proof Note that $-b_1(a_1b_0 - a_0b_1) + b_0^2$ is the resultant of a and b. The resultant vanishes if and only if there exists a common root $u \in \bar{K}$. In this case, $P = (u, 0)$ lies in the support of $D(a, b)$. □

Corollary 19 *Let $J(a,b) \in \mathcal{J}(\mathcal{C})$ with $(a,b) = (x^2+a_1x+a_0, b_1x+b_0)$ satisfying $b_1(a_1b_0 - a_0b_1) + b_0^2 = 0$, $b \neq 0$ and $a_0B^2 + (a_0+1)a_1B + (a_0-1)^2 + a_1^2 \neq 0$. Then $D(a,b) = (u,v) + (r,0)$, where*

$$r = -\frac{b_0}{b_1}, \quad u = -a_1 - \frac{b_0}{b_1}, \quad v = -a_1b_1;$$

and $\phi(J(a,b)) = [D_P + D_Q - 2D'_\infty]$, where D_P is the divisor from Lemma 13 and $D_Q = D(a_Q, 0)$ with

$$a_Q = \begin{cases} x^2 - A' & \text{if } r = 0, \\ E'x^2 - B'x + C' & \text{if } r^2 - Ar + 1 = 0, \\ 1 & \text{if } r^2 - Br + C = 0. \end{cases}$$

Proof This is a consequence of Propositions 12 and 14. □

For the case $a_0B^2 + (a_0+1)a_1B + (a_0-1)^2 + a_1^2 = 0$, we refer to Proposition 23. Moreover, we excluded the case $b = 0$, which happens if and only if $J(a,b) \in \mathcal{J}(\mathcal{C})[2]$. The formulas for $\phi(J(a,b))$ in this case can be easily extracted from Proposition 12. While we leave this to the reader, we observe that $\langle J(x^2-1,0), J(x^2-A',0)\rangle$ defines the dual isogeny $\hat{\phi} : \mathcal{J}(\mathcal{C}') \to \mathcal{J}(\mathcal{C})$. This is implied by the corollary below.

Corollary 20 *Let $J(a,b) \in \mathcal{J}(\mathcal{C})$ with $(a,b) = (x^2 + a_1x + a_0, b_1x + b_0)$ satisfying $-b_1(a_1b_0 - a_0b_1) + b_0^2 = 0$ and $b = 0$. Then*

$$\phi(J(a,b)) \in \langle J(x^2-1,0), J(x^2-A',0)\rangle.$$

Proof This is a consequence of Propositions 12. □

A.2: Image points supported at infinity

The curve $\mathcal{C}'$ is defined by a degree-6 equation,

$$\mathcal{C}' : y^2 = (x^2-1)(x^2-A')(E'x^2 - B'x + C'),$$

hence has two points at infinity.[7] Let us fix an element $e' \in \bar{K}$ satisfying $e'^2 = E'$, then the projective coordinates for the points at infinity are $\infty_+ = (1 : e' : 0)$ and $\infty_- = (1 : -e' : 0)$. In this context, we denote $\mathrm{sgn}(e') = +1$ and $\mathrm{sgn}(-e') = -1$. As opposed to the divisor $D'_\infty = \infty_+ + \infty_-$, the points ∞_+, ∞_- are not necessarily K-rational. But in case they are rational, we need to deal with elements on the Jacobian $\mathcal{J}(\mathcal{C}')$ of the form $[P - \infty_+] = [P + \infty_- - D'_\infty]$ and $[P - \infty_-] = [P + \infty_+ - D'_\infty]$. We therefore introduce the notation

$$J(x + a_0, b_0, -) = [(-a_0, b_0) + \infty_- - D'_\infty],$$

and

$$J(x + a_0, b_0, +) = [(-a_0, b_0) + \infty_+ - D'_\infty].$$

Similarly, we denote

$$D(a,b,+) = D(a,b) + \infty_+, \quad D(a,b,-) = D(a,b) + \infty_-.$$

[7] For the sake of simplicity, we assume $A \neq B$ so that $E' = \frac{A-B}{E(1-C)} \neq 0$ here. But the reader can convince themselves that the formulas for $A = B$ are very similar.

Note that these cases are not captured by the notation introduced in Sect. 2.2.

The next two propositions describe cases, where the image of an element $J(a,b) \in \mathcal{J}(\mathcal{C})$ under the isogeny ϕ is of the special form described above. In other words, $\phi(J(a,b)) = J(a',b',\pm)$. It is easy to see from the description of the Richelot correspondence (Proposition 11) that this happens if and only if $\gcd(a, x^2 - Bx + 1) \neq 1$. First, we treat the case, where $a = x + a_0$ is a factor of $x^2 - Bx + 1$ (Proposition 21). Then we consider the cases, where $a = x^2 + a_1 x + a_0$. Lemma 22 provides an easy criterion to check, whether $\gcd(a, x^2 - Bx + 1) \neq 1$. We distinguish two cases. Proposition 23 deals with the case where $\gcd(a, x^2 - Bx + 1)$ has degree 1. This implies that a has two K-rational roots, which can be computed using the Euclidean algorithm. This allows to determine two rational divisors $D_P, D_Q \in \mathrm{Div}(\mathcal{C}')$ such that $\phi(J(a,b)) = [D_P + D_Q - 2D'_\infty]$. The case $a = x^2 - Bx + 1$ is treated in Proposition 24. Here, some interesting configurations occur. For example if $b = b_1 x$, then $\phi(J(a,b)) \in \pm[\infty_+ - \infty_-]$.

