Efficient theta-based algorithms for computing (ℓ, ℓ)-isogenies on Kummer surfaces for arbitrary odd ℓ

Ryo Yoshizumi[1], Hiroshi Onuki[2], Ryo Ohashi[2], Momonari Kudo[3], and Koji Nuida[1,4]

[1] Kyushu University, Japan
[2] The University of Tokyo, Japan
[3] Fukuoka Institute of Technology, Japan
[4] National Institute of Advanced Industrial Science and Technology, Japan

Abstract. Isogeny-based cryptography is one of the candidates for post-quantum cryptography. Recently, several isogeny-based cryptosystems using isogenies between Kummer surfaces were proposed. Most of those cryptosystems use $(2,2)$-isogenies. However, to enhance the possibility of cryptosystems, higher degree isogenies, i.e., (ℓ, ℓ)-isogenies for an odd ℓ, are also crucial. For an odd ℓ, Lubicz–Robert proposed a formula to compute $(\ell)^g$-isogenies in general dimensions g. In this paper, we propose explicit and efficient algorithms to compute (ℓ, ℓ)-isogenies between Kummer surfaces, based on the Lubicz–Robert formula. In particular, we propose two algorithms for computing the codomain of the isogeny and two algorithms for evaluating the image of a point under the isogeny. Then, we count the number of arithmetic operations required for each proposed algorithm and determine the most efficient algorithm in terms of the number of operations for each algorithm for each ℓ. As an application, we implemented the SIDH attack on B-SIDH in `SageMath` using the most efficient algorithm. In a setting that originally claimed 128-bit security, our implementation was able to recover the secret key in approximately 11 h.

Keywords: Post-quantum cryptography · Isogeny-based cryptography · B-SIDH · Kummer surface· Theta function

1 Introduction

Isogeny-based cryptography is one of the candidates for post-quantum cryptography. Its advantage is that it has relatively small keys, ciphertexts, and signatures.

Acknowledgments.This research was supported by WISE program (MEXT) at Kyushu University and JSPS KAKENHI Grant Numbers JP23K12949 and JP22K11906, and conducted in part under a contract of "Research and Development for Expansion of Radio Wave Resources (JPJ000254)" by the Ministry of Internal Affairs and Communications, Japan.

By contrast, its processing speed is slower than that of several other candidates for post-quantum cryptography. This arises from the computation of isogenies. Therefore, improving the computation of isogenies is important. Several studies have addressed this topic ([42,21,15,3,18,11,29]).

Vélu's formulas [42] serve as a method for calculating ℓ-isogenies between elliptic curves, where an *ℓ-isogeny* is defined as an isogeny whose kernel is a cyclic group of order ℓ. The computational complexity of Vélu's formulas is $O(\ell)$ operations in the base field. Although the classical Vélu's formulas are formulas of the Weierstrass form and use x- and y-coordinates, it is possible to obtain formulas using only x-coordinates, that is, formulas on Kummer lines. In particular, formulas on Montgomery curves are well known. The first formulas on Montgomery curves were given by Jao and De Feo [21]. They showed a method to derive formulas for isogenies of arbitrary degree; however, explicit formulas for isogenies of degree greater than four were not given. Costello and Hisil [15] gave explicit formulas for isogenies of arbitrary odd degree on Montgomery curves. Their formulas are more efficient than those derived from the method of Jao and De Feo. Based on their formulas, isogeny-based schemes such as CSIDH [8] and B-SIDH [14] were proposed. Later, the formula for the codomain curve of isogeny was improved by [31]. The computational complexity of an ℓ-isogeny was reduced to $\tilde{O}(\sqrt{\ell})$ by [3].

A generalization of ℓ-isogenies to 2-dimensional isogenies is called *(ℓ, ℓ)-isogenies*. In recent years, cryptosystems that combine isogenies between elliptic curves and isogenies between higher-dimensional abelian varieties have been proposed (for example, [2,9,17,36]). Several of these schemes use $(2, 2)$-isogenies for the higher-dimensional isogenies. The reason is that the computation of $(2, 2)$-isogenies is relatively efficient compared to that of higher-dimensional isogenies of other degrees. In particular, there is an efficient formula for $(2, 2)$-isogenies on Kummer surfaces by [18]. To enhance the variety of isogeny-based schemes, it is important to have efficient formulas for isogenies of higher degrees. Indeed, formulas for $(3, 3)$-isogenies on Kummer surfaces were given by [11]. For a general odd number ℓ, formulas for (ℓ, ℓ)-isogenies were given by Lubicz and Robert [29].

Lubicz–Robert formula. Let k be an algebraically closed field of characteristic zero or odd prime number p. Let A be an abelian variety of dimension g over k, $\mathscr{L} = \mathscr{L}_0^n$ be a line bundle on A where $\mathscr{L}_0$ is a principal polarization and n is even, and $\Theta_{\mathscr{L}}$ be a symmetric theta structure for $(A, \mathscr{L})$. For any odd number ℓ coprime to p and a maximal isotropic subgroup $K \subset A[\ell]$ of rank g with respect to the Weil pairing, the isogeny $f : A \to B = A/K$, called $(\ell)^g$-isogeny, induces a line bundle $\mathscr{M}$ on B and a symmetric theta structure $\Theta_{\mathscr{M}}$ for $(B, \mathscr{M})$ of level n. The theta structure of level n gives a morphism $\varphi_n : A \to \mathbb{P}^{n^g-1}$, and for $x \in A$, the projective coordinate $\varphi_n(x) \in \mathbb{P}^{n^g-1}$ is called a *theta coordinate* of x. Especially, $\varphi_n(0)$ is called *theta-null point*. We take a representation of ℓ as a sum of squares of integers: $\ell = \sum_{u=1}^{r} a_u^2$. Then,the Lubicz–Robert formula [29] gives a theta coordinate of $f(x) \in B$ for $x \in A$ up to multiplication by a constant from some theta coordinates on A:

$$\theta_i^B(f(x)) = \sum_{e \in K} \prod_{u=1}^{r} \mathtt{Mult}(a_u, \widetilde{x+e})_{a_u i} \ . \qquad (1)$$

By Lagrange's four-square theorem, we can take a representation $\ell = \sum_{u=1}^{r} a_u^2$ such as $r \leq 4$. Then, the complexity of the above formula is $O_r(\ell^g n^g)$ operations on k where O_r denotes the big O notation under the assumption that $r = O(1)$. For a precise formula, see [29, Corollary 4.6] or Section 2.4.

When $n = 2$, if A is indecomposable, it is known that the above morphism φ_2 gives the embedding of the Kummer variety K_A to $\mathbb{P}^{2^g-1}$, where the Kummer variety is the quotient $A/\langle \pm 1 \rangle$. Thus, the above formula for $n = 2$ gives an efficient method to calculate a morphism between the Kummer varieties.

In [29], an $(\ell)^g$-isogeny calculation algorithm based on (1) (for general dimension and general level) is given as [29, Algorithm 4]. However, we consider that there are the following points where improvements can be made:

1. Using (1), we can compute both the theta-null point of a codomain and the theta coordinate of the image under f for a given point. However, if we separate the codomain and evaluation, are there improvements for each?
2. Which of the possible representations $\ell = \sum_{u=1}^{r} a_u^2$ makes the algorithm the most efficient?
3. To use (1), we need to construct excellent lifts from given affine lifts, called normalization. Then, how can normalization be calculated efficiently?
4. In a cryptographic situation, calculating multiplicative inversion is expensive. Can we construct inversion-free algorithms?
5. What are explicit algorithms and their numbers of arithmetic operations on the base field k?

Our contribution. We propose some explicit algorithms of (ℓ, ℓ)-isogeny calculations between Kummer surfaces based on the Lubicz–Robert formula (1). Then, in our algorithms, we make the following contributions for the above listed points:

1. We consider the codomain and evaluation separately and propose algorithms for each. In particular, for the codomain, we reduce the computation steps to half, as discussed in Section 3.4.
2. We separate two cases: representations $\ell = \sum_{u=1}^{r} a_u^2$ and $\ell = 1^2 + \cdots + 1^2$. Of course, the latter representation can be applied as the special case of the former one, however, the latter one can utilize the specificity of its representation to construct efficient algorithms. Then, for both the codomain and evaluation, we provide two algorithms using these two representations. In addition, we investigate the former case in detail in Section 3.5.
3. We provide a method to calculate normalization, which improves our isogeny calculation algorithms. In Section 3.3, we provide some necessary equations and in Section 3.4, we propose the concrete method.

4. In our proposed algorithms, we avoid calculating multiplicative inversion on the base field k.
5. From the above items 1 and 2, we propose four algorithms. For them, we give explicit algorithms, complexities, and the numbers of arithmetic operations for small ℓ. For details, see Sections 3.4 and 4.

	$\ell = \sum_{u=1}^{r} a_u^2$,	$\ell = 1^2 + \cdots + 1^2$
Codomain	`CodSq`	`CodOne`
Evaluation	`EvalSq`	`EvalOne`

Table 1. Our proposed algorithms in Section 3.4

Section 3.1 provides an overview of these algorithms, and Section 3.4 presents the concrete algorithms. Here, `CodSq` is an $O_r(\ell^2)$ operations algorithms and `CodOne` is an $O(\ell^2 \log(\ell))$ operations algorithm. Similarly, `EvalSq` is an $O_r(\ell^2)$ operations algorithms and `EvalOne` is an $O(\ell^2 \log(\ell))$ operations algorithm.

Moreover, we give implementations of these algorithms and count these operations on k for odd prime numbers $3 \leq \ell < 200$ (Table 6 in Section 4). As a result, we determine a more efficient algorithm for each ℓ: for codomain, for $3 \leq \ell \leq 11$ and $\ell = 19, 23$. `CodOne` is more efficient than `CodSq`, `CodSq` is more efficient for all other ℓ. For evaluation, for all $3 \leq \ell < 200$, `EvalOne` is more efficient.

