A Note on $(2, 2)$-isogenies via Theta Coordinates

Jianming Lin[1], Saiyu Wang[1], and Chang-An Zhao[1,2]

[1] School of Mathematics, Sun Yat-sen University,
Guangzhou 510275, P.R.China
linjm28@mail2.sysu.edu.cn
wangsy58@mail2.sysu.edu.cn
zhaochan3@mail.sysu.edu.cn
[2] Guangdong Key Laboratory of Information Security,
Guangzhou 510006, P.R. China

Abstract. In this paper, we revisit the algorithm for computing chains of $(2, 2)$-isogenies between products of elliptic curves via theta coordinates proposed by Dartois et al. For each fundamental block of this algorithm, we provide a explicit inversion-free version. Besides, we exploit a novel technique of x-only ladder to speed up the computation of gluing isogeny. Finally, we present a mixed optimal strategy, which combines the inversion-elimination tool with the original methods together to execute a chain of $(2, 2)$-isogenies.

We make a cost analysis and present a concrete comparison between ours and the previously known methods for inversion elimination. Furthermore, we implement the mixed optimal strategy for benchmark. The results show that when computing $(2, 2)$-isogeny chains with lengths of 126, 208 and 632, compared to Dartois, Maino, Pope and Robert's original implementation, utilizing our techniques can reduce 30.8%, 20.3% and 9.9% multiplications over the base field $\mathbb{F}_p$, respectively. Even for the updated version which employs their inversion-free methods, our techniques still possess a slight advantage.

1 Introduction

Isogeny-based cryptography is a candidate of post-quantum cryptography. Compared to other post-quantum schemes, isogeny-based cryptosystems benefit from their short key sizes. Nevertheless, the executions of the isogeny-based protocols are inefficient since the isogeny computations are extremely expensive.

The key exchange protocol SIDH [15] was regarded as an efficient isogeny-based scheme. In 2017, the key escapsulation based on SIDH which is called SIKE was submitted to the NIST post-quantum cryptography standardization. However, SIDH was broken in polynomial time [7,16,26]. The aforementioned attacks on SIDH are revelant for researching on the higher dimension abelian varieties. Furthermore, many cryptographers utilize these attacks as powerful tools to design new cryptographic schemes.

Basso et al. use the SIDH-attack as a trapdoor, and propose the first public key encryption (PKE) scheme named FESTA using Kani's [2] theorem. Based

on FESTA, an optimization using quaternion algebra called QFESTA [20], a key encapsulation scheme IS-CUBE [17] and a variant combined with LIT-diagram named LIT-SiGamal [18] are constructed. On the signature side, a high-dimension version of SQIsign [10], called SQIsignHD [8], which uses isogenies of dimensions 4 or 8 was proposed in 2023. More recently, it was improved by exploiting new techniques to make the protocol itself work with isogenies of dimensions 2 or 4. The corresponding variants of SQIsign, SQIsign2D-West [1], SQIsign2D-East [21] and SQIPrime [11] have been successively proposed. To sum up, the computation for chains of $(2,2)$-isogenies between products of elliptic curves is the core of the protocols mentioned above. Consequently, enhancing the performance of computing $(2,2)$-isogenies is a crucial part for the develop of the high-dimension isogeny-based protocols.

The isogeny computations between elliptic curves have been extensively investigated, such as [12,13,14,3]. As for the case of genus 2, the explicit methods and formulas for the computation of $(2,2)$-isogenies between the Jacobians of hyperelliptic curves were first given by Richelot [23], and were further improved by Bost [4] and Cassels [6]. In a period of time, using Richelot isogenies was considered a satisfactory method to compute $(2,2)$-isogenies. Nevertheless, Dartois, Maino, Pope and Robert revisited the techniques in [24,25] and proposed a new algorithm to compute $(2,2)$-isogenies between dimension-2 abelian varieties in the theta models [9]. Compared to the Richelot correspondance, utilizing theta coordinates is much more efficient, which is the state-of-the-art for computing $(2,2)$-isogenies between products of elliptic curves.

Notation Let $\mathbf{M}$, $\mathbf{S}$ and $\mathbf{I}$ denote the cost of multiplication, squaring and inversion of an element over the base field $\mathbb{F}_p$ throughout the paper, respectively.

1.1 Contributions

In this paper, we propose several methods to optimize the $(2,2)$-isogeny computation. In the phase of precomputation and codomain recovery, the original algorithms in [9] require inversions over a finite field at each step of a $(2,2)$-isogeny chain. We present some novel techniques to avoid all the inversions in the computation of $(2,2)$-isogenies, including gluing and generic isogenies. Besides, we make a concrete cost analysis, and present the comparison (See Tables 1 and 2) for each building block in the isogeny computation between our methods, the approaches in [18] and the original/latest algorithms of [9] in terms of multiplications over the base field $\mathbb{F}_p$. The results show that our approach outperforms the methods in [9,18] when executing codomain recoveries and image point evaluations in the gluing isogeny computation. As for the generic isogeny, applying our techniques requires less multiplications to perform than utilizing the inversion-free methods in the latest verison of [9] and [18] in the phase of precomputations and codomain recoveries.

Table 1. Cost comparison of different methods in gluing isogeny.

	Ours	[9, **both original and latest**]	[18]
Doubling	$8\mathbf{S}+12\mathbf{M}+2\mathbf{C}$[a]	$12\mathbf{S}+10\mathbf{M}+2\mathbf{C}$	$8\mathbf{S}+12\mathbf{M}+2\mathbf{C}$[a]
Codomain	$8\mathbf{S}+4\mathbf{M}$[b]	$8\mathbf{S}+13\mathbf{M}+1\mathbf{I}$[b]	$8\mathbf{S}+6\mathbf{M}$[b]
Evaluation	$18\mathbf{S}+81\mathbf{M}$[c]	$18\mathbf{S}+82\mathbf{M}+1\mathbf{I}$[c]	$18\mathbf{S}+78\mathbf{M}+1\mathbf{I}$[c]

[a] An additional cost $6\mathbf{S}+22\mathbf{M}$ is required to recover the y-coordinates by using our method. While in [18], it needs to execute two square-roots.
[b] $1\mathbf{M}, 8\mathbf{M}, 3\mathbf{M}$ are included to compute the inversed dual theta-null points, respectively.
[c] $10\mathbf{S}+72\mathbf{M}$ is included to calculate the point additions and basis transformation.

Table 2. Cost comparison of different methods in generic isogeny.

	Ours	[9, **latest**]	[9, **original**]	[18]
Precomputation	$4\mathbf{S}+12\mathbf{M}$	$6\mathbf{S}+16\mathbf{M}$	$4\mathbf{S}+21\mathbf{M}+1\mathbf{I}$	$4\mathbf{S}+15\mathbf{M}+1\mathbf{I}$
Doubling	$8\mathbf{S}+8\mathbf{M}$	$8\mathbf{S}+8\mathbf{M}$	$8\mathbf{S}+6\mathbf{M}$	$8\mathbf{S}+7\mathbf{M}$
Codomain	$8\mathbf{S}+9\mathbf{M}$[a]	$8\mathbf{S}+11\mathbf{M}$[a,b]	$8\mathbf{S}+13\mathbf{M}+1\mathbf{I}$[a,c]	$9\mathbf{S}+16\mathbf{M}$[a]
Evaluation	$4\mathbf{S}+4\mathbf{M}$	$4\mathbf{S}+4\mathbf{M}$	$4\mathbf{S}+3\mathbf{M}$	$4\mathbf{S}+4\mathbf{M}$

[a] $3\mathbf{M}, 3\mathbf{M}, 4\mathbf{M}, 6\mathbf{M}$ are included to calculate the inversed dual theta-null points, respectively.
[b] $8\mathbf{S}+13\mathbf{M}$ is required if the precomputation is not performed.
[c] $8\mathbf{S}+23\mathbf{M}+1\mathbf{I}$ is required if the precomputation is not performed.