Proposition 21 *Let $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ as described above. Let $J(a,b) \in \mathcal{J}(\mathcal{C})$ satisfying $a = x + a_0$ and $a_0^2 + Ba_0 + 1 = 0$, then*

$$\phi(J(a,b)) = \left[D_P + D_Q - 2D'_\infty\right] \in \mathcal{J}(\mathcal{C}'),$$

where $D_Q = (x^2 - C, 0)$ and

$$D_P = D\left(x - \frac{B}{2}, \frac{(4C - B^2)(A - B)}{8} \frac{a_0(B + 2a_0)}{b_0}, \operatorname{sgn}\left((B - A)\frac{a_0}{b_0}\right)\right).$$

Proof We proceed similarly as in the proof of Proposition 12. To summarize, we have $\mathcal{J}(x + a_0, b_0) = [(-a_0, b_0) - \infty]$, hence

$$\phi(J(a,b)) = \psi((-a_0, b_0)) - \psi(\infty),$$

where $\psi : \mathcal{C} \to \mathcal{J}(\mathcal{C}')$ is the map induced by the Richelot correspondence $\mathcal{R}$ in Proposition 11 with respect to some embedding $\iota : \mathcal{C}' \to \mathcal{J}(\mathcal{C}')$, $P \mapsto [P - P']$. It holds that $\psi(\infty) = [D(x^2 - C, 0) - 2P']$.

The computation of $\psi((-a_0, b_0)) = [D_P - 2P']$ however differs from that in Proposition 12. Inserting the coordinates of $P = (u, v)$ into the equation from the Richelot correspondence 11, we find that there is only one (affine) solution $u_1 = \frac{B}{2}$. The second solution is $u_2 = \infty$. The corresponding y-coordinates can be determined from the second equation of the Richelot correspondence. We obtain

$$v_1 = \frac{(4C - B^2)(A - B)}{8} \frac{a_0(B + 2a_0)}{b_0}, \quad v_2 = (B - A)\frac{a_0}{b_0}.$$

Note that v_2 is indeed a square root of E', the leading coefficient of the hyperelliptic equation for $\mathcal{C}'$, in particular $v_2 = \pm e'$. We denote $\operatorname{sgn}(v_2) \in \{\pm\}$ for the sign of v_2. This means

$$D_P = (u_1, v_1) + \infty_{\operatorname{sgn}(v_2)} = D(x - u_1, v_1, \operatorname{sgn}(v_2)).$$

□

Lemma 22 *Let $a = x^2 + a_1 x + a_0$ and $g = x^2 - Bx + 1$ be polynomials in $K[x]$. Then $\gcd(a, g) \neq 1$ if and only if*

$$a_0 B^2 + (a_0 + 1)a_1 B + (a_0 - 1)^2 + a_1^2 = 0.$$

Proof The expression above is the resultant of the polynomials a and g. □

Proposition 23 *Let $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ as described above. Let $J(a,b) \in \mathcal{J}(\mathcal{C})$ with $a = x^2 + a_1 x + a_0$ satisfying $\gcd(a, x^2 - Bx + 1) = (x - s)$ and write $t = b_1 s + b_0$. Then*

$$\phi(J(a,b)) = [D - 2D'_\infty], \quad \text{where } D = D_P + D_Q$$

with

$$D_Q = D\left(x - \frac{B}{2}, \frac{(4C - B^2)(B - A)}{8}\frac{s(B-2s)}{t}, \operatorname{sgn}\left((A-B)\frac{s}{t}\right)\right)$$

and D_P is as described below.

1. *If $a = (x-u)(x-s)$ with $s \neq u$, then D_P is the divisor from Lemma 13; unless $P = (u, 0)$ is a Weierstrass point, in which case $D_P = D(a_P, 0)$ with $a_P \in \{1, x^2 - A', E'x^2 - B'x + C'\}$ as in Corollary 19.*
2. *If $a = (x-s)^2$, then $D(a,b) = 2 \cdot (s,t)$ and $D_P = D_Q$.*

Proof In Case 1, $\gcd(a, x^2 - Bx + 1) = x - s$ for some $s \in K$. We write $a = (x-u)(x-s)$ and $v = b(u)$, $t = b(s)$. Then $P = (u, v)$ and $Q = (s, t)$. For the point P the divisor D_P is described in Lemma 13 or Corollary 19 depending on whether P is a Weierstrass point. For $Q = (s,t)$ the computation is identical to the proof of Proposition 21, when setting $a_0 = -s$ and $b_0 = t$.

In the second case $P = Q = (s,t)$ and the result follows from the first case. □

Proposition 24 *Let $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ as described above, and $J(a,b) \in \mathcal{J}(\mathcal{C})$ with $a = x^2 - Bx + 1$.*

1. *If $B = \pm 2$, then $D(a,b) = 2 \cdot (\pm 1, b_0)$ and $\phi(J(a,b)) = 2 \cdot J\left(x \mp 1, 0, \mp \operatorname{sgn}\left(\frac{A-2}{b_0}\right)\right)$.*

Otherwise, when $B \neq \pm 2$, the following cases occur.