In addition, by using the most efficient algorithms selected above, we achieve an SIDH attack on B-SIDH in approximately 11 h (see Section 5).

Our implementation of the (ℓ, ℓ)-isogeny counting and an attack on B-SIDH is written in the computer algebra system `SageMath` [41] and is found at https://github.com/Yoshizumi-Ryo/ellell-isogeny_sage.

Related works. Corte-Real Santos, Costello, and Smith [11] proposed a method to compute $(3, 3)$-isogenies between Kummer surfaces. They implicitly utilized theta functions in their algorithm, but it should be noted that their algorithm is not derived from the Lubicz–Robert formula (that is, our proposed algorithm is completely different from the algorithm of [11]). As a result, their $(3, 3)$-isogeny computation algorithm is significantly more efficient than our algorithm (cf. [11, §4.3]).

Moreover, Corte-Real Santos and Flynn [12] generalized (ℓ, ℓ)-isogenies for any odd number ℓ. The asymptotic complexity of their algorithm with respect to ℓ is higher than that of theta-based algorithms, such as those based on the Lubicz–Robert formula and the Cosset–Robert formula [13]. However, as mentioned in [12, Section 6.3], for $\ell \leq 11$, their implementations outperformed the `AVIsogenies v0.7` [5], which is an implementation of the algorithm based on the Cosset–Robert formula. In addition, their algorithm outputs not only the defining equations of the codomain, but also the isogeny itself, in contrast to theta function-based algorithms.

By contrast, our algorithms are based on the Lubicz–Robert formula. We will show that the algorithm based on the Lubicz–Robert formula is more efficient than that based on the Cosset–Robert formula (see Remark 6). Thus, for a sufficiently large ℓ, it can be said that our algorithm is more efficient than the Corte-Real Santos and Flynn's algorithm. We compare the performance of our algorithm with Corte-Real Santos and Flynn's algorithm, both of which are implemented in `Magma` [6], in Appendix A.

Organisations. In Section 2, we introduce theta functions, their addition algorithms, and the Lubicz–Robert formula. In Section 3, we describe the arithmetic costs on Kummer surfaces and give relations for normalization. Then, we give explicit algorithms for the codomain and evaluation and give their asymptotic complexities. In Section 4, we count the number of operations of their algorithms and decide which is efficient for each odd prime number ℓ. In Section 5, we introduce the B-SIDH and SIDH attacks briefly and show the result of the implementation. Finally, Section 6 presents the conclusions.

2 Preliminaries

In this section, we summarize some facts about abelian varieties and theta functions [4,33,34,37] as well as relevant algorithms [26,28,29,37] that serve as bases of our proposed method.

For simplicity, we consider our arguments on the complex number field $\mathbb{C}$. However, using algebraic theta functions introduced by Mumford [33], these arguments are applicable even to the case of an algebraically closed field of characteristic p, where p is coprime to 2ℓ. For more details, we refer to [37].

In addition, we only consider the case of dimension $g = 2$, although the arguments of this section hold for general $g \geq 1$.

2.1 Theta functions

Let $\mathbb{H}_2$ denote the Siegel upper half-space of degree two defined by

$$\mathbb{H}_2 = \{\Omega \in M(2, \mathbb{C}) \mid {}^t\Omega = \Omega, \mathrm{Im}(\Omega) > 0\}$$

where $M(2, \mathbb{C})$ is the set of all 2×2 matrices with complex entries. Then, an abelian surface A over $\mathbb{C}$ is isomorphic to $\mathbb{C}^2/\Lambda_\Omega$, where $\Lambda_\Omega = \Omega\mathbb{Z}^2 \oplus \mathbb{Z}^2$ for some $\Omega \in \mathbb{H}_2$. In addition, this Ω determines a principal line bundle $\mathscr{L}_0$ on A. For any $a, b \in \mathbb{Q}^2$, the theta function with characteristics (a, b) is an analytic function given by

$$\theta\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right](z, \Omega) := \sum_{m \in \mathbb{Z}^2} \exp(\pi \mathrm{i}\ {}^t(m + a)\Omega(m + a) + 2\pi \mathrm{i}\ {}^t(m + a)(z + b))$$

for any $z \in \mathbb{C}^2$. We say that an analytic function f on $\mathbb{C}^2$ is a Λ_Ω-periodic function of level n if $f(z + m) = f(z)$ and $f(z + \Omega m) = \exp(-\pi \mathrm{i} n\ {}^t m\Omega m -$

$2\pi i n\ {}^t z m) f(z)$ for all $z \in \mathbb{C}^2$ and $m \in \mathbb{Z}^2$. Then, the set R_Ω^n of all Λ_Ω-periodic functions of level n is an n^2-dimensional $\mathbb{C}$-vector space. Moreover, the following n^2 functions $\theta\left[\begin{smallmatrix}0\\b\end{smallmatrix}\right](z, \frac{\Omega}{n})$ for $b \in \frac{1}{n}\mathbb{Z}^2/\mathbb{Z}^2$ form a basis of R_Ω^n [35]. Since $\theta\left[\begin{smallmatrix}0\\b\end{smallmatrix}\right](z, \frac{\Omega}{n}) = \theta\left[\begin{smallmatrix}0\\b+\beta\end{smallmatrix}\right](z, \frac{\Omega}{n})$ for all $b \in \mathbb{Q}^2$ and $\beta \in \mathbb{Z}^2$, these functions do not depend on the representation of $b \in \frac{1}{n}\mathbb{Z}^2/\mathbb{Z}^2$. We can identify R_Ω^n with the vector space $\Gamma(A, \mathscr{L}_0^n)$ of global sections, and thus the basis $\{\theta\left[\begin{smallmatrix}0\\b\end{smallmatrix}\right](z, \frac{\Omega}{n})\}_b$ of R_Ω^n gives the morphism

$$\begin{array}{rcl} \rho_n \colon A = \mathbb{C}^2/\Lambda_\Omega & \longrightarrow & \mathbb{P}^{n^2-1} \\ z & \longmapsto & (\theta\left[\begin{smallmatrix}0\\b\end{smallmatrix}\right](z, \frac{\Omega}{n}))_b \ . \end{array}$$

We call $\rho_n(0) \in \mathbb{P}^{n^2-1}$ the *theta-null point* and call $\rho_n(x)$ the *theta coordinate* of $x \in A$. We write $\theta_i(z) := \theta\left[\begin{smallmatrix}0\\i/n\end{smallmatrix}\right](z, \frac{\Omega}{n})$ for $i \in (\mathbb{Z}/n\mathbb{Z})^2$. Then, we have $\theta_i(-z) = \theta_{-i}(z)$. When $n \geq 3$, ρ_n is an embedding [4, Theorem 4.5.1]. When $n = 2$, since $\rho_2(-z) = \rho_2(z)$, $\rho_2 : A \to \mathbb{P}^3$ induces a morphism $K_A \to \mathbb{P}^3$ from a *Kummer surface* K_A, which is the quotient of A by automorphisms $\langle \pm 1_A \rangle$. If A is not a product of elliptic curves, this morphism $K_A \to \mathbb{P}^3$ is an embedding [4, Theorem 4.8.1].

Next, we consider Riemann relations (Theorem 1), and using these, we derive formulas for arithmetic operations on abelian surfaces. We first introduce the notion of the Riemann position [29, Definition 3.2].

Definition 1. *For any abelian group G, an 8-tuple $(g_1, g_2, g_3, g_4; g_1', g_2', g_3', g_4')$ of elements of G is said to be in* Riemann position *(on G) if there exists some element $h \in G$ such that $g_i' = g_i + h$ for $i = 1, \ldots, 4$ and $g_1 + g_2 + g_3 + g_4 = -2h$.*

Theorem 1 (Riemann relations [33], [26, Theorem 3.1]). *Let n be an even integer. For any 8-tuple $(z_1, z_2, z_3, z_4; z_1', z_2', z_3', z_4')$ of elements of $\mathbb{C}^2$ in Riemann position, any 8-tuple $(i_1, i_2, i_3, i_4; i_1', i_2', i_3', i_4')$ of elements of $(\mathbb{Z}/n\mathbb{Z})^2$ in Riemann position, and any character $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})^2}$ of the group $(\mathbb{Z}/2\mathbb{Z})^2$, we have*

$$\begin{aligned} &\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta_{i_1+t}(z_1)\theta_{i_2+t}(z_2)\right)\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta_{i_3+t}(z_3)\theta_{i_4+t}(z_4)\right) \\ = &\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta_{i_1'+t}(z_1')\theta_{i_2'+t}(z_2')\right)\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta_{i_3'+t}(z_3')\theta_{i_4'+t}(z_4')\right) \end{aligned}$$

Here, for the indices of functions $\theta_i(z)$, we regard $(\mathbb{Z}/2\mathbb{Z})^2$ as a subgroup of $(\mathbb{Z}/n\mathbb{Z})^2$ via the embedding $\overline{a} \mapsto \overline{\frac{n}{2}a}$ ($a \in \mathbb{Z}^2$).