For the point doublings in the gluing isogeny step, we first employ the x-only ladder algorithm to obtain the x-coordinates of the target points, and recover the corresponding y-coordinates by Okeya-Skurai formula [22] at last. This adjustment can reduce the computational cost for the gluing isogeny compared to the previously known implementation [9,18].

According to Table 2, if we eliminate these inversions by using the projective space, it will bring several extra multiplications while executing a doubling or image evaluation. If the length of the chain of $(2,2)$-isogenies is short, it is preferred to utilize the technique of inversion elimination, the same goes for the opposite. Thus we provide a mixed optimal strategy which combines our techniques of inversion elimination with the original methods in [9] that require performing inverisons to make a trade-off. While in the updated version of [9], Dartois, Maino, Pope and Robert work with inversion-free projective points along the whole chain for simplicity. Moreover, based on the Rust code in [9], we implement the mixed optimal strategy in this paper. The experimental results (See Tables 3 and 4 for detail) illustrate that in terms of multiplications over a finite field $\mathbb{F}_p$, when computing $(2,2)$-isogeny chains with lengths of 126, 208 and 632, utilizing our techniques obtain savings of 30.8% (resp. 4.0%), 20.3% (resp. 2.6%) and 9.9% (resp. 1.9%) compared to the original (resp. the latest) version of implementation of [9], respectively. As for the running time, the improvements

become 28.0%, 21.2% and 10.4%, respectively. Besides, even compared to the latest implementation, our approaches have a slight advantage.

2 Mathematical preliminaries

In this section, we introduce the corresponding mathematical preliminaries of the $(2,2)$-isogeny computations, including theta coordinates, Hadamard transformation and duplication formula.

2.1 Theta coordinates

In this subsection, we introduce theta coordinates and their properties. See [9] for more details.

Let k be a field with characteristic different from 2 and let A be a abelian variety defined over k of dimension g. We assume that $\mathcal{L}$ is a symmetric line bundle on A and let T_P be the translation-by-P map over A. Defined by $K(\mathcal{L})$ the set of points such that the pullback of $\mathcal{L}$ under T_P is isomorphic to $\mathcal{L}$:

$$K(\mathcal{L}) = \{P \in A \mid T_P^*\mathcal{L} \cong \mathcal{L}\}.$$

Then the set $\mathcal{G}(\mathcal{L})$ defined as:

$$\mathcal{G}(\mathcal{L}) = \{(x, \phi_x) \mid x \in K(\mathcal{L}), \phi_x : \mathcal{L} \to T_x^*\mathcal{L}\}$$

has a group structure, we call it *Mumford theta group* . There is an exact sequence:

$$0 \to \bar{k}^* \to \mathcal{G}(\mathcal{L}) \to K(\mathcal{L}) \to 0.$$

Besides, there exist two maximal isotropic groups such that $K(\mathcal{L}) = K(\mathcal{L})_1 \oplus K(\mathcal{L})_2$. Let a sequence $\delta = (d_1, \cdots, g_g)$ such that:

$$K(\delta) = \bigoplus_{i=1}^{g} \mathbb{Z}/d_i\mathbb{Z}, \ H(\delta) = K(\delta) \oplus \mathrm{Hom}(K(\delta), \bar{k}^*).$$

A *theta structure of type* δ is an isomorphism $\Theta^{\mathcal{L}} : \mathcal{G}(\delta) \to \mathcal{G}(\mathcal{L})$. And the space $V(\delta) = \mathrm{Hom}(K(\delta), \bar{k})$ is a unique irreducible representation of $\mathcal{G}(\delta)$. Moreover, the space of the global section $\Gamma(A, \mathcal{L})$ is also a unique irreducible representation of $\mathcal{G}(\mathcal{L})$. Consequently, there is an isomorphism between $\Gamma(A, \mathcal{L})$ and $V(\delta)$. A basis $(\theta_i)_{i \in K(\mathcal{L})_1}$ derived from a canonical basis $(\delta_i)_{i \in K(\delta)}$ of $V(\delta)$ via the isomorphism is called *theta coordinates*. If $d_1 = \cdots = d_g = n$, these coordinates are called theta coordinates of *level* n.

For application purposes, we only consider theta coordinates of level 2 and assume that A is a product of elliptic curves ($g = 2$) in the rest of this paper. Such coordinates over A have the form of

$$(\theta_i^A)_{i \in \mathbb{Z}_2 \times \mathbb{Z}_2 = K(2,2)}.$$

We call the projective point $(\theta_i^A(0))_{i\in K(2,2)}$ *theta-null point*. Let $S_1, S_2 \in A[2]$ be points derived from the canonical basis of $K(2,2)$. Define $T_1, T_2 \in A[2]$ to be those from $\mathrm{Hom}(K(2,2), \bar{k}^*)$. Then the theta structure Θ^A is induced by a symplectic 4-torsion basis $S_1', S_2', T_1', T_2' \in A[4]$, where S_i' and T_i' lay above S_i and T_i, respectively [9]. Given points $P \in A$ and $T \in K(\mathcal{L})$, we can represent $P + T$ in theta coordinates as

$$(\theta_i^A(P+T))_i = (\chi(i)\theta_{i+s}^A(P))_i$$

if T corresponds to $(s, \chi) \in H(2,2)$.

2.2 Hadamard transform and duplication formula

In this subsection we introduce the definition of Hadamard transform, and then present the duplication formula, which is the core of $(2,2)$-isogeny computation.

Denote by $\mathcal{H}$ the action of the Hadamard transform on a theta coordinate $(\theta_i)_i$. The image coordinates after this transform are called *dual theta coordinates*. We denote such coordinates by $(\tilde{\theta}_i)_i$. A point P in theta coordinates is represented by $P = (\theta_{00} : \theta_{10} : \theta_{01} : \theta_{11})$[3] $= (x : y : z : t)$ in dimension 2. Then we have:

$$\begin{pmatrix} \tilde{\theta}_{00}(P) \\ \tilde{\theta}_{10}(P) \\ \tilde{\theta}_{01}(P) \\ \tilde{\theta}_{11}(P) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \theta_{00}(P) \\ \theta_{10}(P) \\ \theta_{01}(P) \\ \theta_{11}(P) \end{pmatrix}.$$

It is clear that $\mathcal{H}(\mathcal{H}((\theta_i(P))_i)) = (\theta_i(P))_i$ for any P.