2. *If $b_0 = 0$, then $(A-B)^2 b_1^2 = E'$ and $\phi(J(a,b)) = s \cdot [\infty_+ - \infty_-]$, where $s = \operatorname{sgn}((A-B)b_1)$.*
3. *If $b_0 \neq 0$, then $\phi(J(a,b)) = J\left(\left(x - \frac{B}{2}\right)^2, \frac{(4C-B^2)(B-A)}{4b_0}\right)$.*

Proof Let us write $a = x^2 - Bx + 1 = (x-u)(x-s) \in \bar{K}[x]$. We denote $v = b(u)$ and $t = b(s)$, hence $D(a,b) = P + Q$ with $P = (u,v)$ and $Q = (s,t)$. Similar as in Case 1 of Proposition 23, we find that $\phi(J(a,b)) = D_P + D_Q$, where

$$D_P = D(x - B/2, v_1, \operatorname{sgn}(v_2)), \quad \text{and} \quad D_Q = D(x - B/2, t_1, \operatorname{sgn}(t_2)),$$

with

$$v_1 = \frac{(4C - B^2)(B - A)}{8}\frac{u(B-2u)}{v}, \quad v_2 = (A-B)\frac{u}{v}.$$

and

$$t_1 = \frac{(4C - B^2)(B - A)}{8}\frac{s(B-2s)}{t}, \quad t_2 = (A-B)\frac{s}{t}.$$

If $B = \pm 2$, then $D_P = D_Q = P_1 + \infty_{\operatorname{sgn}(v_2)}$ are K-rational. The image $\phi(J(a,b))$ is easily computed by inserting $B = \pm 2$ everywhere.

From now on we assume $B \neq \pm 2$, hence $u \neq s$. In that case $D_P = P_1 + \infty_{\operatorname{sgn}(v_2)}$ and $D_Q = Q_1 + \infty_{\operatorname{sgn}(t_2)}$ are K-rational. In order to compute their composition $D_P + D_Q$, note that $t_1 = \pm v_1$ and $t_2 = \pm v_2$, since these points share the same x-coordinate on $\mathcal{C}'$.

If $b_0 = 0$, then $u/v = s/t$, hence $v_2 = t_2$, and

$$\frac{v_1}{t_1} = \frac{u(B-2u)t}{s(B-2s)v} = \frac{B-2u}{B-2s} = -1.$$

This means that $P_1 = \tau(Q_1)$, where τ is the hyperelliptic involution, hence $[P_1+Q_1-D'_\infty] = 0$. And $[\infty_{\mathrm{sgn}(v_2)} + \infty_{\mathrm{sgn}(t_2)} - D'_\infty] = s \cdot [\infty_+ - \infty_-] \in \mathcal{J}(\mathcal{C}')$, where $s = \mathrm{sgn}((A-B)b_1)$.

Otherwise if $b_0 \neq 0$, then, we have $v_1 = t_1$ and $v_2 = -t_2$. In that case $[\infty_{\mathrm{sgn}(v_2)} + \infty_{\mathrm{sgn}(t_2)} - D'_\infty] = 0$ and we find that

$$P_1 = Q_1 = \left(\frac{B}{2}, \frac{(4C-B^2)(B-A)}{4b_0}\right).$$

□

A.3: Shared support

Let $J(a,b) = [P+Q-D_\infty]$ and let $D_P = D(a_P, b_P)$ and $D_Q = D(a_Q, b_Q)$ be the divisors associated to P and Q under the Richelot correspondence. In the last step of Algorithm 1, the composition $D = D(a', b')$ of D_P and D_Q is computed. In most cases a_P and a_Q are coprime, so that $a' = a_P \cdot a_Q$. In this part, we take care of the cases where this is not true. First, we provide a criterion to distinguish this scenario from the general case (Lemma 25). This criterion shows that there are two subcases which are covered in Propositions 26 and 27 respectively.

Lemma 25 *Let $J(a,b) = [P+Q-D_\infty] \in \mathcal{J}(\mathcal{C})$. Consider the map $\pi'_* \circ \pi^* : \mathcal{C} \to \mathrm{Div}(\mathcal{C}')$ induced by the Richelot correspondence. Denote $D_P = D(a_P, b_P) = \pi'_* \circ \pi^*(P)$ and $D_Q = D(a_Q, b_Q) = \pi'_* \circ \pi^*(Q)$. We assume that $a_0 B^2 + (a_0+1)a_1 B + (a_0-1)^2 + a_1^2 \neq 0$. Then the $\gcd(a_P, a_Q) \neq 1$ if and only if $a_0 = 1$ or $a_1^2 = 4a_0$. Moreover, in these cases $a_P = a_Q$.*