The theta coordinates are given as projective coordinates on $\mathbb{P}^{n^2-1}$. However, to treat each component as an element of k, we have to fix their theta coordinates as affine coordinates on $\mathbb{A}^{n^2} \smallsetminus \{0\}$ considering some representatives. Here, we define this content precisely [29]:

Definition 2. *Let* $\kappa : \mathbb{A}^{n^2} \setminus \{0\} \to \mathbb{P}^{n^2-1}$ *be the natural projection. For* $x \in A$, *any preimage of* $\rho_{\underline{n}}(x)$ *under* κ *is called an* affine lift *of* x. *We write an affine lift of* x *as* $\tilde{x}$ *or* $(\tilde{\theta}_i(x))_i$. *For* $i \in (\mathbb{Z}/n\mathbb{Z})^2$, *we write the* i^{th}*-coordinate of* $\tilde{x}$ *as* $(\tilde{x})_i$ *or* $\tilde{\theta}_i(x)$. *For* $\lambda \in \mathbb{C}^*$ *and an affine lift* $\tilde{x}$, *we define* $\lambda * \tilde{x}$ *as* $(\lambda \cdot (\tilde{x})_i)_i$.

For later use, we extend the notion of Riemann relations to affine lifts.

Definition 3. *Let* $(x_1, x_2, x_3, x_4; x'_1, x'_2, x'_3, x'_4)$ *be at the Riemann position on* A *and* $(\tilde{x_1}, \ldots, \tilde{x'_4})$ *be their affine lifts. Then, we say that* $(\tilde{x_1}, \ldots, \tilde{x'_4})$ *satisfies Riemann relations if for any* $(i_1, i_2, i_3, i_4; i'_1, i'_2, i'_3, i'_4)$ *in Riemann position on* $(\mathbb{Z}/n\mathbb{Z})^2$ *and any character* $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})^2}$, *the following equation holds:*

$$\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)(\tilde{x_1})_{i_1+t}(\tilde{x_2})_{i_2+t}\right)\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)(\tilde{x_3})_{i_3+t}(\tilde{x_4})_{i_4+t}\right)$$
$$= \left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)(\tilde{x'_1})_{i'_1+t}(\tilde{x'_2})_{i'_2+t}\right)\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)(\tilde{x'_3})_{i'_3+t}(\tilde{x'_4})_{i'_4+t}\right) .$$

Then, by Riemann relations (Theorem 1), we have the following lemma:

Lemma 1. *For given* $(x_1, x_2, x_3, x_4; x'_1, x'_2, x'_3, x'_4)$ *being in Riemann position on* A *and any affine lifts* $\tilde{x_2}, \tilde{x_3}, \tilde{x_4}, \tilde{x'_1}, \tilde{x'_2}, \tilde{x'_3}, \tilde{x'_4}$, *there exists an affine lift* $\tilde{x_1}$ *such that* $(\tilde{x_1}, \ldots, \tilde{x'_4})$ *satisfies the Riemann relations in the sense of Definition 3.*

Proof. Let $z_1, \ldots, z'_4 \in \mathbb{C}^2$ be eight elements that represent $x_1, \ldots, x'_4 \in A$. Then, there are seven constants $\lambda_2, \ldots, \lambda'_4 \in \mathbb{C}^*$ such that $\lambda_2 * \tilde{x_2} = (\theta_i(z_2))_i, \ldots, \lambda'_4 * \tilde{x'_4} = (\theta_i(z'_4))_i$. Here, we take an affine lift of x_1 as $\tilde{x_1} := \frac{\lambda'_1\lambda'_2\lambda'_3\lambda'_4}{\lambda_2\lambda_3\lambda_4} * (\theta_i(z_1))_i$. By Theorem 1, $(\tilde{x_1}, \ldots, \tilde{x'_4})$ satisfies the Riemann relations in the sense of Definition 3. □

2.2 Arithmetic on Kummer surfaces

In this subsection, we consider some arithmetic operations on Kummer surfaces using theta functions of level $n = 2$ [28]. As mentioned in the previous subsection, if A is not a product of elliptic curves, the level 2 theta functions give the embedding of the Kummer surface to the projective space $K_A \to \mathbb{P}^3$.

In the following, we introduce some known methods for arithmetic calculation on Kummer surfaces using theta coordinates [28, Section 5]. Here, we assume that $A = \mathbb{C}^2/\Lambda_\Omega$ is not isomorphic to a product of elliptic curves as a principally polarized abelian surface. In other words, all abelian surfaces in this subsection are Jacobians of some genus-2 hyperelliptic curves. Note that if A is isomorphic to a product of elliptic curves as a polarized abelian surface, we can perform the arithmetic calculation by calculating on each elliptic curve.

The condition that A is the Jacobian of a genus-2 hyperelliptic curve is equivalent to the condition that the following ten values, called the even theta-null points of level $(2,2)$ are all non-zero:

$$\theta\left[\begin{smallmatrix} a/2 \\ b/2 \end{smallmatrix}\right](0,\Omega) \quad \text{for } a,b \in (\mathbb{Z}/2\mathbb{Z})^2 \text{ such that } {}^t a \cdot b = 0 \in \mathbb{Z}/2\mathbb{Z}.$$

For more details, see [25, Section 3.2]. Under this assumption, by the same argument as that in [25, Lemma 3], we have

$$\sum_{t \in (\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta_{i+t}(0)\theta_t(0) \neq 0 \tag{2}$$

for all $i \in (\mathbb{Z}/2\mathbb{Z})^2$ and $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})}^2$ such that $\chi(i) = 1 \in \langle \pm 1 \rangle$.

In the remainder of this paper, we fix an affine lift $(\tilde{\theta}_i(0))_i$ of the theta-null point. Here, we summarize the known methods for calculating the following arithmetic operations on Kummer surfaces:

Differential Addition: Given affine lifts $(\tilde{\theta}_i(x))_i, (\tilde{\theta}_i(y))_i, (\tilde{\theta}_i(x-y))_i$, output an affine lift $(\tilde{\theta}_i(x+y))_i$.

Doubling: Given an affine lift $(\tilde{\theta}_i(x))_i$, output an affine lift $(\tilde{\theta}_i(2x))_i$.

Three-way Addition: Given affine lifts $(\tilde{\theta}_i(x))_i, (\tilde{\theta}_i(y))_i, (\tilde{\theta}_i(z))_i$, $(\tilde{\theta}_i(x+y))_i$, $(\tilde{\theta}_i(y+z))_i$, $(\tilde{\theta}_i(z+x))_i$, output an affine lift $(\tilde{\theta}_i(x+y+z))_i$.

Scalar Multiplication: Given an affine lift $(\tilde{\theta}_i(x))_i$ and an integer N, output an affine lift $(\tilde{\theta}_i(Nx))_i$.

Normal Addition: Given affine lifts $(\tilde{\theta}_i(x))_i, (\tilde{\theta}_i(y))_i$, output a set of affine lifts $\{(\tilde{\theta}_i(x+y))_i, (\tilde{\theta}_i(x-y))_i\}$.

Compatible Addition: Given affine lifts $(\tilde{\theta}_i(y))_i, (\tilde{\theta}_i(z))_i, (\tilde{\theta}_i(x+y))_i$, $(\tilde{\theta}_i(x+z))_i$, output an affine lift $(\tilde{\theta}_i(y+z))_i$.

These concrete algorithms are presented in Section C.2 and their costs are presented in Section 3.2.

Remark 1. In our proposed isogeny algorithm in Section 3.4, we do not use Normal Addition and Compatible Addition since we provide enough information as inputs. Thus, we do not introduce them in this subsection and they are presented in Appendix B. However, these algorithms are required to construct attacks on B-SIDH in Section 5.2.

For Differential Addition, an affine lift $(\tilde{\theta}_i(x+y))_i$ could be obtained from given affine lifts $(\tilde{\theta}_i(x))_i, (\tilde{\theta}_i(y))_i, (\tilde{\theta}_i(x-y))_i$ by just applying Lemma 1 to $(x+y, x-y, 0, 0; y, -y, -x, -x)$ at the Riemann position on A. However, the computation can be made more efficient in the following manner. First, we note that for any $i \in (\mathbb{Z}/2\mathbb{Z})^2$, considering $(i, 0, i, 0; i, 0, i, 0)$ in Riemann position on

$(\mathbb{Z}/2\mathbb{Z})^2$, we have

$$\begin{aligned}&\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_{i+t}(x+y)\tilde{\theta}_t(x-y)\right)\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_{i+t}(0)\tilde{\theta}_t(0)\right)\\ =&\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_{i+t}(x)\tilde{\theta}_t(x)\right)\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_{i+t}(y)\tilde{\theta}_t(y)\right)\end{aligned} \tag{3}$$

where we used $\theta_i(x)=\theta_i(-x)$ and $\theta_i(y)=\theta_i(-y)$. Secondly, we define certain values z_i^χ and κ_{ij} as follows. For any $(i,\chi)\in(\mathbb{Z}/2\mathbb{Z})^2\times\widehat{(\mathbb{Z}/2\mathbb{Z})^2}$ such that $\chi(i)=1$, we define

$$z_i^\chi := \frac{\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_{i+t}(x)\tilde{\theta}_t(x)\right)\left(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_{i+t}(y)\tilde{\theta}_t(y)\right)}{\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_{i+t}(0)\tilde{\theta}_t(0)} \tag{4}$$

where the denominator of the right-hand side is not zero by (2). Here, these z_i^χ are computed from $(\tilde{\theta}_i(x))_i$ and $(\tilde{\theta}_i(y))_i$. Then, we define κ_{ij} for any $i,j\in(\mathbb{Z}/2\mathbb{Z})^2$ as follows:

$$\kappa_{ij} := \frac{1}{4}\sum_{\substack{\chi\in\widehat{(\mathbb{Z}/2\mathbb{Z})^2}\\ \text{s.t.}\chi(i+j)=1}}\frac{\chi(i)+\chi(j)}{2}z_{i+j}^\chi \quad . \tag{5}$$