Recall from Section 2.1, A is a product of two elliptic curves endowed with a type-$(2,2)$ theta structure induced by symplectic 4-torsion basis $\langle S_1', S_2', T_1', T_2' \rangle$. Consider the isogeny $f : A \to B$ with $\ker(f) = K(\mathcal{L})_2 = \langle T_1, T_2 \rangle$. Define $\star$ to be an operator such that $(x_i)_i \star (y_i)_i = (x_i y_i)_i$. Then, the duplication formula [19,24] shows that:

$$\left(\theta_i^A(P+Q)\right)_i \star \left(\theta_i^A(P-Q)\right)_i = \mathcal{H}\left(\left(\tilde{\theta}_i^B(f(P)))_i \star ((\tilde{\theta}_i^B(f(Q))\right)_i\right), \quad (1)$$

$$\mathcal{H}\left(\left(\theta_i^A(\tilde{f}(R)))_i \star ((\theta_i^A(\tilde{f}(S))\right)_i\right) = (\tilde{\theta}_i^B(R+S))_i \star (\tilde{\theta}_i^B(R-S))_i, \quad (2)$$

where $\tilde{f}$ is the dual isogeny of f. One can execute the point doubling or isogeny computation by using the duplication formula repeatly. See [9] for more detail.

3 Main Results

In this section, we present novel methods to optimize the implementation of $(2^n, 2^n)$-isogeny computation. For the point doubling and image point computation, we remove the inversions in the precomputation process before computing

[3] The subscript order follows the order in [9].

an isogeny chain by working on projective space, with some additional multiplications instead. The inversions in the phase of isogeny codomain computation are also eliminated. Besides, we use the technique of x-only ladder on Montgomery curves to speed up the scalar multiplication in gluing isogeny computation. Finally, we propose a mixed optimal strategy, which combines the previous method [9] and our new inversion-free techniques in the whole isogeny chain computation.

We first introduce some notations. Assume that we compute a $(2^n, 2^n)$-isogeny $f : E_1 \times E_2 \to E_1' \times E_2'$ with $\ker(f) = \left\langle \tilde{T}_1, \tilde{T}_2 \right\rangle$, where E_i, E_i' $(i = 1, 2)$ are Montgomery curves. Then the isogeny f can be splitted into n 2-isogenies:

$$E_1 \times E_2 \xrightarrow{f_1} A_1 \xrightarrow{f_2} \cdots \xrightarrow{f_{n-1}} A_{n-1} \xrightarrow{f_n} E_1' \times E_2', \tag{3}$$

where f_1, f_n and other f_i are called gluing isogeny, splitting isogeny and generic isogeny, respectively. Therefore, before performing each 2-isogeny, we need to execute several point doublings.

Denote by $(a : b : c : d)$ and $(\alpha : \beta : \gamma : \delta)$ the theta and dual theta coordinates on a product of elliptic curves, respectively. We also define the operators $\mathcal{S}, \mathcal{I}$ and $\mathcal{C}_{(a:b:c:d)}$ to be a square, an inversion and a scaling map, respectively. We assume that the abelian variety $E_1 \times E_2$ has already endowed with a compatible theta structure of type-$(2, 2)$, and omit the computational process of the transform between Montgomery coordinates and theta coordinates. See [9,27] for more details.

3.1 Inversion elimination

In this subsection, we state how to perform point doublings and isogeny computations without executing inversions.

According to the original algorithms in [9], for each generic isogeny in the isogeny chain, one can precompute six field elements a/b, a/c, a/d, α^2/β^2, α^2/γ^2, α^2/δ^2 at a cost of $4\mathbf{S} + 21\mathbf{M} + 1\mathbf{I}$. Utilizing the duplication formula, the point doubling can be done as:

$$(\theta_i^A([2]P))_i = \mathcal{C}_{(1:\frac{a}{b}:\frac{a}{c}:\frac{a}{d})} \circ \mathcal{H} \circ \mathcal{C}_{(1:\frac{\alpha^2}{\beta^2}:\frac{\alpha^2}{\gamma^2}:\frac{\alpha^2}{\delta^2})} \circ \mathcal{S} \circ \mathcal{H} \circ \mathcal{S}((\theta_i^A(P))_i).$$

Hence it requires $8\mathbf{S} + 6\mathbf{M}$ to execute a point doubling.

The isogeny computation includes the codomain and image point recoveries. Let f be a 2-isogeny from A to B with $\ker(f) = \langle T_1, T_2 \rangle$. Assume that the theta structure of A is induced by $\langle S_1, S_2, T_1, T_2 \rangle \subset A[4]$. We set $S_1 = (b : a : d : c)$, $S_2 = (c : d : a : b)$, $T_1 = (a : -b : c : -d)$ and $T_2 = (a : b : -c : -d)$. From the duplication formula, we have:

$$\mathcal{H} \circ \mathcal{S}((\theta_i^A(P))_i) = (\tilde{\theta}_i^B(f(P)))_i \star (\tilde{\theta}_i^B(0))_i. \tag{4}$$

Using the precomputed results α^2/β^2, α^2/γ^2, α^2/δ^2 and two 8-torsion points $T_i''(i = 1, 2)$ above T_i, it takes $8\mathbf{S} + 13\mathbf{M} + 1\mathbf{I}$ to recover the dual theta-null point

and its inverse on the codomain curve B. If no precomputation is performed, the cost would increase to $8\mathbf{S} + 23\mathbf{M} + 1\mathbf{I}$. But whatever the case, it takes $4\mathbf{S} + 3\mathbf{M}$ to obtain the coordinates of the image point.

Note that the original methods in [9] require two field inversions in each generic step of isogeny chain. If the length of chain is short, the implementation would become costly. Therefore, we aim to present the inversion-free algorithms for the doubling, codomain recovery and image evaluation.

Doubling For the point doubling in generic step, the coordinates $(1 : \frac{a}{b} : \frac{a}{c} : \frac{a}{d})$ and $(1 : \frac{\alpha^2}{\beta^2} : \frac{\alpha^2}{\gamma^2} : \frac{\alpha^2}{\delta^2})$ can be represented as

$$(1 : \frac{a}{b} : \frac{a}{c} : \frac{a}{d}) = (bcd : acd : abd : abc), \tag{5}$$

$$(1 : \frac{\alpha^2}{\beta^2} : \frac{\alpha^2}{\gamma^2} : \frac{\alpha^2}{\delta^2}) = (\beta^2\gamma^2\delta^2 : \alpha^2\gamma^2\delta^2 : \alpha^2\beta^2\delta^2 : \alpha^2\beta^2\gamma^2), \tag{6}$$

since we work on projective space. The doubling process is as follows:

$$(\theta_i^A([2]P))_i = \mathcal{C}_{(bcd:acd:abd:abc)} \circ \mathcal{H} \circ \mathcal{C}_{(\beta^2\gamma^2\delta^2:\alpha^2\gamma^2\delta^2:\alpha^2\beta^2\delta^2:\alpha^2\beta^2\gamma^2)} \circ \mathcal{S} \circ \mathcal{H} \circ \mathcal{S}((\theta_i^A(P))_i).$$

Algorithm 1 illustrates the computational details of point doubling.

Algorithm 1 Doubling

Input: The theta coordinates of P, the precomputed results $(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (bcd, acd, abd, abc)$ and $(\lambda_1', \lambda_2', \lambda_3', \lambda_4') = (\beta^2\gamma^2\delta^2, \alpha^2\gamma^2\delta^2, \alpha^2\beta^2\delta^2, \alpha^2\beta^2\gamma^2)$.
Output: The theta coordinates of $[2]P$.