Proof Denote $P = (u, v)$ and $Q = (s, t)$. Since $a_0 B^2 + (a_0+1)a_1 B + (a_0-1)^2 + a_1^2 \neq 0$, we are not in the situation of Section A.2. In particular, $u^2 - Bu + 1 \neq 0$ and $s^2 - Bs + 1 \neq 0$, hence the first relation in the Richelot correspondence yields

$$a_P = x^2 + \frac{2(C-1)u}{u^2 - Bu + 1} \cdot x + \frac{-Cu^2 + Bu - C}{u^2 - Bu + 1},$$
$$a_Q = x^2 + \frac{2(C-1)s}{s^2 - Bs + 1} \cdot x + \frac{-Cs^2 + Bs - C}{s^2 - Bs + 1}.$$

The resultant of a_P and a_Q is

$$\mathrm{res}(a_P, a_Q) = (u-s)^2(us-1)^2 \cdot \frac{(C-1)^2(4C-B^2)}{(u^2 - Bu + 1)^2(s^2 - Bs + 1)^2},$$

which is zero if and only if $u = s$ or $u = 1/s$. Translated to the Mumford coordinates of $D = (u, v) + (s, t)$, this means that $a_1^2 = 4a_0$ or $a_0 = 1$.

If $u = s$, it is clear that $a_P = a_Q$. If $u = 1/s$, then

$$a_P = x^2 + \frac{2(1-C)}{B + a_1} x - \frac{B + a_1 C}{B + a_1} = a_Q.$$

□

Proposition 26 *Let $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ as described above and $J(a,b) \in \mathcal{J}(\mathcal{C})$ with $a = x^2 + a_1x + a_0$, $b = b_1x + b_0$ and $4a_0 = a_1^2$. Then,*

$$\phi(J(a,b)) = [2D(a_P, b_P) - 2D'_\infty] \in \mathcal{J}(\mathcal{C}'),$$

where $D(a_P, b_P)$ is as in Lemma 13 with $(u,v) = \left(-\frac{a_1}{2}, b_0\right)$.

Proof Clearly $J(a,b) = [2P - D_\infty]$, where $P = \left(-\frac{a_1}{2}, b_0\right)$. This implies $\phi(J(a,b)) = [2D_P - 2D'_\infty]$, where D_P is as in Lemma 13. □

Proposition 27 *Let $\phi : \mathcal{J}(\mathcal{C}) \to \mathcal{J}(\mathcal{C}')$ as described above. Let $J(a,b) \in \mathcal{J}(\mathcal{C})$ with $a = x^2 + a_1x + 1$, $b_1x + b_0$ (i.e. $a_0 = 1$) and assume $a_1 \notin \{2, -A, -B\}$, $-b_1(a_1b_0 - b_1) + b_0^2 \neq 0$, $a_0B^2 + (a_0+1)a_1B + (a_0-1)^2 + a_1^2 \neq 0$. Then $\phi(J(a,b)) = [2P - D_\infty]$, where*

$$P = \left(\frac{d_0}{d_1}, \frac{(B^2 - 4C)(C-1)(a_1 + A)}{d_1}\right),$$

with

$$\begin{aligned} d_0 &= (B(b_1 - a_1b_0) + 2b_0C)(a_1 + B) - 2b_0B(C-1), \\ d_1 &= (2(b_1 - a_1b_0) + b_0B)(a_1 + B) - 4b_0(C-1). \end{aligned}$$

Proof Let $J(a,b) = [P + Q - D_\infty] \in \mathcal{J}(\mathcal{C})$. Consider the map $\pi'_* \circ \pi^* : \mathcal{C} \to \mathrm{Div}(\mathcal{C}')$ induced by the Richelot correspondence and denote $D_P = D(a_P, b_P) = \pi'_* \circ \pi^*(P)$ and $D_Q = D(a_Q, b_Q) = \pi'_* \circ \pi^*(Q)$. As per Lemma 25, $a_P = a_Q$ and we denote

$$a_P = a_Q = (x - u_1)(x - u_2) \in \bar{K}[x].$$

In order to compute the composition $D_P + D_Q$, we show that $b_P \neq b_Q$ and compute the intersection of the two polynomials.

Using the presentation for b_P and b_Q from Lemma 13, we deduce that

$$\begin{aligned} b_{P,0} - b_{Q,0} &= -\frac{(C-1)(A + a_1)(2u + a_1)}{(B + a_1)^2 vt}\, d_0, \\ b_{P,1} - b_{Q,1} &= \frac{(C-1)(A + a_1)(2u + a_1)}{(B + a_1)^2 vt}\, d_1. \end{aligned}$$

One can show that $d_1 \neq 0$ in our setting, hence $b_{P,1} \neq b_{Q,1}$ and b_P and b_Q intersect in a unique point

$$S = (\hat{x}, \hat{y}) = \left(\frac{d_0}{d_1}, \frac{(B^2 - 4C)(C-1)(a_1 + A)}{d_1}\right).$$

Moreover, we find that $a_P(\hat{x}) = 0$, hence S is a point in the support of both D_P and D_Q. Taking into account that $b_P \neq b_Q$, we deduce that $D_P = S + P_2$ and $D_Q = S + Q_2$ with $P_2 = (u_2, v_2)$ and $Q_2 = (u_2, -v_2)$, for some $u_2 \in \bar{K}$. Consequently,

$$[D_P + D_Q - 2D'_\infty] = [2(\hat{x}, \hat{y}) + (u_2, v_2) + (u_2, -v_2) - 2D'_\infty] = [2(\hat{x}, \hat{y}) - D'_\infty].$$

□

Appendix B: Verification of the formulas

In the following, we provide SageMath code that can be used to verify the formulas obtained in this work. This code is also made available in our GitHub repository [14].