Thus, we can calculate all κ_{ij} from the values z_i^χ such as $\chi(i)=1$ (note that κ_{ij} is symmetric with respect to i and j). Then, by the inverse Fourier transform, we have the following relations for $i,j\in(\mathbb{Z}/2\mathbb{Z})^2$:

$$\tilde{\theta}_i(x+y)\tilde{\theta}_j(x-y)+\tilde{\theta}_j(x+y)\tilde{\theta}_i(x-y)=2\kappa_{ij} \quad . \tag{6}$$

Differential Addition and Doubling Using equality (6), when $\tilde{\theta}_i(x-y)\neq 0$ for all i, we have

$$\tilde{\theta}_i(x+y)=\frac{\kappa_{ii}}{\tilde{\theta}_i(x-y)} \quad . \tag{7}$$

Thus, from the affine lifts $(\tilde{\theta}_i(x))_i,(\tilde{\theta}_i(y))_i,(\tilde{\theta}_i(x-y))_i$, we can calculate an affine lift $(\tilde{\theta}_i(x+y))_i$ satisfying (3). We call this operation *Differential Addition.* When $x=y$, we call this *Doubling.*

Remark 2. Even if $\tilde{\theta}_i(x-y)=0$ for some $i\in(\mathbb{Z}/2\mathbb{Z})^2$, we can still compute $(\tilde{\theta}_i(x+y))_i$. In fact, we first consider $j\in(\mathbb{Z}/2\mathbb{Z})^2$ such that $\tilde{\theta}_j(x-y)\neq 0$ and compute $\tilde{\theta}_j(x+y)$ using (7). Then, for $i\in(\mathbb{Z}/2\mathbb{Z})^2\setminus\{j\}$, by (6), we have

$$\tilde{\theta}_i(x+y)=\frac{2\kappa_{ij}-\tilde{\theta}_j(x+y)\tilde{\theta}_i(x-y)}{\tilde{\theta}_j(x-y)} \quad .$$

Three-way Addition. For given affine lifts $(\tilde{\theta}_i(x))_i, (\tilde{\theta}_i(y))_i, (\tilde{\theta}_i(z))_i$, $(\tilde{\theta}_i(x+y))_i, (\tilde{\theta}_i(y+z))_i, (\tilde{\theta}_i(z+x))_i$, we can calculate $(\tilde{\theta}_i(x+y+z))_i$ as follows. Note that this Three-way Addition algorithm does not always work on A but works on some Zariski dense subset of A. For details, we refer to [27, Section 3.6]. Here, for simplicity, we assume $\tilde{\theta}_i(x) \neq 0, \tilde{\theta}_i(y) \neq 0$, and $\tilde{\theta}_i(z) \neq 0$ for all i, and under this condition, the Three-way Addition algorithm works. First, for $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})^2}$, we define

$$E^{\chi} := \frac{(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\tilde{\theta}_t(0)\tilde{\theta}_t(y+z)) \cdot (\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\tilde{\theta}_t(z+x)\tilde{\theta}_t(x+y))}{\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\tilde{\theta}_t(y)\tilde{\theta}_t(z)} \ . \tag{8}$$

These E^{χ} are computed from the given affine lifts. By applying Lemma 1 to points $(x+y+z, x, y, z; 0, -y-z, -z-x, -x-y)$ in Riemann position on A and by focusing (among the resulting Riemann relations) on indices $(0,0,0,0;0,0,0,0)$ in Riemann position on $(\mathbb{Z}/2\mathbb{Z})^2$, for any $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})^2}$, we have $\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\tilde{\theta}_t(x+y+z)\tilde{\theta}_t(x) = E^{\chi}$. Then, by the inverse Fourier transform, for any $i \in (\mathbb{Z}/2\mathbb{Z})^2$, we have

$$\tilde{\theta}_i(x+y+z) = \frac{\sum_{\chi\in\widehat{(\mathbb{Z}/2\mathbb{Z})^2}} \chi(i)E^{\chi}}{4\tilde{\theta}_i(x)} \ . \tag{9}$$

Thus, we have obtained the affine lift $(\tilde{\theta}_i(x+y+z))_i$. This operation is called *Three-way Addition* (or *Extended Addition*).

Scalar Multiplication For a given affine lift $(\tilde{\theta}_i(x))_i$ and any integer $N \geq 3$, there are various methods of calculating the affine lift of Nx, and the result is denoted by $\texttt{Mult}(N, (\tilde{\theta}_i(x))_i)$. One method to compute it is using the Montgomery ladder [32]. This method requires $n-1$ Doubling and n Differential Addition, where n is the bit length of $N-1$. In our implementation in Section 4, we used this calculation.

Remark 3. Let $x_1, \ldots, x_r \in A$ be any element, $(\tilde{\theta}_i(x_j))_i, (\tilde{\theta}_i(x_{j_1}+x_{j_2}))_i$ be affine lifts for $1 \leq j \leq r$ and $1 \leq j_1 < j_2 \leq r$, and $m_1, \ldots, m_r \in \mathbb{Z}$ be any integer. Then, we can compute an affine lift $(\tilde{\theta}_i(m_1x_1 + \cdots + m_rx_r))_i$ in several different ways by using Differential Addition, Doubling, Three-way Addition, and Scalar Multiplication. Then, the computation result does not depend on the order of these operations (cf. [26, Corollary 3.13]).

2.3 Excellent lifts

Here, we recall the notion of excellentness for some conditions (cf. [29, Definitions 3.6, 3.7, 3.10]). In the following definition, $\texttt{Multadd}(N, \tilde{x}, \tilde{y}, \widetilde{x+y})$ denotes the affine lift of $Nx+y \in A$ computed from affine lifts $\tilde{x}, \tilde{y}, \widetilde{x+y}$.

Definition 4. *Let ℓ be any odd number and $K \subset A[\ell]$ be a maximal isotropic subgroup of rank g with respect to the Weil pairing.*

1. *For any ℓ-torsion point $e \in A[\ell]$, an affine lift $\tilde{e}$ of e is said to be* excellent *if* $\mathtt{Mult}(\ell'+1, \tilde{e}) = \mathtt{Mult}(\ell', \tilde{e})$ *as affine lifts where* $\ell' = \frac{\ell-1}{2}$.
2. *A set of affine lifts $\tilde{K} = \{\tilde{e} \mid e \in K\}$ of K is said to be* excellent *if for any eight elements in Riemann position on K, their affine lifts in $\tilde{K}$ satisfy Riemann relations according to Definition 3.*
3. *For any affine lift $\tilde{x}$ of $x \in A$ and an excellent lift $\tilde{e}$ of $e \in A[\ell]$, an affine lift $\widetilde{x+e}$ of $x+e$ is said to be* excellent *with respect to $\tilde{x}$ and $\tilde{e}$ if* $\mathtt{Multadd}(\ell, \tilde{e}, \tilde{x}, \widetilde{x+e}) = \tilde{x}$ *as affine lifts.*
4. *For any excellent lift $\tilde{K}$ and any affine lift $\tilde{x}$ of $x \in A$, a set of affine lifts $\widetilde{x+K} = \{\widetilde{x+e} \mid e \in K\}$ is said to be* excellent *with respect to $\tilde{x}$ and $\tilde{K}$ if for any eight elements at the Riemann position on A included in $K \cup (x+K)$. Their affine lifts in $\tilde{K} \cup \widetilde{x+K}$ satisfy Riemann relations according to Definition 3.*

Remark 4. The condition $\mathtt{Mult}(\ell'+1, \tilde{e}) = \mathtt{Mult}(\ell', \tilde{e})$ in 1 of Definition 4 is equivalent to satisfying two conditions: $\mathtt{Mult}(\ell, \tilde{e}) = (\tilde{\theta}_i(0))_i$ and $\mathtt{Mult}(\ell-1, \tilde{e}) = \tilde{e}$. This fact can be proven using Lemma 7 in Section 3.3.

Theorem 2 ([29, Theorems 3.8, 3.11]). *With the notation above, the following statements hold:*

(i) For any basis $\{e_1, e_2\}$ of $K \simeq (\mathbb{Z}/\ell\mathbb{Z})^2$ and excellent lifts $\tilde{e_1}, \tilde{e_2}, \widetilde{e_1+e_2}$, a set $\tilde{K}$ of affine lifts of K computed from $\tilde{e_1}, \tilde{e_2}, \widetilde{e_1+e_2}$ is excellent.
(ii) Let $\tilde{K}$ be any excellent lift of K and $\tilde{x}$ be any affine lift of $x \in A$. In addition, let $\widetilde{x+e_1}, \widetilde{x+e_2}$ be excellent lifts. Then, a set $\widetilde{x+K}$ of affine lifts of $x+K$ computed from them is excellent.

Remark 5. For an excellent lift $\tilde{e}$ and $\lambda \in \mathbb{C}^*$, $\lambda * \tilde{e}$ is also excellent if and only if $\lambda^\ell = 1$ by Lemma 7 in Section 3.3. Therefore, excellent lifts of e are not necessarily unique and are at most finitely many.

2.4 Lubicz–Robert formula

In this subsection, we introduce an isogeny calculation formula given by Lubicz–Robert [29]. In their paper, the formula is given for a general dimension and general even level theta structure. Here, we simply use the formula in dimension 2 and level 2 theta structure, that is, on Kummer surfaces. For a theta structure, we refer to [29,33].

Let k be an algebraically closed field of characteristic 0 or $p > 0$ where p is coprime to 2. Let $(A, \mathscr{L}_0)$ be a principal polarized abelian surface over k, $\mathscr{L} = \mathscr{L}_0^2$, and $\Theta_{\mathscr{L}}$ be a symmetric theta structure. In addition, ℓ be an odd number and $K \subset A[\ell]$ be a maximal isotropic subgroup of rank g with respect to the Weil pairing. Then, the isogeny $f : A \to B = A/K$ induces a line bundle on B and a symmetric theta structure. From an excellent lift $\tilde{K}$ of K, the formula ((10) in Theorem 3) gives the theta-null point $(\theta_i^B(0))_i$ of B. Moreover, for $x \in A$, an affine lift $\tilde{x}$, and excellent lifts $\widetilde{x+K}$, the formula ((11) in Theorem 3) gives a theta coordinate $(\theta_i^B(f(x)))_i$ of $f(x) \in B$.