1: $X_P, Y_P, Z_P, T_P \leftarrow \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, t_P)$
2: $X_{f(P)}^2 \leftarrow \lambda_1' \cdot X_P \cdot X_P$
3: $Y_{f(P)}^2 \leftarrow \lambda_2' \cdot Y_P \cdot Y_P$
4: $Z_{f(P)}^2 \leftarrow \lambda_3' \cdot Z_P \cdot Z_P$
5: $T_{f(P)}^2 \leftarrow \lambda_4' \cdot T_P \cdot T_P$
6: $X_P', Y_P', Z_P', T_P' \leftarrow \mathcal{H}(X_{f(P)}^2, Y_{f(P)}^2, Z_{f(P)}^2, T_{f(P)}^2)$
7: $X_{[2]P}, Y_{[2]P}, Z_{[2]P}, T_{[2]P} \leftarrow \lambda_1 \cdot X_P', \lambda_2 \cdot Y_P', \lambda_3 \cdot Z_P', \lambda_4 \cdot T_P'$
8: **return** $X_{[2]P}, Y_{[2]P}, Z_{[2]P}, T_{[2]P}$ ▷ Total cost: $8\mathbf{S} + 8\mathbf{M}$

Codomain recovery and image evaluation According to [9] we know that the theta-null point of the codomain can be determined by two 8-torsion points T_1'', T_2'' above T_1, T_2. By the properties of 2-torsion points and since we are on Kummer surface, the form of the dual theta coordinates of $f(T_1'')$ and $f(T_2'')$ can be determined as:

$$(\tilde{\theta}_i^B(f(T_1'')))_i = (x : x : y : y),$$
$$(\tilde{\theta}_i^B(f(T_2'')))_i = (z : t : z : t).$$

Therefore, employing duplication formula we have:

$$\mathcal{H} \circ \mathcal{S}((\theta_i^A(T_1''))_i) = (x\alpha : x\beta : y\gamma : y\delta) = (r_1 : r_2 : r_3 : r_4), \tag{7}$$

$$\mathcal{H} \circ \mathcal{S}((\theta_i^A(T_2''))_i) = (z\alpha : t\beta : z\gamma : t\delta) = (s_1 : s_2 : s_3 : s_4). \tag{8}$$

Similarly, the coordinates $(1 : \frac{\alpha}{\beta} : \frac{\alpha}{\gamma} : \frac{\alpha}{\delta})$ can be represented as $(\beta\gamma\delta : \alpha\gamma\delta : \alpha\beta\delta : \alpha\beta\gamma)$ since we are on the projective space. We can compute it through Equations (7) and (8):

$$\begin{aligned}(\frac{1}{\alpha} : \frac{1}{\beta} : \frac{1}{\gamma} : \frac{1}{\delta}) &= xzt \cdot (\beta\gamma\delta : \alpha\gamma\delta : \alpha\beta\delta : \alpha\beta\gamma) \\ &= (r_2s_3s_4 : r_1s_3s_4 : r_2s_1s_4 : r_1s_2s_3).\end{aligned} \tag{9}$$

As the squared dual theta-null point on B has been obtained after computing $\beta^2\gamma^2\delta^2,\ \alpha^2\gamma^2\delta^2,\ \alpha^2\beta^2\delta^2,\ \alpha^2\beta^2\gamma^2$, one can compute the dual theta-null point $(\alpha : \beta : \gamma : \delta)$ on B through $(\frac{1}{\alpha} : \frac{1}{\beta} : \frac{1}{\gamma} : \frac{1}{\delta}) \star (\alpha^2 : \beta^2 : \gamma^2 : \delta^2)$. However, we can use the following equation to compute $(\alpha : \beta : \gamma : \delta)$:

$$\begin{aligned}(\alpha : \beta : \gamma : \delta) &= xzt\alpha\beta \cdot (\alpha : \beta : \gamma : \delta) \\ &= (r_1s_1s_2 : r_2s_1s_2 : r_1s_2s_3 : r_2s_1s_4),\end{aligned} \tag{10}$$

which can further save one multiplication[4]. We describe it in Algorithm 2 using the batch multiplication.

Algorithm 2 Codomain

Input: The theta coordinates $(x_{T_1''} : y_{T_1''} : z_{T_1''} : t_{T_1''})$ and $(x_{T_2''} : y_{T_2''} : z_{T_2''} : t_{T_2''})$ of 8-torsion points T_1'' and T_2'', respectively.

Output: The dual theta-null point $(\alpha : \beta : \gamma : \delta)$, the inverse of dual theta-null point $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$ and the theta null-point $(a' : b' : c' : d')$ on codomain B.

1: $(r_1, r_2, r_3, r_4) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_1''}, y_{T_1''}, z_{T_1''}, t_{T_1''})$
2: $(s_1, s_2, s_3, s_4) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_2''}, y_{T_2''}, z_{T_2''}, t_{T_2''})$
3: $t_1 \leftarrow r_1 \cdot s_2$
4: $t_2 \leftarrow s_1 \cdot r_2$
5: $t_3 \leftarrow s_3 \cdot s_4$
6: $(\alpha, \beta, \gamma, \delta) \leftarrow (t_1 \cdot s_1, t_2 \cdot s_2, t_1 \cdot s_3, t_2 \cdot s_4)$
7: $(\alpha^{-1}, \beta^{-1}, \gamma^{-1}, \delta^{-1}) \leftarrow (r_2 \cdot t_3, r_1 \cdot t_3, \delta, \gamma)$
8: $(a', b', c', d') \leftarrow \mathcal{H}(\alpha, \beta, \gamma, \delta)$
9: **return** $(\alpha, \beta, \gamma, \delta), (\alpha^{-1}, \beta^{-1}, \gamma^{-1}, \delta^{-1})$ and (a', b', c', d') ▷ Total cost: $8\mathbf{S} + 9\mathbf{M}$

After obtaining the inverse of the dual theta-null point on B, we can compute the image point $f(P)$ using Equation (4). The corresponding process is described in Algorithm 3.

[4] Notice that the last two components of Equations (9) and (10) interchange with each other.

Algorithm 3 Evaluation (Image point)

Input: The theta coordinates $(x_P : y_P : z_P : t_P)$ of P and the inverse of dual theta-null point $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$ on B.
Output: The theta coordinates $(x_{f(P)} : y_{f(P)} : z_{f(P)} : t_{f(P)})$ of the image point $f(P)$.

1: $(X_P, Y_P, Z_P, T_P) \leftarrow \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, t_P)$
2: $(x'_{f(P)}, y'_{f(P)}, z'_{f(P)}, t'_{f(P)}) \leftarrow (X_P \cdot \alpha^{-1}, Y_P \cdot \beta^{-1}, Z_P \cdot \gamma^{-1}, T_P \cdot \delta^{-1})$
3: $(x_{f(P)}, y_{f(P)}, z_{f(P)}, t_{f(P)}) \leftarrow \mathcal{H}(x'_{f(P)}, y'_{f(P)}, z'_{f(P)}, t'_{f(P)})$
4: **return** $(x_{f(P)}, y_{f(P)}, z_{f(P)}, t_{f(P)})$ ▷ Total cost: $4\mathbf{S} + 4\mathbf{M}$

Gluing isogeny In this subsection, we consider the gluing isogeny $f : E_1 \times E_2 \to A$. According to [9], one of the coordinates of the dual theta-null point $(\alpha : \beta : \gamma : \delta)$ may be zero since we change the symplectic basis to make the abelian variety be compatible with an expected theta structure.