B.1: Proofs for Sect. 2

```
print("Corollary 5")
R.<r1,r2,r3,r4> = PolynomialRing(QQ)
s1 = r1+r2+r3+r4
s2 = r1*r2 + r1*r3 + r1*r4 + r2*r3 + r2*r4 + r3*r4
s3 = r1*r2*r3 + r1*r2*r4 + r1*r3*r4 + r2*r3*r4
s4 = r1*r2*r3*r4
b1 = -r1^2
b2 = -r2^2
print(r1*r2 == (s1*s3*b1*b2 + (s4-b1*b2)^2) / (b1*b2*s1^2
+ (s4-b1*b2)*(s2-b1-b2)))
```

B.2: Proofs for Sect. 4

```
def Richelot(G, delta):
Gd = [g.derivative() for g in G]
H = [(Gd[(i+1)
return H

#Type 1 Equation:
R.<A,B,C,E,u,v> = QQ[]
S.<x> = Frac(R)[]
F = E*x*(x^2-A*x+1)*(x^2-B*x+C)
G = [E*x,x^2-A*x+1, (x^2-B*x+C)]

print("Proposition 10")
delta = -E*(1-C)
H = Richelot(G,delta);
Ap = C
Bp = 2/E
Cp = (B-A*C)/(E*(1-C))
Ep = (A-B)/(E*(1-C))
print(prod(H)*(1-C)^2 == (x^2-1)*(x^2-Ap)*(Ep*x^2-Bp*x+Cp))

print("Proposition 11")
P.<up> = Frac(R)[]
rel1 = (G[0](u)*H[0](up)+G[1](u)*H[1](up))*(1-C);
rel2 = (G[0](u)*H[0](up))*(u-up)*(1-C);  #rel2=(1-C)*v'*v
print(rel1 == -(u^2-B*u+1)*up^2 - 2*(C-1)*u*up + C*u^2-B*u+C)
print(rel2 == (A-B)*u*up^3 - ((A-B)*u^2+2*(1-C)*u)* up^2
+ (2*(1-C)*u^2 - (A*C-B)*u)*up + (A*C-B)*u^2)

print("Lemma 13")
aP1 = 2*(C-1)*u/(u^2-B*u+1)
aP0 = (-C*u^2+B*u-C)/(u^2-B*u+1)
print(rel1/(-u^2+B*u-1) == up^2 + aP1*up + aP0)
```

```
bP1 = u*(1-C)*(u^2-A*u+1)/(u^2-B*u+1)^2 * (2*u^3-B*u^2
+ (-B^2+4*C-2)*u+B)
bP0 = -u*(1-C)*(u^2-A*u+1)/(u^2-B*u+1)^2 * (B*u^3
+(-B^2+2*C)*u^2 - B*u+2*C)
print(rel2

print("Theorem 15")
K.<A,B,C,E,u,a0,a1,b0,b1> = QQ[]
R.<x> = K[]
#Relations among the elements
# 1) u is a root of a(x) = x^2+a1*x+a0
# 2) a0,a1,b0,b1 describe a divisor on the curve y^2
= x(x^2-Ax+1)(x^2-Bx+C)
rel1 = u^2 + a1*u + a0
F = E*x*(x^2-A*x+1)*(x^2-B*x+C)

b = b1*x+b0
a = x^2+a1*x+a0
[q,r] = (F-b^2).quo_rem(a) #r must be zero
relations = [rel1] + r.coefficients()
I = K.ideal(relations)
v = b0 + b1*u
s = -a1 - u
t = b0 + b1*s
#expressions for aP, bP from above
aP1 = 2*(C-1)*u/(u^2-B*u+1)
aP0 = (-C*u^2+B*u-C)/(u^2-B*u+1)
bP1 = u*(1-C)*(u^2-A*u+1)/(u^2-B*u+1)^2/v * (2*u^3-B*u^2
+ (-B^2+4*C-2)*u+B)
bP0 = -u*(1-C)*(u^2-A*u+1)/(u^2-B*u+1)^2/v * (B*u^3
+(-B^2+2*C)*u^2 - B*u+2*C)
aP = x^2+aP1*x+aP0
bP = bP1*x +bP0

aQ1 = 2*(C-1)*s/(s^2-B*s+1)
aQ0 = (-C*s^2+B*s-C)/(s^2-B*s+1)
bQ1 = +s*(1-C)*(s^2-A*s+1)/(s^2-B*s+1)^2/t * (2*s^3-B*s^2
+ (-B^2+4*C-2)*s+B)