Theorem 3 ([29, Corollary 4.6]). *The notation is the same as above. Let $\tilde{K}$ be an excellent lift of K and $a_1, \ldots, a_r$ be positive integers such that $\ell = \sum_{u=1}^{r} a_u^2$. For any $i \in (\mathbb{Z}/2\mathbb{Z})^2$, up to multiplication by a constant not depending on i, we have*

$$\theta_i^B(0) = \sum_{e \in K} \prod_{u=1}^{r} \texttt{Mult}(a_u, \tilde{e})_{a_u i} \ . \tag{10}$$

For $x \in A$, let $\tilde{x}$ be any affine lift and $\widetilde{x+K}$ be an excellent lift with respect to $\tilde{x}$ and $\tilde{K}$. Then, for any $i \in (\mathbb{Z}/2\mathbb{Z})^2$, up to multiplication by a constant not depending on i, we have

$$\theta_i^B(f(x)) = \sum_{e \in K} \prod_{u=1}^{r} \texttt{Mult}(a_u, \widetilde{x+e})_{a_u i} \ . \tag{11}$$

Here, we consider the complexity. By Lagrange's four-square theorem, we can take a representation $\ell = \sum_{u=1}^{r} a_u^2$ such that $r \leq 4$. Thus, we assume $r = O(1)$ here, and O_r denotes the big O notation under the assumption that $r = O(1)$. For (10), first, for a basis $\{e_1, e_2\}$ of K, we compute $\texttt{Mult}(a_u, e_1), \texttt{Mult}(a_u, e_2), \texttt{Mult}(a_u, e_1 + e_2)$ for all $1 \leq u \leq r$. These computations require $O_r(\log(\ell))$ arithmetic operations. Second, we compute their linear combinations $\texttt{Mult}(a_u, m_1 e_1 + m_2 e_2)$ for all $0 \leq m_1, m_2 < \ell$ and $1 \leq u \leq r$. These computations require $O_r(\ell^2)$ arithmetic operations. Third, for each $e \in K$, we compute the products running over $1 \leq u \leq r$. This calculation requires $O_r(\ell^2)$ arithmetic operations. From the above, computing (10) requires $O_r(\ell^2)$ arithmetic operations. The case for (11) is similar (cf. [29, p.16]).

As a special case, in the formulas (10), (11), considering $\ell = 1^2 + \cdots + 1^2$, we obtain the following formulas:

$$\theta_i^B(0) = \sum_{e \in K} (\tilde{e})_i^{\ell} \ , \tag{12}$$

$$\theta_i^B(f(x)) = \sum_{e \in K} (\widetilde{x+e})_i^{\ell} \ . \tag{13}$$

For the formulas (12), (13), we need to calculate the ℓ^{th} power on k for each $e \in K$. Thus, their complexities are $O(\ell^2 \log(\ell))$. Note that, irrespective of the asymptotically higher complexity than that of the previous method, the current method may still be more efficient for some concrete choice of ℓ.

In [29], the isogeny calculation algorithm based on (11) (for general dimension and general level) is given as [29, Algorithm 4]. Here, we consider the case of dimension 2 and level 2 as before. In the algorithm, we first compute the excellent lift of $e_1, e_2, e_1 + e_2, x, x + e_1, x + e_2$ where $\{e_1, e_2\}$ is a basis of kernel K. Then, we compute the excellent lifts $\tilde{K}$ and $\widetilde{x+K}$. Finally, we compute the right-hand side of (11) for $i \in (\mathbb{Z}/2\mathbb{Z})^2$.

Remark 6. In [13], Cosset–Robert gave another (ℓ, ℓ)-isogeny calculation algorithm based on Koizumi's formula [24] in $O(\ell^r)$ operations, where $r = 2$ when

$\ell \equiv 1 \pmod 4$ and $r = 4$ when $\ell \equiv 3 \pmod 4$. In the same manner as that above, we write $\ell = \sum_{u=1}^{r} a_u^2$ with $a_u \in \mathbb{N}$. Moreover, let $\tilde{K}$ be an excellent lift of the kernel $K \simeq (\mathbb{Z}/\ell\mathbb{Z})^2$. Let F be an integer $(r \times r)$-matrix such that $F^t F = \ell\,\mathrm{id}_r$ and the first row is $(a_1, \dots, a_r)$. Now, we define $F_K : K^r \to K^r$ as the $\mathbb{F}_\ell$-linear map induced by the matrix F. In [13, Equation (6)], the formula to compute the theta-null point of the codomain is

$$\theta_i^B(0) = \sum_{{}^t(e_1,\dots,e_r) \in \mathrm{Ker}(F_K)} \prod_{u=1}^{r} (\tilde{e}_u)_{a_u i} \tag{14}$$

up to multiplication by a constant. Here, for any $e \in K$, we have ${}^t(a_1 e, \dots, a_r e) \in \mathrm{Ker}(F_K)$. Hence, in (14), we need to compute and take the sum of the values $\prod_{u=1}^{r} \widetilde{(a_u e)}_{a_u i}$ for each $e \in K$. Thus, the complexity of (10) is the same as or more efficient than the complexity of (14). For more details and general arguments, we refer to [37, Section 4.4.3].

3 Proposed Algorithms

In this section, we propose some explicit algorithms of isogeny calculations between Kummer surfaces based on the Lubicz–Robert formula (Theorem 3). As described in Section 2.4, k is an algebraically closed field of characteristic 0 or $p > 0$ where p is coprime to 2. As noted at the beginning of Section 2, the arguments of Sections 2.1 and 2.2 are applicable to the case of not only $\mathbb{C}$ but also the above k.

In our algorithms, calculations of the multiplicative inverse on k are avoided as they are expensive, especially in cryptographic situations. Hence, we evaluate the costs of the algorithms by counting the multiplication and square operations on k.

Throughout this section, A is an abelian surface over k, $\mathscr{L} = \mathscr{L}_0^2$ is a totally symmetric line bundle where $\mathscr{L}_0$ is principal, and $\Theta_{\mathscr{L}}$ is a symmetric theta structure of level 2 for $(A, \mathscr{L})$. In addition, ℓ is an odd number, and $K \subset A[\ell]$ is a maximal isotropic subgroup of rank g with respect to the Weil pairing. Then, $B := A/K$ has an induced level-2 symmetric theta structure. Moreover, $f : A \to B$ is the isogeny with kernel K.

First, in Section 3.1, we give an overview of our algorithm. In Section 3.2, we present the arithmetic costs on Kummer surfaces given in Section 2.2. The results will be used in our isogeny calculation algorithms. Then, in Section 3.3, we present relations on the excellent lifts. In Section 3.4, we present the explicit algorithms of isogeny calculations. In Section 3.5, for odd prime numbers ℓ, we investigate a representation $\ell = \sum_{u=1}^{r} a_u^2$.

3.1 Overview of our proposed algorithms

In this subsection, we introduce an overview of our isogeny calculation algorithms. The explicit algorithms are provided in Section 3.4.

As seen in Theorem 3, the theta-null point of codomain $B := A/K$ can be computed from an excellent lift $\tilde{K}$ of the kernel K. In addition, for $x \in A$, the theta coordinate of the image $f(x)$ can be computed from an excellent lift $\widetilde{x+K}$. Here, we construct algorithms of the codomain and evaluation with the following inputs and outputs. We basically denote an excellent lift by $\tilde{e}$ and any affine lift by $\overline{e}$ for $e \in A$.

Codomain:

Input: Any affine lifts $\overline{e_1}, \overline{e_2}, \overline{e_1 + e_2}$ of $e_1, e_2, e_1 + e_2$ for a basis $\{e_1, e_2\}$ of K.

Output: Theta-null point $(\theta_i^B(0))_i$ of B.

Evaluation:

Input: Any affine lifts $\overline{e_1}, \overline{e_2}, \overline{e_1 + e_2}, \tilde{x}, \overline{x + e_1}, \overline{x + e_2}$ of $e_1, e_2, e_1 + e_2, x, x + e_1, x + e_2$ for a basis $\{e_1, e_2\}$ of K and any point $x \in A$.

Output: Theta coordinate $(\theta_i^B(f(x)))_i$ of $f(x) \in B$.

For both cases, we consider two representations, $\ell = \sum_{u=1}^{r} a_u^2$ and $\ell = 1^2 + \cdots + 1^2$. Concretely, for the codomain, we consider two algorithms `CodSq` based on (10) and `CodOne` based on (12). Similarly, for evaluation, we consider two algorithms `EvalSq` based on (11) and `EvalOne` based on (13), as shown in Table 1 in Section 1. As noted in Section 2.4, `CodSq` and `EvalSq` require $O_r(\ell^2)$ operations, while `CodOne` and `EvalOne` require $O(\ell^2 \log(\ell))$ operations. Here, O_r denotes the big O notation under the assumption that $r = O(1)$. Moreover, in Section 3.5, we will discuss the former representation $\ell = \sum_{u=1}^{r} a_u^2$ in more detail.