Assume that $\alpha = 0$. In this case, $(0 : \beta : \gamma : \delta)$ and $(0 : \frac{1}{\beta} : \frac{1}{\gamma} : \frac{1}{\delta})$ can be represented as $(0 : \frac{\beta}{\delta} : \frac{\gamma}{\delta} : 1)$ and $(0 : \frac{\delta}{\beta} : \frac{\delta}{\gamma} : 1)$, respectively. Similar to the generic step in the previous implementation [9], one can compute them by Equations (7) and (8).

We note that the dual theta-null point and its inverse can also be represented in the following form:

$$\begin{aligned}(0 : \frac{1}{\beta} : \frac{1}{\gamma} : \frac{1}{\delta}) &= yt \cdot (0 : \gamma\delta : \beta\delta : \beta\gamma) \\ &= (0 : y\gamma t\delta : t\beta y\delta : t\beta y\gamma) \\ &= (0 : r_3 s_4 : r_4 s_2 : r_3 s_2), \qquad (11) \\ (0 : \beta : \gamma : \delta) &= yt\delta \cdot (0 : \beta : \gamma : \delta) \\ &= (0 : t\beta y\delta : y\gamma t\delta : t\delta y\delta) \\ &= (0 : r_4 s_2 : r_3 s_4 : r_4 s_4). \qquad (12)\end{aligned}$$

Computing the coordinates in Equations (11) and (12) reduces the cost to $8\mathbf{S} + 4\mathbf{M}$. The implementation details are stated in Algorithm 4.

To recover the coordinates of the image point $(\tilde{\theta}_i^A(f(P)))_i$, we need the following two equations:

$$\mathcal{H} \circ \mathcal{S}((\theta_i^{E_1 \times E_2}(P))_i) = (0 : \beta\tilde{\theta}_{10}^A(f(P)) : \gamma\tilde{\theta}_{01}^A(f(P)) : \delta\tilde{\theta}_{11}^A(f(P))), \quad (13)$$

$$\mathcal{H} \circ \mathcal{S}((\theta_i^{E_1 \times E_2}(P + T'_1))_i) = (0 : \beta\tilde{\theta}_{00}^A(f(P)) : \gamma\tilde{\theta}_{11}^A(f(P)) : \delta\tilde{\theta}_{01}^A(f(P))). \quad (14)$$

Multiplying Equation (13) by $(0 : \beta^{-1} : \gamma^{-1} : \delta^{-1})$, we obtain all coordinates except $\tilde{\theta}_{00}^A(f(P))$. Actually, multiplying the second component of Equation (14) by β^{-1}, we obtain $\lambda\tilde{\theta}_{00}^A(f(P))$ for some factor λ since we are working with projective coordinates. If $\tilde{\theta}_{01}^A(f(P)) \neq 0$, we can determine λ^{-1} by utilizing the last component of Equation (14), δ^{-1} and $\tilde{\theta}_{01}^A(f(P))$. The implementation details are given in Algorithm 5.

Algorithm 4 Special Codomain, $\alpha = 0$

Input: The theta coordinates $(x_{T_1''} : y_{T_1''} : z_{T_1''} : t_{T_1''})$ and $(x_{T_2''} : y_{T_2''} : z_{T_2''} : t_{T_2''})$ of 8-torsion points T_1'' and T_2'', respectively.

Output: The dual theta-null point $(0 : \beta : \gamma : \delta)$, the "inverse" of dual theta-null point $(0 : \beta^{-1} : \gamma^{-1} : \delta^{-1})$ and the theta null-point $(a' : b' : c' : d')$ on codomain A.

$(0, r_2, r_3, r_4) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_1''}, y_{T_1''}, z_{T_1''}, t_{T_1''})$
$(0, s_2, s_3, s_4) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_2''}, y_{T_2''}, z_{T_2''}, t_{T_2''})$
$(t_1, t_2, t_3, t_4) \leftarrow (r_4 \cdot s_2, r_3 \cdot s_4, r_4 \cdot s_4, r_3 \cdot s_2)$
$(\beta, \gamma, \delta) \leftarrow (t_1, t_2, t_3)$
$(\beta^{-1}, \gamma^{-1}, \delta^{-1}) \leftarrow (t_2, t_1, t_4)$
$(a', b', c', d') \leftarrow \mathcal{H}(0, \beta, \gamma, \delta)$
return $(0, \beta, \gamma, \delta), (0, \beta^{-1}, \gamma^{-1}, \delta^{-1})$ and (a', b', c', d') ▷ Total cost: $8\mathbf{S} + 4\mathbf{M}$

Algorithm 5 Special Evaluation (Image point), $\alpha = 0$

Input: The theta coordinates $(x_P : y_P : z_P : t_P)$ and $(x_{P+T_1'} : y_{P+T_1'} : z_{P+T_1'} : t_{P+T_1'})$ of P and $P + T_1'$ respectively and the inverse of the dual theta-null point $(0 : \beta^{-1} : \gamma^{-1} : \delta^{-1})$ on A.

Output: The theta coordinates $(x_{f(P)} : y_{f(P)} : z_{f(P)} : t_{f(P)})$ of $f(P)$

$(0, Y_P, Z_P, T_P) \leftarrow \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, t_P)$
$(0, Y_{P+T_1'}, Z_{P+T_1'}, T_{P+T_1'}) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{P+T_1'}, y_{P+T_1'}, z_{P+T_1'}, t_{P+T_1'})$
$(Y_{f(P)}, Z_{f(P)}, T_{f(P)}) \leftarrow (\beta^{-1} \cdot Y_P, \gamma^{-1} \cdot Z_P, \delta^{-1} \cdot T_P)$
if $Z_{f(P)} \neq 0$ **then**
    $(r, s) \leftarrow (Z_{f(P)}, \delta^{-1} \cdot T_{P+T_1'})$
else
    $(r, s) \leftarrow (T_{f(P)}, \gamma^{-1} \cdot Z_{P+T_1'})$
end if
$(X_{f(P)}, Y_{f(P)}, Z_{f(P)}, T_{f(P)}) \leftarrow (r \cdot \beta^{-1} \cdot Y_{P+T_1'}, s \cdot Y_{f(P)}, s \cdot Z_{f(P)}, s \cdot T_{f(P)})$
$(x_{f(P)}, y_{f(P)}, z_{f(P)}, t_{f(P)}) \leftarrow \mathcal{H}(X_{f(P)}, Y_{f(P)}, Z_{f(P)}, T_{f(P)})$
return $(x_{f(P)}, y_{f(P)}, z_{f(P)}, t_{f(P)})$ ▷ Total cost: $8\mathbf{S} + 9\mathbf{M}$

3.2 x-only ladder in gluing isogeny

For the gluing isogeny, doubling is done on the elliptic curves via Montgomery coordinates, not on the Kummer surface via theta coordinates.