bQ0 = -s*(1-C)*(s^2-A*s+1)/(s^2-B*s+1)^2/t * (B*s^3
+(-B^2+2*C)*s^2 - B*s+2*C)
aQ = x^2+aQ1*x+aQ0
bQ = bQ1*x +bQ0

a00 =  a0*B^2 + (a0*a1 + a1)*B*C + (a0^2 + a1^2 - 2*a0
 + 1)*C^2
a11 =  4*a0*B*C + (2*a0*a1 + 2*a1)*C^2 + (-4*a0)*B
+ (-2*a0*a1 - 2*a1)*C
a22 =  (-2*a0)*B^2 + (-a0*a1 - a1)*B*C + 4*a0*C^2
 + (-a0*a1 - a1)*B
```

```
+ (-2*a0^2 - 2*a1^2 - 4*a0 - 2)*C + 4*a0
a33 =  (-4*a0)*B*C + 4*a0*B + (-2*a0*a1 - 2*a1)*C + 2*a0*a1 +
2*a1
aden =  a0*B^2 + (a0*a1 + a1)*B + a0^2 + a1^2 - 2*a0 + 1
ap = (a00 + a11*x + a22*x^2 + a33*x^3+aden*x^4)/aden
acomp = (aP*aQ).coefficients()
print("representation for a':")
print("a0':", K(a00-acomp[0].numerator()).reduce(I) == 0)
print("a1':", K(a11-acomp[1].numerator()).reduce(I) == 0)
print("a2':", K(a22-acomp[2].numerator()).reduce(I) == 0)
print("a3':", K(a33-acomp[3].numerator()).reduce(I) == 0)
print("a4':", all([c.denominator().reduce(I) == aden for c in
acomp[:3]]))

b00 =  (a0*a1*b0 - a0^2*b1)*A*B + (a0^2*b0 + a1^2*b0
- a0*a1*b1
 - a0*b0)*A*C
+ (a0*a1^2*b0 - a0^2*a1*b1 - a0^2*b0 + a0*b0)*B
+ (a0^2*a1*b0 + a1^3*b0 - a0^3*b1 - a0*a1^2*b1
- 2*a0*a1*b0 + 2*a0^2*b1 + a1*b0 - a0*b1)*C
b11 =  a0*b0*A*B + (2*a0*a1*b0 - a0^2*b1 + a1*b0 - a0*b1)*A*C
+ (-2*a0*a1*b0 + 2*a0^2*b1)*A + (a0*a1*b0 - a0^2*b1
 + a0*b1)*B
+ (2*a0*a1^2*b0 - 2*a0^2*a1*b1 - a0^2*b0 + a1^2*b0 + b0)*C
- 2*a0*a1^2*b0 + 2*a0^2*a1*b1 + 2*a0^2*b0 - 2*a0*b0
b22 =  (-a0*a1*b0 + a0^2*b1)*A*B + 2*a0*b0*A*C
+ (-a0^2*b0 - a1^2*b0 + a0*a1*b1 - a0*b0)*A
+ (-a0*a1^2*b0 + a0^2*a1*b1 + a0^2*b0 - a0*b0)*B
+ (2*a0*a1*b0 - 2*a0^2*b1 + 2*a0*b1)*C
- a0^2*a1*b0 - a1^3*b0 + a0^3*b1 + a0*a1^2*b1 - a1*b0 - a0*b1
b33 =  (-a0*b0)*A*B + (-a0^2*b1 - a1*b0 + a0*b1)*A
+ (-a0*a1*b0 + a0^2*b1 - a0*b1)*B - a0^2*b0 - a1^2*b0
+ 2*a0*b0 - b0
bden =  -1*(a0 - 1) * (-a1*b0*b1 + a0*b1^2 + b0^2)
bp = (b33*x^3+b22*x^2+b11*x+b00)/bden
print("representation for b':")
print("b'=bP(mod aP):", all([c.numerator().reduce(I) == 0 for
c in
((bp-bP)
print("b'=bQ(mod aQ):", all([c.numerator().reduce(I) == 0 for
c in
((bp-bQ)

Ap = C
Bp = 2/E
Cp = (B-A*C)/(E*(1-C))
Ep = (A-B)/(E*(1-C))
Fp = (x^2-1)*(x^2-Ap)*(Ep*x^2-Bp*x+Cp)
print("b'^2=f(mod a'):", all([c.numerator().reduce(I) == 0
```

```
 for c in
((Fp-bp^2)%ap).coefficients()]))
```