For `CodSq`, First, we compute the affine lifts $\overline{s_1 e_2 + s_2 e_2}$ for $0 \leq s_1, s_2 < \ell$. Then, instead of computing an affine lift $\overline{m_1 a_u e_1 + m_2 a_u e_2}$, we use $\overline{s_1 e_1 + s_2 e_2}$ where $0 \leq s_1, s_2 < \ell$ and $m_1 a_u \equiv s_1 \pmod{\ell}$, $m_2 a_u \equiv s_2 \pmod{\ell}$. Note that the above $\overline{m_1 a_u e_1 + m_2 a_u e_2}$ and $\overline{s_1 e_1 + s_2 e_2}$ are in general different as affine lifts. This difference is important when constructing the excellent lifts and will be discussed in Section 3.3.

In addition, for `CodSq` and `CodOne`, we do not need to compute $\overline{m_1 e_1 + m_2 e_2}$ for all $0 \leq m_1, m_2 < \ell$. It is sufficient to compute them for half of $0 \leq m_1, m_2 < \ell$, since $m_1 e_1 + m_2 e_2 = (\ell - m_1) e_1 + (\ell - m_2) e_2$. For more details, see Section 3.4.

We note that the inputs for our proposed algorithms are affine lifts such as $\overline{e_1}, \overline{e_2}, \overline{e_1 + e_2}$, while (10) and (11) require excellent lifts. Thus, we need to compute relations between affine lifts and excellent lifts. This is discussed further in Section 3.3.

3.2 Algorithms of arithmetic on Kummer surfaces

In this subsection, we describe explicit algorithms and their costs of calculation methods given in Section 2.2.

As in Section 2.2, we use affine lifts $(\tilde{\theta}_i(x))_{i \in (\mathbb{Z}/2\mathbb{Z})^2}$ of level-2 theta coordinates and algorithms based on Riemann relations of Definition 3, which take affine lifts as inputs and produce affine lifts as outputs. Note that we can select and fix any affine lift of the theta-null point at the very beginning.

Notation. In our algorithms, to avoid calculating inverse elements on k, we often hold an element in k as a fraction, that is. Then, if we hold an element $a \in k$ as a pair of $n \in k$ and $d \in k$ such that $a = \frac{n}{d}$, we write the data as (n, d) .

Moreover, we always hold any affine lift $\tilde{\theta}_{00}(x), \tilde{\theta}_{01}(x), \tilde{\theta}_{10}(x), \tilde{\theta}_{11}(x)$ as five elements $\theta'_{00}(x), \theta'_{01}(x),\ \theta'_{10}(x),\ \theta'_{11}(x),\ d_x \in k$ such that $\tilde{\theta}_i(x) = \frac{\theta'_i(x)}{d_x}$ (with common denominator d_x) for all $i \in (\mathbb{Z}/2\mathbb{Z})^2$. In this case, we write the data as $(\theta'_i(x), d_x)_i$, and write $\theta'_i(x) = \mathtt{Num}((\tilde{\theta}_i(x))_i, i)$ and $d_x = \mathtt{Den}((\tilde{\theta}_i(x))_i)$. Since we can take any affine lift $(\tilde{\theta}_i(0))_i$ of the theta-null point, we select $(\tilde{\theta}_i(0))_i$ with denominator $d_0 = 1$. We omit the affine lift of the theta-null point from inputs for algorithms.

To count the numbers of operations in the algorithms, we indicate a multiplication (resp. square) operation on the base field k by M (resp. S). Moreover, we indicate a multiplication (resp. square) operation computed only from the theta-null point by M_0 (resp. S_0). The values are reused after being computed once. We do not count the numbers of addition on k and arithmetic operations on $\mathbb{Z}$. We note that since we hold some elements as fractions, the counting results are not equal to the existing results, such as in [28], although the computation methods are similar.

Lemma 2. *For any integers $n, N \geq 1$, let $(b_{i,j}, a_i)$ for $0 \leq i \leq n-1$ and $0 \leq j \leq N-1$ be nN fractions in k. Then, the following statements hold:*

(i) We can reduce the fractions to

$$\mathtt{Commondenom}\left((b_{i,j}, a_i)_{\substack{0\leq i\leq n-1 \\ 0\leq j\leq N-1}}\right) := (b_{i,j} a_0 \cdots a_{i-1} a_{i+1} \cdots a_{N-1}, \alpha)_{\substack{0\leq i\leq n-1 \\ 0\leq j\leq N-1}}$$

with common denominator $\alpha = a_0 \cdots a_{N-1}$ in $C_{cd}(N, n) := ((n+3)N-5)\mathsf{M}$.

(ii) We can compute only the numerators of the result of (i):

$$\mathtt{Projcommondenom}\left((b_{i,j}, a_i)_{\substack{0\leq i\leq n-1 \\ 0\leq j\leq N-1}}\right) := (b_{i,j} a_0 \cdots a_{i-1} a_{i+1} \cdots a_{N-1})_{\substack{0\leq i\leq n-1 \\ 0\leq j\leq N-1}}$$

in $C_{pcd}(N, n) := ((n+3)N-6)\mathsf{M}$.

Proof. (i) For N elements $a_0, \ldots, a_{N-1} \in k^*$, by Lemma 14 in Appendix C.1, we can compute N elements $a_0 \cdots a_{M-1} a_{M+1} \cdots a_{N-1}$ for $0 \leq M \leq N-1$ and a product $\alpha = a_0 \cdots a_{N-1}$ in $(3N-5)\mathsf{M}$. After that we multiply $b_{M,m}$ by numerators $a_0 \cdots a_{M-1} a_{M+1} \cdots a_{N-1}$ for $0 \leq M \leq N-1$ and $0 \leq m \leq n-1$ in $nN\mathsf{M}$. Thus, we can compute the fractions with a common denominator in $((n+3)N-5)\mathsf{M}$.

(ii) This is the same as (i), except for computing α.

□

Arithmetic costs on Kummer surfaces. Next, we evaluate costs of arithmetics on Kummer surfaces. We give the cost to compute κ_{ii} defined by (5) as follows. First, we calculate z_0^χ using (4) for $i=0$:

$$z_0^\chi = \frac{(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_t(x)^2)(\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_t(y)^2)}{\sum_{t\in(\mathbb{Z}/2\mathbb{Z})^2}\chi(t)\tilde{\theta}_t(0)^2} \ .$$

Then, we calculate κ_{ii} using (5) for $i=j$, that is, $\kappa_{ii} = \frac{1}{4}\sum_{\chi\in\widehat{(\mathbb{Z}/2\mathbb{Z})^2}}\chi(i)z_0^\chi$.

The next lemma is approximately the same as [28, Lemma 5.1] except that we hold each affine lift as fractions with a common denominator. From Algorithm 6 in Appendix C.2, we have the following number of arithmetic operations:

Lemma 3. *With the above notation, the following statements hold:*

(i) Computing κ_{ii} for all $i\in(\mathbb{Z}/2\mathbb{Z})^2$ requires $4\mathsf{S}_0+9\mathsf{S}+17\mathsf{M}$.
(ii) When $x=y$, (i) reduces to $4\mathsf{S}_0+10\mathsf{S}+12\mathsf{M}$.

Once we calculate $(\kappa_{ii})_i$ for some x_1 and y_1, we can reuse $\theta'_i(0)^2$ for other x_2 and y_2. Thus, we will count the $4\mathsf{S}_0$ only once.

The following lemma gives the number of arithmetic operations for Differential Addition and Doubling based on (7). From Algorithm 7 in Appendix C.2, we have the following number of arithmetic operations:

Lemma 4 (Differential Addition, Doubling). *For given any affine lifts $(\tilde{\theta}_i(x))_i$, $(\tilde{\theta}_i(y))_i$, $(\tilde{\theta}_i(x-y))_i$ with $\tilde{\theta}_i(x-y)\neq 0$ for all i, computing the affine lift $(\tilde{\theta}_i(x+y))_i$ requires $C_{dfa} := 4\mathsf{S}_0+9\mathsf{S}+33\mathsf{M}$. When $x=y$, the cost reduces to $C_{dbl} := 4\mathsf{S}_0+10\mathsf{S}+28\mathsf{M}$.*

Remark 7. As mentioned in Remark 2, we can compute $(\tilde{\theta}_i(x+y))_i$ if $\tilde{\theta}_i(x-y)=0$ for some i. However, for simplicity, in our algorithms below, we always assume the condition $\tilde{\theta}_i(x-y)\neq 0$ for all i when using Differential Addition. Note that if $z\in A$ is 4-torsion point, this assumption $\tilde{\theta}_i(z)\neq 0$ often does not hold. Similarly, our implementation works only on this assumption. Unless we consider 4-torsion points, this assumption almost certainly holds experimentally.

Remark 8. After a calculation of $(\tilde{\theta}_i(x+y))_i$ once, the cost to calculate $(\tilde{\theta}_i(x+z))_i$ using Differential Addition reduces to $C_{rdfa} := 5\mathsf{S}+33\mathsf{M}$, since we can reuse the data $\theta'_i(0)^2$, $\theta'_i(x)^2$.

The following lemma gives the number of arithmetic operations for Three-way Addition based on (8) and (9). From Algorithm 8 in Appendix C.2, we have the following number of arithmetic operations:

Lemma 5 (Three-way Addition). *For given affine lifts $(\tilde{\theta}_i(x))_i$, $(\tilde{\theta}_i(y))_i$, $(\tilde{\theta}_i(z))_i$, $(\tilde{\theta}_i(x+y))_i$, $(\tilde{\theta}_i(y+z))_i$, $(\tilde{\theta}_i(z+x))_i$ with $\tilde{\theta}_i(x)\neq 0, \tilde{\theta}_i(y)\neq 0, \tilde{\theta}_i(z)\neq 0$ for all i, computing $(\tilde{\theta}_i(x+y+z))_i$ using (8) and (9) requires* $48\mathsf{M}$.