In the original implementation of [9], the y-coordinates of $P = (P_1, P_2) \in E_1 \times E_2$ are also be recovered while computing $[2^\bullet]P$ to get the 8-torsion point. It requires $6\mathbf{S} + 5\mathbf{M} + 1\mathbf{C}$ to compute a double of a point, where $\mathbf{C}$ represents multiplying by a constant. Thus it takes $12\mathbf{S} + 10\mathbf{M} + 2\mathbf{C}$ to obtain $([2]P_1, [2]P_2)$. Let $s = 2^\bullet$. Since we are on the Montgomery curve, we can use x-only double and add algorithm (Algorithm 6) combined with Montgomery ladder [5] to obtain couples of points $([s]P_1, [s]P_2)$ and $([s-1]P_1, [s-1]P_2)$, which is the same as the implementation of [18]. In each step of Montgomery ladder, the cost becomes $8\mathbf{S} + 12\mathbf{M} + 2\mathbf{C}$.

Algorithm 6 The double and add algorithm

Input: The x-coordinates (projective coordinate) of P, Q, $Q-P$ and $(a:1) = (A+2:4)$ where $(A:1)$ is the Montgomery coefficient

Output: $(X_{[2]P} : Z_{[2]P})$ and $(X_{P+Q} : Z_{P+Q})$

1: $t_0 \leftarrow X_P + Z_P$
2: $t_1 \leftarrow X_P - Z_P$
3: $X_{[2]P} \leftarrow t_0^2$
4: $t_2 \leftarrow X_Q - Z_Q$
5: $X_{P+Q} \leftarrow X_Q + Z_Q$
6: $t_0 \leftarrow t_0 \cdot t_2$
7: $Z_{[2]P} \leftarrow t_1^2$
8: $t_1 \leftarrow t_1 \cdot X_{P+Q}$
9: $t_2 \leftarrow X_{[2]P} - Z_{[2]P}$
10: $X_{[2]P} \leftarrow X_{[2]P} \cdot Z_{[2]P}$
11: $X_{P+Q} \leftarrow a \cdot t_2$
12: $Z_{P+Q} \leftarrow t_0 - t_1$
13: $Z_{[2]P} \leftarrow X_{P+Q} + Z_{[2]P}$
14: $X_{P+Q} \leftarrow t_0 + t_1$
15: $Z_{[2]P} \leftarrow Z_{[2]P} \cdot t_2$
16: $Z_{P+Q} \leftarrow Z_{P+Q}^2$
17: $X_{P+Q} \leftarrow X_{P+Q}^2$
18: $Z_{P+Q} \leftarrow X_{Q-P} \cdot Z_{P+Q}$
19: $X_{P+Q} \leftarrow Z_{Q-P} \cdot X_{P+Q}$
20: **return** $(X_{[2]P} : Z_{[2]P}), (X_{P+Q} : Z_{P+Q})$

▷ Total cost: $4\mathbf{S} + 6\mathbf{M} + 1\mathbf{C}$

With the help of the x-coordinates of $[s]P_i$ and $[s+1]P_i$, one can recover $y_{[s]P_i}$ by Okeya-Skurai formula [22]:

$$y_{[s]P_i} = \frac{(x_{[s]P_i}x_{P_i} + 1) \cdot (x_{[s]P_i} + x_{P_i} + 2A) - 2A - (x_{[s]P_i} - x_{P_i})^2 \cdot x_{[s+1]P_i}}{2y_{P_i}}.$$

In a similar manner, we can recover $y_{[s]P_i}$ through $x_{[s]P_i}$ and $x_{[s-1]P_i}$:

$$y_{[s]P_i} = \frac{(x_{[s]P_i}x_{P_i} + 1) \cdot (x_{[s]P_i} + x_{P_i} + 2A) - 2A - (x_{[s]P_i} - x_{P_i})^2 \cdot x_{[s-1]P_i}}{-2y_{P_i}},$$

Since we work on the projective coordinates, it takes an additional cost $3\mathbf{S}+11\mathbf{M}$ to recover $[s]P_i = (X_{[s]P_i} : Y_{[s]P_i} : Z_{[s]P_i})$. While in [18], Moriya et al. perform square-roots to recover the y-coordinates of $[s]P_i$. Consequently, utilizing our techniques is more efficient than the using the tools in [9,18] for doublings in the gluing isogeny step. We omit the detail of cost calculation for simplicity.

3.3 Mixed optimal strategy

Based on the above analysis, we need to compute eight field elements in the phase of precomputation at a cost of $4\mathbf{S} + 12\mathbf{M}$. Consequently, compared to the original implementation [9], in each generic step of the isogeny chain we can save $13\mathbf{M} + 2\mathbf{I}$ in precomputation and codomain recovery, with a trade-off of one and two more multiplications in each image computation and doubling step, respectively (See Table 2 for detail). This adjustment allows us to make use of a mixed optimal strategy to determine which method to use when executing doubling and image evaluation.

Consider dividing the remaining cost into the following four parts:

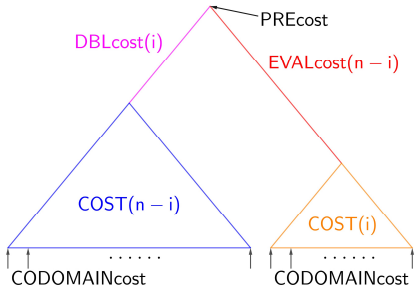
$$\texttt{PREcost},\ \texttt{DBLcost},\ \texttt{EVALcost} \text{ and } \texttt{CODOMAINcost}$$

which correspond to the cost of precomputation, doubling, image point computation and codomain computation, respectively. The calculation of the minimum cost can be written as the following iterative formula:

$$\texttt{COST(n)} = \begin{cases} \texttt{PREcost} + \min\limits_{1 \leqslant i < n} \Big\{ \texttt{DBLcost(i)} + \texttt{COST(n - i)} \\ \qquad\qquad + \texttt{EVALcost(n - i)} + \texttt{COST(i)} \Big\} & n > 1, \\ \texttt{CODOMAINcost} & n = 1. \end{cases}$$

We also present Figure 1 for illustration.

Fig. 1. Cost of a strategy tree with height n



To construct our mixed optimal strategy, we need a more concrete analysis for the above formula. As shown in Figure 1, the strategy tree is splitted into four parts: doubling, evaluation and two sub optimal strategies. Each step in the isogeny chain should be determined whether or not to use the inversion-free method. Consequently, the costs of going left and right should not be calculated independently of each other. Specifically, the choice for the computation of `EVALcost(n-i)` depends on how those codomains are computed at the leaf nodes of the left subtree. As a result, not only do we compute the minimum cost, but we also need to keep track of the method used at each step. To accomplish this, we introduce a new parameter `flag` to denote the method (inversion or inversion-free) used at each step. If the inversion-free method is employed, we set `flag` to be `True`.

Furthermore, it would be seen that for some cases the precomputation is unnecessary in Section 4. Hence, we introduce another new parameter `precomp` to indicate whether the `PREcost` has been computed, where the computation of `PREcost` needs to be taken into account when `precomp` is `False`. Finally, we utilize the `leftmost` parameter to indicate whether it is a gluing isogeny, which allows the cost of gluing step to be taken into account.