B.3: Proofs for Appendix A

```
#Type 1 Equation:
K.<u,A,B,C,E,a0,a1,b0,b1> = QQ[]
S.<x> = K[]
F = E*x*(x^2-A*x+1)*(x^2-B*x+C)
Ap = C
Bp = 2/E
Cp = (B-A*C)/(E*(1-C))
Ep = (A-B)/(E*(1-C))
Fp =  (x^2-1)*(x^2-Ap)*(Ep*x^2-Bp*x+Cp)

#Richelot correspondence:
rel1 = (u^2-B*u+1)*x^2 + 2*(C-1)*u*x - C*u^2+B*u-C
rel2 = (A-B)*u*x^3 - ((A-B)*u^2+2*(1-C)*u)* x^2 +
(2*(1-C)*u^2 - (A*C-B)*u)*x + (A*C-B)*u^2

#relations among Mumford coefficients
a = x^2+a1*x+a0
b = b1*x+b0
[q,r] = (F-b^2).quo_rem(a) #r must be zero
relations = [a(u)] + r.coefficients()
I = K.ideal(relations)
v = b0 + b1*u
s = -a1 - u
t = b0 + b1*s

print("Lemma 18:", -b1*(a1*b0-a0*b1)+b0^2 == a.resultant(b))

print("Proposition 21")
print("v1:", rel2(u=-a0, x = B/2) == 1/8*(4*C-B^2)*
(A-B)*a0*(B+2*a0))
print("v2:", (rel2(u=-a0, x=1/x)*x^3)(0) == (B-A)*a0)

print("Lemma 22:", a0*B^2 + (a0+1)*a1*B+(a0-1)^2+a1^2 ==
a.resultant(x^2-B*x+1))

print("General checks")
s = -a1-u
t = b1*s+b0
J = I + K.ideal(s^2-B*s+1)
s1 = B/2
t1 = (4*C-B^2)*(B-A)*s*(B-2*s)/(8*t)
e = (A-B)*s/t
print("Q1 on curve:", (t1^2-Fp(s1)).numerator().reduce(J)
```

```
 == 0)
print("square root E", (Ep-e^2).numerator().reduce(J) == 0)

print("Proposition 24:", 0)
J = I + K.ideal([a1+B,a0-1])
u1 = B/2
v1 = (4*C-B^2)*(B-A)*u*(B-2*u)/(8*v)
print("case b0=0:",(v1/t1).numerator().reduce(J
+ K.ideal(u*t-s*v))
== -1*(v1/t1).denominator().reduce(J+ K.ideal(u*t-s*v)) )
print("case b0!=0:",
(v1 - (4*C-B^2)*(B-A)/(4*b0)).numerator().reduce(J
+K.ideal(u*t+s*v)) == 0)

#Section A.3
aP1 = 2*(C-1)*u/(u^2-B*u+1)
aP0 = (-C*u^2+B*u-C)/(u^2-B*u+1)
bP1 = u*(1-C)*(u^2-A*u+1)/(u^2-B*u+1)^2/v * (2*u^3-B*u^2
+ (-B^2+4*C-2)*u+B)
bP0 = -u*(1-C)*(u^2-A*u+1)/(u^2-B*u+1)^2/v * (B*u^3
+(-B^2+2*C)*u^2 - B*u+2*C)
aP = x^2+aP1*x+aP0
bP = bP1*x +bP0

aQ1 = 2*(C-1)*s/(s^2-B*s+1)
aQ0 = (-C*s^2+B*s-C)/(s^2-B*s+1)
bQ1 = +s*(1-C)*(s^2-A*s+1)/(s^2-B*s+1)^2/t * (2*s^3-B*s^2
+ (-B^2+4*C-2)*s+B)
bQ0 = -s*(1-C)*(s^2-A*s+1)/(s^2-B*s+1)^2/t * (B*s^3
+(-B^2+2*C)*s^2 - B*s+2*C)
aQ = x^2+aQ1*x+aQ0
bQ = bQ1*x +bQ0

print("Lemma 25:", (aQ.resultant(aP).numerator().reduce(I)
==-(C-1)^2*(u-s)^2*(4*C-B^2)*(1-u*s)^2).reduce(I))

print("Proposition 27:")
J = I + K.ideal([a0-1])
rel1_s = (B+a1)*x^2 - 2*(C-1)*x - (B+a1*C)
print(all([c.reduce(J) == 0 for c in (rel1 +u*rel1_s).
coefficients()]))
d1 = (B*b0 - 2*(a1*b0-b1))*(a1+B) -4*b0*(C-1)
d0 = (B*(b1-a1*b0)+ 2*C*b0)*(a1+B) - 2*B*b0*(C-1)
nz = [B+a1,4*C-B^2, C-1, A+a1, -a1*b0*b1+b0^2+b1^2]
#nonzero terms
print("bP1-bQ1:", all([(bP1-bQ1).numerator().reduce(J) ==
-nz[0]^2*nz[2]*nz[3]*(2*u+a1)*d1,
(bP1-bQ1).denominator().reduce(J) == (v*t*nz[0]^4).
reduce(J)]))
```

```
print("bP0-bQ0:", all([(bP0-bQ0).numerator().reduce(J)
== nz[0]^2*nz[2]*nz[3]*(2*u+a1)*d0,
(bP1-bQ1).denominator().reduce(J) == (v*t*nz[0]^4).
reduce(J)]))
xhat = (bQ0-bP0)/(bP1-bQ1)
print("xhat:", (xhat - d0/d1).numerator().reduce(J) == 0)
yhat = bP1*xhat + bP0
print("yhat:", (yhat - nz[1]*nz[2]*nz[3]/d1).numerator().
reduce(J) == 0)
print("xhat is a root of aP:", rel1(xhatr).numerator().
reduce(J) == 0)

print("check that d1 nonzero (by contradiction):")
print("if d1=d0=0, ")
J1 = J + K.ideal([d0,d1])
print("then b1=b0=0 (contradiction):", all([prod(nz)^2*b0
 in J1, prod(nz)^2*b1 in J1]))
print("if d0 nonzero, then bP1=bQ1=0.", True)
#geometric argument
J2 = J + K.ideal([d1, (bP1+bQ1).numerator().reduce(J)])
print("then 0=1 (contradiction):", prod(nz)^2 in J2)
```