The following lemma is used in isogeny calculations. Here, for any odd number ℓ and $\ell' := \frac{\ell-1}{2}$, we define a subset $H_\ell\subset\mathbb{Z}^2$ as

$$\begin{aligned} H_\ell := &\{(m_1,0)\in\mathbb{Z}^2 \mid 1\le m_1\le \ell'\} \sqcup \{(0,m_2)\in\mathbb{Z}^2 \mid 1\le m_2\le\ell'\} \\ &\sqcup\{(m_1,m_2)\in\mathbb{Z}^2 \mid 1\le m_1,\ 1\le m_2,\ m_1+m_2<\ell\} \\ &\sqcup\{(m_1,m_2)\in\mathbb{Z}^2 \mid \ell'<m_1<\ell,\ m_1+m_2=\ell\} \ . \end{aligned} \tag{15}$$

If we define $\overline{H_\ell} := \{(\overline{m_1}, \overline{m_2}) \in (\mathbb{Z}/\ell\mathbb{Z})^2 \mid (m_1, m_2) \in H_\ell\}$, then for any $x \in (\mathbb{Z}/\ell\mathbb{Z})^2 \setminus \{0\}$, we have $x \in \overline{H_\ell}$ if and only if $-x \notin \overline{H_\ell}$.

Lemma 6. *With the notation above, we have the following costs:*

(i) For given affine lifts $(\tilde{\theta}_i(e_1))_i, (\tilde{\theta}_i(e_2))_i$, and $(\tilde{\theta}_i(e_1 + e_2))_i$, computing all affine lifts $(\tilde{\theta}_i(m_1 e_1 + m_2 e_2))_i$ for $(m_1, m_2) \in H_\ell$ requires $C_{hlc}(\ell) := 2C_{dbl} + (\frac{\ell^2-1}{2} - 5)C_{rdfa}$ when $\ell \geq 5$. When $\ell = 3$, it requires one Differential Addition; thus, $C_{hlc}(3) := C_{dfa}$.

(ii) For given affine lifts $(\tilde{\theta}_i(e_1))_i, (\tilde{\theta}_i(e_2))_i, (\tilde{\theta}_i(e_1+e_2))_i$, $(\tilde{\theta}_i(x))_i$, $(\tilde{\theta}_i(x+e_1))_i$, $(\tilde{\theta}_i(x + e_2))_i$, computing all affine lifts $(\tilde{\theta}_i(x + m_1 e_1 + m_2 e_2))_i$ for $0 \leq m_1, m_2 < \ell$ requires $C_{lc+}(\ell) := 48\mathsf{M} + 2C_{dfa} + (\ell^2 - 6)C_{rdfa}$ when $\ell \geq 3$.

Proof. For (i), since $\#H_\ell = \frac{\ell^2-1}{2}$ and we already have $(\tilde{\theta}_i(e_1))_i, (\tilde{\theta}_i(e_2))_i, (\tilde{\theta}_i(e_1+e_2))_i$, the number of (m_1, m_2) not having $(\tilde{\theta}_i(m_1 e_1 + m_2 e_2))_i$ is $\frac{\ell^2-1}{2} - 3$. Among them, we can compute $(\tilde{\theta}_i(2e_1))_i$, $(\tilde{\theta}_i(2e_2))_i$ by Doubling. Then, we reuse some values to compute other affine lifts; see Remark 8. For (ii), we first compute $(\tilde{\theta}_i(x + e_1 + e_2))_i$ by Three-way Addition in $48\mathsf{M}$. For the remaining (m_1, m_2), is that the process is similar to (i). ☐

In the above lemma, asymptotically, we have $C_{hlc}(\ell) = \frac{5(\ell^2-1)}{2}\mathsf{S} + \frac{33(\ell^2-1)}{2}\mathsf{M} + O(1)\mathsf{M}$ and $C_{lc+}(\ell) = 5\ell^2\mathsf{S} + 33\ell^2\mathsf{M} + O(1)\mathsf{M}$.

3.3 Normalization

In this subsection, as noted in Section 3.1, we give relations of affine lifts and excellent lifts.

First, we give a fundamental equality used later. This lemma is a generalization of [26, Lemma 3.10] and [27, Lemma 2]. Moreover, it is essentially the same claim as [40, Theorem 10.1.1].

Lemma 7. *Let $x_1, \ldots, x_r \in A$ and let $\overline{x_i}, \overline{x_i + x_j}$ be any affine lifts for $1 \leq i \leq r, 1 \leq i < j \leq r$. Let $\overline{\sum_{i=1}^r m_i x_i}$ for $m_i \in \mathbb{Z}$ be the affine lifts computed from $\overline{x_i}, \overline{x_i + x_j}$ by using computation of Section 2.2. In addition, we take any $\lambda_i, \lambda_{ij} \in k^*$, and we put $\tilde{x_i} := \lambda_i * \overline{x_i}$ and $\widetilde{x_i + x_j} := \lambda_{ij} * \overline{x_i + x_j}$. Let $\widetilde{\sum_{i=1}^r m_i x_i}$ for $m_i \in \mathbb{Z}$ be the affine lift computed from $\tilde{x_i}, \widetilde{x_i + x_j}$. Then, we have*

$$\widetilde{\sum_{i=1}^r m_i x_i} = \left(\left(\prod_{1 \leq i \leq r} \lambda_i^{m_i^2} \right) \cdot \left(\prod_{1 \leq i < j \leq r} \left(\frac{\lambda_{ij}}{\lambda_i \lambda_j} \right)^{m_i m_j} \right) \right) * \overline{\sum_{i=1}^r m_i x_i} \ .$$

Proof. We show the claim by induction for $r \geq 1$. The case of $r = 1$ is just Equation (17) of [26, Lemma 3.10]. Next, we consider the case of $r = 2$. The case of $m_2 = 1$ is just Equation (16) of [26, Lemma 3.10]. For a general integer m_2, we have

$$\begin{aligned}
\widetilde{m_1x_1 + m_2e_2} &= \texttt{Multadd}(m_2, \tilde{x_2}, \widetilde{m_1x_1}, \widetilde{m_1x_1 + x_2}) \\
&= \texttt{Multadd}\left(m_2, \lambda_2 * \overline{x_2}, \lambda_1^{m_1^2} * \overline{m_1x_1}, \left(\lambda_1^{m_1^2}\lambda_2\left(\frac{\lambda_{12}}{\lambda_1\lambda_2}\right)^{m_1}\right) * \overline{m_1x_1 + x_2}\right) \\
&= \left(\lambda_1^{m_1^2}\lambda_2^{m_2^2}\left(\frac{\lambda_{12}}{\lambda_1\lambda_2}\right)^{m_1m_2}\right) * \overline{m_1x_1 + m_2x_2} \ .
\end{aligned}$$

Thus, we obtained the result for $r = 2$. Next, we assume that the result holds for r. Here, $\widetilde{m_1e_1 + \cdots + m_{r+1}e_{r+1}}$ is the result of Three-way Addition of $\widetilde{m_1e_1 + \cdots + m_{r-1}e_{r-1}}$ and $\widetilde{m_re_r}$ and $\widetilde{m_{r+1}e_{r+1}}$. $\overline{m_1e_1 + \cdots + m_{r+1}e_{r+1}}$ is similar. Thus, from [27, Lemma 2] and the induction hypothesis, we obtain the result for $r + 1$. □

Codomain When we use the Lubicz–Robert formula, we need excellent lifts of the kernel. $(A, \mathscr{L}, \Theta_{\mathscr{L}})$ and $K \subset A[\ell]$ are the same notations as earlier.

For $e \in K$, let $\overline{e}$ be any affine lift and $\tilde{e}$ be an excellent lift with $\tilde{e} = \lambda * \overline{e}$ for $\lambda \in k^*$. Then, since $\texttt{Mult}(m, \tilde{e}) = \lambda^{m^2} * \texttt{Mult}(m, \overline{e})$ for $m \in \mathbb{Z}$ by Lemma 7, we have

$$\lambda^\ell = \frac{\texttt{Mult}(\ell', \overline{e})_i}{\texttt{Mult}(\ell' + 1, \overline{e})_i} \tag{16}$$

where $\ell' = \frac{\ell - 1}{2}$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$.

Let $\{e_1, e_2\}$ be a basis of $K \simeq (\mathbb{Z}/\ell\mathbb{Z})^2$ and $\tilde{e_1}, \tilde{e_2}, \widetilde{e_1 + e_2}$ be excellent lifts. Then, the set $\tilde{K} = \{\widetilde{m_1e_2 + m_2e_2} \mid 0 \le m_1, m_2 < \ell\}$ computed from $\tilde{e_1}, \tilde{e_2}, \widetilde{e_1 + e_2}$ is excellent by Theorem 2. In addition, for any affine lifts $\overline{e_1}, \overline{e_2}, \overline{e_1 + e_2}$, we write the affine lift of $m_1e_2 + m_2e_2$ computed from $\overline{e_1}, \overline{e_2}, \overline{e_1 + e_2}$ by $\overline{m_1e_2 + m_2e_2}$. If $\tilde{e_1} = \lambda_1 * \overline{e_1}, \tilde{e_2} = \lambda_2 * \overline{e_2}, \widetilde{e_1 + e_2} = \lambda_{12} * \overline{e_1 + e_2}$ for $\lambda_1, \lambda_2, \lambda_{12} \in k^*$, we have the following relational expressions. Here, $\prod \tilde{e}$ means $(\prod \tilde{e_i})_i$.