Ultimately, with the above analysis, our cost function can be written as `COST(n, flag, leftmost, precomp)`, which returns two variables `mincost` and `minflag`. The former is the minimum cost, and the latter is the method used in each isogeny step to get the minimum cost. Note that the sub-cost functions `PREcost`, `DBLcost`, `EVALcost` and `CODOMAINcost` require the specific parameters related to their respective calculations. Among them, it is important to point out that `EVALcost` can only be executed after the strategy of the left subtree is finished, because it depends on the return value `minflag` for calculation.

Compare the following two costs

$$\texttt{COST}(\texttt{n}, \texttt{False}, \texttt{True}, \texttt{True}) \text{ and } \texttt{COST}(\texttt{n}, \texttt{True}, \texttt{True}, \texttt{True}),$$

we can obtain the minimum cost for the whole strategy. To get the optimal strategy corresponding to the minimum cost, we add a global variable `checkpoint` just like the previous implementation in [9]. More details for the implementation of our mixed optimal strategy are described in Procedure 7.

Procedure 7 `COST(n, flag, leftmost, precomp)`

Input: `n, flag, leftmost, precomp`
Output: `mincost, minflag`

1: **if** `n` $\leqslant$ `1` **then**
2: **return** `CODOMAINcost(flag, leftmost, precomp), flag`
3: **end if**
4: `mincost` $\leftarrow \infty$
5: **for** `i` $=$ `1` to `n` $-$ `1` **do**
6: `thiscost` $\leftarrow$ `0`
7: **if** `precomp` is `False` **then**
8: `thiscost` $\leftarrow$ `PREcost(flag)`
9: **end if**
10: `thiscost` $\leftarrow$ `thiscost` $+$ `2` $*$ `DBLcost(i, flag, leftmost)`
11: `Lcost, Lflag` $\leftarrow$ `COST(n` $-$ `i, flag, leftmost, True)`
12: `thiscost` $\leftarrow$ `thiscost` $+$ `Lcost` $+$ `2` $*$ `EVALcost(n` $-$ `i, Lflag, leftmost)`
13: `RcostOLD, RflagOLD` $\leftarrow$ `COST(i, False, False, False)`
14: `RcostNEW, RflagNEW` $\leftarrow$ `COST(i, True, False, False)`
15: **if** `RcostOLD` $<$ `RcostNEW` **then**
16: `Rcost, Rflag` $\leftarrow$ `RcostOLD, RflagOLD`
17: **else**
18: `Rcost, Rflag` $\leftarrow$ `RcostNEW, RflagNEW`
19: **end if**
20: `thiscost` $\leftarrow$ `thiscost` $+$ `Rcost`
21: `thisflag` $\leftarrow$ `(Lflag, Rflag)`
22: **if** `thiscost` $<$ `mincost` **then**
23: `mincost` $\leftarrow$ `thiscost`
24: `minflag` $\leftarrow$ `thisflag`
25: **end if**
26: **end for**
27: **return** `mincost, minflag`

4 Cost analysis and implementation results

In this section, we make a concrete computational cost comparison for each block of the isogeny computation (gluing isogeny and generic isogeny) between using our techniques and the previous methods [9,18]. Besides, we implement the mixed strategy obtained by using Procedure 7, and compare the efficiency with the previous strategy in [9].

We first provide a cost comparison for the different methods in the gluing isogeny step, including our inversion-free method, the original and latest methods in [9] and the technique used in LIT-SiGamal [18, Appendix A.4]. To obtain the inverse of dual theta-null coordinates $(0 : \frac{1}{\beta} : \frac{1}{\gamma} : \frac{1}{\delta})$, the original/latest implementations of [9] utilizes the batch inversion, while the method in [18, Appendix A.4, Algorithm 2] makes use of the projective inversion instead and we exploit our inversion-free method described in Section 3. In terms of computational cost, the batch inversion in [9] requires $6\mathbf{M} + 1\mathbf{I}$, while the approach in [18, Algorithm 2] requires $5\mathbf{M}$ ($3\mathbf{M}$ in their implementation) and we only need $3\mathbf{M}$. Moreover, for implementation we can reduce the cost to $1\mathbf{M}$, as shown in Algorithm 4. In the phase of computing the image point, the implementation in [9] requires $8\mathbf{S} + 10\mathbf{M} + 1\mathbf{I}$, while in our implementation the corresponding cost is only $8\mathbf{S} + 9\mathbf{M}$. The details of the above comparison are illustrated in Table 1.

For the generic isogeny computation, in the latest version of [9], Dartois et al. apply an inversion-free method which is different from ours. In the precomputation phase, they multiply $(1 : \frac{a}{b} : \frac{a}{c} : \frac{a}{d})$ (resp. $(1 : \frac{\alpha^2}{\beta^2} : \frac{\alpha^2}{\gamma^2} : \frac{\alpha^2}{\delta^2})$) by $abcd$ (resp. $\alpha^2\beta^2\gamma^2\delta^2$), where the factor a (resp. α^2) is unsecessary according to our Equations (5) and (6). This results their cost being $6\mathbf{S} + 16\mathbf{M}$, compare to which we only need $4\mathbf{S} + 12\mathbf{M}$. To compute the dual theta-null point, they apply the same equation as Equation (10) but without reusing the intermediate results. As for the inversed dual theta-dull point, they multiply $(\tilde{\theta}_i^B(0))_i$ by the inverse of $(\tilde{\theta}_i^B(0))_i^2$ if the precomputation has been finished, otherwise they directly compute $(\beta\gamma\delta : \alpha\gamma\delta : \alpha\beta\delta : \alpha\beta\gamma)$ through $(\tilde{\theta}_i^B(0))_i = (\alpha : \beta : \gamma : \delta)$ by batch multiplication. In this way, it requires $4\mathbf{M}$ or $6\mathbf{M}$ to obtain the inversion of the dual theta-dull point by employing their techniques, depending on whether the precomputation is performed. In contrast, we observe that there are duplicates in Equations (9) and (10), which makes the computation for the inversed dual theta-null point be executed at a cost of only $3\mathbf{M}$. Table 2 presents the cost comparison in generic isogeny step.

In the implementation of strategy in [9], the cost of precomputation and codomain computation are both ignored. There is no impact if the aforementioned costs remain the same in each isogeny step, but we point out that this assumption does not always hold. At the rightmost leaf node of each subtree, the precomputation need not to be performed. Therefore, the cost of precomputation will reduced to zero, and the cost of codomain computation will increase to $8\mathbf{S} + 23\mathbf{M} + 1\mathbf{I}$ or $8\mathbf{S} + 13\mathbf{M}$ in [9], as written in the footnote in Table 2. We take both the cost of precomputation and codomain computation into account to get a strategy, which avoids the redundant precomputations in some cases.

This is one of the reasons why our new strategy algorithm is more accurate and considerable.

At the same time, we also combine the previous method with our inversion-free method, which has been explained in Section 3.3. The comparison is shown in Table 3. For chains with lengths of $126, 308, 632$, we obtain 30.8% (resp. 4.0%), 20.3% (resp. 2.6%), 9.9% (resp. 1.9%) savings of multiplications over the base field $\mathbb{F}_p$ compared to the original (resp. the latest) implementation of [9].