References

1. Bosma W., Cannon J., Playoust C.: The Magma algebra system. I. The user language. J. Symbol. Comput. **24**(3–4), 235–265 (1997) **Computational algebra and number theory (London, 1993)**.
2. Castryck W., Decru T.: An efficient key recovery attack on SIDH. In: Hazay C., Stam M. (eds.) EUROCRYPT 2023, Part V, LNCS, vol. 14008, pp. 423–447. Springer (April 2023).
3. Castryck W., Decru T., Smith B.: Hash functions from superspecial genus-2 curves using Richelot isogenies. J. Math. Cryptol. **14**(1), 268–292 (2020).
4. Castryck W., Lange T., Martindale C., Panny L., Renes J.: CSIDH: an efficient post-quantum commutative group action. In: Thomas P., Steven G. (eds.) ASIACRYPT 2018, pp. 395–427. Part III, LNCS, vol. 11274. Springer, Heidelberg (December 2018).
5. Charles D.X., Lauter K.E., Goren E.Z.: Cryptographic hash functions from expander graphs. J. Cryptol. **22**(1), 93–113 (2009).
6. Chen C., Zhang F.: Richelot isogenies, pairings on squared Kummer surfaces and applications. Cryptology ePrint Archive, Report 2021/1617 (2021). https://eprint.iacr.org/2021/1617.
7. Craig C.: Computing supersingular isogenies on Kummer surfaces. In: Thomas P., Steven G. (eds.) ASIACRYPT 2018, pp. 428–456. Part III, LNCS, vol. 11274. Springer, Heidelberg (2018).
8. Craig C., Kristin L.: Group law computations on Jacobians of hyperelliptic curves. In: Ali M., Serge V. (eds.) SAC 2011, vol. 7118, pp. 92–117. LNCS. Springer, Heidelberg (2012).
9. Craig C., Benjamin S.: The supersingular isogeny problem in genus 2 and beyond. In: Jintai D., Jean-Pierre T. (eds.) Post-Quantum Cryptography - 11th International Conference, pp. 151–168. PQCrypto 2020. Springer, Heidelberg (2020).
10. Couveignes J.-M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006). https://eprint.iacr.org/2006/291.
11. De Feo L., Jao D., Plût J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Math. Cryptol. **8**(3), 209–247 (2014).
12. Flynn E.V., Ti Y.B.: Genus two isogeny cryptography. In: Ding J., Steinwandt R. (eds.) Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, pp. 286–306. Springer, Heidelberg (2019).
13. Galbraith S.D., Harrison M., Mireles Morales D.J.: Efficient hyperelliptic arithmetic using balanced representation for divisors. In: van der Poorten A.J., Stein A. (eds.) International Algorithmic Number Theory Symposium, pp. 342–356, Springer, Berlin (2008).
14. Kunzweiler S.: Richelot isogenies. https://github.com/sabrinakunzweiler/richelot-isogenies (2023).

15. Kunzweiler S., Ti Y.B., Weitkämper C.: Secret keys in genus-2 SIDH. In: AlTawy R., Hülsing A. (eds.) SAC 2021, LNCS, vol. 13203 , pp. 483–507. Springer, Heidelberg, September/October (2022).
16. Lange T.: Formulae for arithmetic on genus 2 hyperelliptic curves. Appl. Algebra Eng. Commun. Comput. **15**(5), 295–328 (2005).
17. Liu Q.: Algebraic geometry and arithmetic curves. Oxford University Press on Demand, vol. 6 (2002).
18. Maino L., Martindale C., Panny L., Pope G., Wesolowski B.: A Direct Key Recovery Attack on SIDH. In: Hazay C., Stam M. (eds.) EUROCRYPT 2023, Part V, LNCS, vol. 14008, pp. 448–471. Springer (April 2023).
19. Oudompheng R., Pope G.: A note on reimplementing the Castryck–Decru attack and lessons learned for SageMath. Cryptology ePrint Archive (2022).
20. Richelot F.J.: Ueber die Integration eines merkwürdigen Systems Differentialgleichungen (1842).
21. Robert D.: Breaking SIDH in polynomial time. In: Hazay C., Stam M. (eds.) EUROCRYPT 2023, Part V, LNCS, vol. 14008, pp. 472–503. Springer (April 2023).
22. Rostovtsev A., Stolbunov A.: Public-Key Cryptosystem Based on Isogenies. Cryptology ePrint Archive, Report 2006/145 (2006). https://eprint.iacr.org/2006/145.
23. Smith B.A.: Explicit endomorphisms and correspondences. PhD thesis, University of Sydney (2005).
24. Stoll M.: Lecture notes in arithmetic of hyperelliptic curves, Summer semester (2014).
25. Takashima K.: Efficient algorithms for isogeny sequences and their cryptographic applications. In: Mathematical Modelling for Next-Generation Cryptography, pp. 97–114. Springer, Berlin (2018).
26. The Sage Developers. SageMath, the Sage Mathematics Software System (Version 9.7) (2022).
27. Zarhin Y.G.: Division by 2 on hyperelliptic curves and Jacobians. arXiv preprint arXiv:1606.05252 (2016).