Table 3. Total cost comparison, measured by numbers of **M**

$\log p$	n	**Ours**	[9, **latest**]	[9, **original**]
254	126	24224	25235	35025
381	208	45158	46356	56678
1293	632	157747	160744	175031

Furthermore, we implement our mixed optimal strategy based on the Rust code in [9] for benchmark. Table 4 presents the comparison of running times between our mixed method and both the original and the latest version of [9]. For the Rust implementation, the compiler version is `1.80.0-nightly` and the flag is the same as [9], which is `-C target-cpu=native`. In terms of the running time, our method is 28.0%, 21.2%, 10.4% faster than the original implementation of [9], and have a slight advantage over the latest implementation.

Table 4. Comparison of running times for computing the codomain with different methods in Rust implementation. Times were recorded on a Intel Core i7-10510U CPU with a base clock speed of 1.8 GHz with turbo boost disabled.

$\log p$	n	**Ours**	[9, **latest**]	[9, **original**]
254	126	2.555 ms	2.568 ms	3.551 ms
381	208	10.038 ms	10.299 ms	12.737 ms
1293	632	534.82 ms	540.52 ms	597.20 ms

5 Conclusion

In this paper, we introduced the algorithms that remove the inversions in the phases of the precomputations and the codomain computations at each step of a $(2,2)$-isogeny chain between product of elliptic curves. Especially for the gluing isogeny, we employed the technique of x-only ladder to enhance the performance. Combined with the original methods, we provided a mixed optimal strategy and tested it on the code in [9]. The cost analysis illustrated that our techniques outperformed the previous methods [9,18], which will benefit the high-dimension isogeny-based protocols.

References

1. Basso, A., Feo, L.D., Dartois, P., Leroux, A., Maino, L., Pope, G., Robert, D., Wesolowski, B.: SQIsign2D-West: The Fast, the Small, and the Safer. Cryptology ePrint Archive, Paper 2024/760 (2024), https://eprint.iacr.org/2024/760
2. Basso, A., Maino, L., Pope, G.: "FESTA: Fast Encryption from Supersingular Torsion Attacks". In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. pp. 98–126. Springer Nature Singapore, Singapore (2023)
3. Bernstein, D.J., de Feo, L., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. In: Galbraith, S. (ed.) ANTS-XIV - 14th Algorithmic Number Theory Symposium. Proceedings of the Fourteenth Algorithmic Number Theory Symposium (ANTS-XIV), vol. 4, pp. 39–55. Mathematical Sciences Publishers, Auckland, New Zealand (Jun 2020)
4. Bost, J.B.: Moyenne arithmético-géométrique et périodes des courbes de genre 1 et 2. Gaz. Math. **38**, 36–64 (1988)
5. Campagna, M., Costello, C., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Renes, J., Urbanik, D., et al.: Supersingular isogeny key encapsulation (2019)
6. Cassels, J.W.S., Flynn, E.V.: Prolegomena to a middlebrow arithmetic of curves of genus 2, vol. 230. Cambridge University Press (1996)
7. Castryck, W., Decru, T.: "An Efficient Key Recovery Attack on SIDH". In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 423–447. Springer Nature Switzerland, Cham (2023)
8. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: "SQIsignHD: New Dimensions in Cryptography". In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 3–32. Springer Nature Switzerland, Cham (2024)
9. Dartois, P., Maino, L., Pope, G., Robert, D.: An Algorithmic Approach to $(2, 2)$-isogenies in the Theta Model and Applications to Isogeny-based Cryptography. Cryptology ePrint Archive, Paper 2023/1747 (2023), https://eprint.iacr.org/2023/1747
10. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: "SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies". In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. pp. 64–93. Springer International Publishing, Cham (2020)
11. Duparc, M., Fouotsa, T.B.: SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies. Cryptology ePrint Archive, Paper 2024/773 (2024), https://eprint.iacr.org/2024/773
12. Hu, Z., Liu, Z., Wang, L., Zhou, Z.: Simplified isogeny formulas on twisted Jacobi quartic curves. Finite Fields and Their Applications **78**, 101981 (2022)
13. Hu, Z., Wang, L., Zhou, Z.: "Isogeny Computation on Twisted Jacobi Intersections". In: Deng, R., Bao, F., Wang, G., Shen, J., Ryan, M., Meng, W., Wang, D. (eds.) Information Security Practice and Experience. pp. 46–56. Springer International Publishing, Cham (2021)
14. Huang, Y., Jin, Y., Hu, Z., Zhang, F.: Optimizing the evaluation of -isogenous curve for isogeny-based cryptography. Information Processing Letters **178**, 106301 (2022)
15. Jao, D., De Feo, L.: "Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies". In: Yang, B.Y. (ed.) Post-Quantum Cryptography. pp. 19–34. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

16. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A Direct Key Recovery Attack on SIDH. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 448–471. Springer Nature Switzerland, Cham (2023)
17. Moriya, T.: IS-CUBE: An isogeny-based compact KEM using a boxed SIDH diagram. Cryptology ePrint Archive, Paper 2023/1506 (2023), https://eprint.iacr.org/2023/1506
18. Moriya, T.: LIT-SiGamal: An efficient isogeny-based PKE based on a LIT diagram. Cryptology ePrint Archive, Paper 2024/521 (2024), https://eprint.iacr.org/2024/521
19. Mumford, D.B.: On the equations defining abelian varieties. I. Inventiones mathematicae **1**, "287–354" (1966)
20. Nakagawa, K., Onuki, H.: QFESTA: Efficient Algorithms and Parameters for FESTA using Quaternion Algebras. Cryptology ePrint Archive, Paper 2023/1468 (2023), https://eprint.iacr.org/2023/1468
21. Nakagawa, K., Onuki, H.: SQIsign2D-East: A New Signature Scheme Using 2-dimensional Isogenies. Cryptology ePrint Archive, Paper 2024/771 (2024), https://eprint.iacr.org/2024/771
22. Okeya, K., Sakurai, K.: Efficient elliptic curve cryptosystems from a scalar multiplication algorithm with recovery of the y-coordinate on a Montgomery-form elliptic curve. In: Cryptographic Hardware and Embedded Systems—CHES 2001: Third International Workshop Paris, France, May 14–16, 2001 Proceedings 3. pp. 126–141. Springer (2001)
23. Richelot, F.J.: De transformatione integralium Abelianorum primi ordinis commentatio. Caput secundum. De computatione integralium Abelianorum primi ordinis. Journal für die reine und angewandte Mathematik **1837**(16), 285–341 (1837), https://doi.org/10.1515/crll.1837.16.285
24. Robert, D.: Fonctions thêta et applications à la cryptographie. Ph.D. thesis, Université Henri Poincaré-Nancy I (2010)
25. Robert, D.: Efficient algorithms for abelian varieties and their moduli spaces. Ph.D. thesis, Université de Bordeaux (UB) (2021)
26. Robert, D.: Breaking SIDH in Polynomial Time. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 472–503. Springer Nature Switzerland, Cham (2023)
27. Robert, D.: Some notes on algorithms for abelian varieties. Cryptology ePrint Archive, Paper 2024/406 (2024), https://eprint.iacr.org/2024/406