Lemma 8. *The notation is the same as above. Let $a_1, \dots, a_r$ be positive integers such that $\ell = \sum_{u=1}^r a_u^2$. Let m_1, m_2 be integers such that $0 \le m_1, m_2 < \ell$. For each $1 \le u \le r$, we divide a_um_1 and a_um_2 by ℓ, that is, $a_um_1 = t_{1,u}\ell + s_{1,u}$ and $a_um_2 = t_{2,u}\ell + s_{2,u}$ where $t_{1,u}, t_{2,u}, s_{1,u}, s_{2,u}$ are integers with $0 \le s_{1,u}, s_{2,u} < \ell$. Then, we have*

$$\begin{gathered}
\overline{m_1e_1} = \left(\lambda_1^\ell\right)^{\ell - 2m_1} * \overline{(\ell - m_1)e_1}, \qquad \overline{m_2e_2} = \left(\lambda_2^\ell\right)^{\ell - 2m_2} * \overline{(\ell - m_2)e_2} \ , \\
\overline{m_1e_1 + m_2e_2} = \left(\left(\lambda_1^\ell\right)^{\ell - 2m_1}\left(\lambda_2^\ell\right)^{\ell - 2m_2}\left(\frac{\lambda_{12}^\ell}{\lambda_1^\ell\lambda_2^\ell}\right)^{\ell - m_1 - m_2}\right) * \overline{(\ell - m_1)e_1 + (\ell - m_2)e_2} \ ,
\end{gathered} \tag{17}$$

$$\prod_{u=1}^r (\widetilde{m_1a_ue_1 + m_2a_ue_2}) = \left(\left(\lambda_1^\ell\right)^{h_1}\left(\lambda_2^\ell\right)^{h_2}\left(\frac{\lambda_{12}^\ell}{\lambda_1^\ell\lambda_2^\ell}\right)^{h_{12}}\right) * \prod_{u=1}^r \overline{s_{1,u}e_1 + s_{2,u}e_2} \tag{18}$$

where

$$h_1 := m_1^2 + \ell \sum_{u=1}^{r} t_{1,u}^2 - 2m_1 \sum_{u=1}^{r} a_u t_{1,u}, \quad h_2 := m_2^2 + \ell \sum_{u=1}^{r} t_{2,u}^2 - 2m_2 \sum_{u=1}^{r} a_u t_{2,u},$$

$$h_{12} := m_1 m_2 + \ell \sum_{u=1}^{r} t_{1,u} t_{2,u} - m_1 \sum_{u=1}^{r} a_u t_{2,u} - m_2 \sum_{u=1}^{r} a_u t_{1,u}$$

and they satisfy $0 \le h_1, h_2, h_{12} \le r(\ell - 1)$*. In addition, we have*

$$(\widetilde{m_1 e_1 + m_2 e_2})^\ell = \left(\left(\lambda_1^\ell\right)^{m_1^2} \left(\lambda_2^\ell\right)^{m_2^2} \left(\frac{\lambda_{12}^\ell}{\lambda_1^\ell \lambda_2^\ell} \right)^{m_1 m_2} \right) * (\overline{m_1 e_1 + m_2 e_2})^\ell \ . \quad (19)$$

Proof. By Lemma 7, we have

$$\widetilde{m_1 e_1 + m_2 e_2} = \left(\lambda_1^{m_1^2} \lambda_2^{m_2^2} \left(\frac{\lambda_{12}}{\lambda_1 \lambda_2} \right)^{m_1 m_2} \right) * \overline{m_1 e_1 + m_2 e_2} \ .$$

Then, by raising both sides to the ℓ^{th} power, we have (19). Now, by the excellentness, we have $\widetilde{m_1 e_1} = (\widetilde{\ell - m_1)e_1}$. In addition, by Lemma 7, we have $\widetilde{m_1 e_1} = \lambda_1^{m_1^2} * \overline{m_1 e_1}$ and $(\widetilde{\ell - m_1)e_1} = \lambda_1^{(\ell - m_1)^2} * \overline{(\ell - m_1)e_1}$. Hence, we have $\overline{m_1 e_1} = \lambda_1^{\ell^2 - 2\ell m_1} * \overline{(\ell - m_1)e_1}$. Now, $\widetilde{m_2 e_2}$ and $m_1 \widetilde{e_1 + m_2} e_2$ are similar, thus we have (17).

Similarly, by the excellentness, we have $m_1 a_u \widetilde{e_1 + m_2} a_u e_2 = s_{1,u} \widetilde{e_1 + s_{2,u}} e_2$. Applying Lemma 7 to the right-hand side, we have

$$m_1 a_u \widetilde{e_1 + m_2} a_u e_2 = \left(\lambda_1^{s_{1,u}^2} \lambda_2^{s_{2,u}^2} \left(\frac{\lambda_{12}}{\lambda_1 \lambda_2} \right)^{s_{1,u} s_{2,u}} \right) * \overline{s_{1,u} e_1 + s_{2,u} e_2} \ .$$

Finally, taking the product for $1 \le u \le r$, we can show (18). Then, we have $h_1 \ell = \sum_{u=1}^r s_{1,u}^2$ and $h_2 \ell = \sum_{u=1}^r s_{2,u}^2$ and $h_{12} \ell = \sum_{u=1}^r s_{1,u} s_{2,u}$. Since $0 \le s_{1,u}, s_{2,u} \le \ell - 1$, we have $0 \le h_1, h_2, h_{12} \le r(\ell - 1)$. □

Evaluation. Let $\tilde{K}$ be any excellent lift, $\tilde{x}$ be any affine lift of $x \in A$, and $\widetilde{x + e}$ be an excellent lift with respect to $\tilde{K}$ and $\tilde{x}$. For any affine lifts $\overline{e}$ and $\overline{x + e}$, we put $\tilde{e} = \lambda * \overline{e}$ and $\widetilde{x + e} = \mu * \overline{x + e}$ for $\lambda, \mu \in k^*$. Since $\texttt{Multadd}(\ell, \tilde{e}, \tilde{x}, \widetilde{x + e}) = (\lambda^{\ell^2} \cdot (\frac{\mu}{\lambda})^\ell) * \texttt{Multadd}(\ell, \overline{e}, \tilde{x}, \overline{x + e})$ by Lemma 7, we have

$$\left(\frac{\mu}{\lambda} \right)^\ell = \frac{\tilde{x}_i}{(\lambda^\ell)^\ell \cdot \texttt{Multadd}(\ell, \overline{e}, \tilde{x}, \overline{x + e})_i} \quad (20)$$

for $i \in (\mathbb{Z}/2\mathbb{Z})^2$.

Lemma 9. *The notation is the same as above. Let* $\lambda_1, \lambda_2, \lambda_{12}$ *be the same as Lemma 8 and* $\widetilde{x + e_1} = \mu_1 * \overline{x + e_1}$, $\widetilde{x + e_2} = \mu_2 * \overline{x + e_2}$ *for* $\mu_1, \mu_2 \in k^*$*. Moreover, let* $x + m_1 \widetilde{e_1 + m_2} e_2$ *be the set of affine lifts computed from* $\tilde{e_1}$, $\tilde{e_2}$, $\widetilde{e_1 + e_2}$, $\tilde{x}$,

$\widetilde{x+e_1}$, $\widetilde{x+e_2}$. Similarly, let $\overline{x+m_1e_1+m_2e_2}$ be the set of affine lifts computed from $\overline{e_1}, \overline{e_2}, \overline{e_1+e_2}, \tilde{x}, \overline{x+e_1}, \overline{x+e_2}$. Then, we have the following two equalities:

$$\begin{aligned}&\prod_{u=1}^{r}(a_u x + m_1 \widetilde{a_u e_1} + m_2 a_u e_2)\\ &= \left(\left(\lambda_1^{\ell}\right)^{m_1^2} \left(\lambda_2^{\ell}\right)^{m_2^2} \left(\frac{\lambda_{12}^{\ell}}{\lambda_1^{\ell}\lambda_2^{\ell}}\right)^{m_1 m_2} \left(\frac{\mu_1^{\ell}}{\lambda_1^{\ell}}\right)^{m_1} \left(\frac{\mu_2^{\ell}}{\lambda_2^{\ell}}\right)^{m_2} \right) * \prod_{u=1}^{r} (\overline{a_u x + m_1 a_u e_1 + m_2 a_u e_2}) \ .\end{aligned} \tag{21}$$

$$\begin{aligned}&(x + m_1\widetilde{e_1 + m_2 e_2})^{\ell}\\ &= \left(\left(\lambda_1^{\ell}\right)^{m_1^2} \left(\lambda_2^{\ell}\right)^{m_2^2} \left(\frac{\lambda_{12}^{\ell}}{\lambda_1^{\ell}\lambda_2^{\ell}}\right)^{m_1 m_2} \left(\frac{\mu_1^{\ell}}{\lambda_1^{\ell}}\right)^{m_1} \left(\frac{\mu_2^{\ell}}{\lambda_2^{\ell}}\right)^{m_2} \right) * (\overline{x + m_1 e_1 + m_2 e_2})^{\ell} \ .\end{aligned} \tag{22}$$

Proof. (21) is obtained by Lemma 7. By applying $\ell = 1^2 + \cdots + 1^2$ to (21), we have (22). □

3.4 Explicit algorithms of the Lubicz–Robert formula

In this subsection, we propose explicit algorithms computing the theta-null point of the codomain B and computing the theta coordinate of the image of $x \in A$ under f based on the Lubicz–Robert formula (Theorem 3).

In the remainder of this paper, logarithm always has base 2. Recall that M (resp. S) means the cost of a multiplication (resp. square) operation on k. In addition, for any positive integer N and $\lambda \in k$, $\mathsf{P}(N)$ means the cost of computing λ^N. Hence, $\mathsf{P}(N) = O(\log(N))\mathsf{M}$. Moreover, for positive integers $N_1, \ldots, N_m$ and $\lambda \in k$, $\mathsf{P}_{\text{set}}(\{N_1, \ldots, N_m\})$ means the cost of computing the N_i^{th} powers $\lambda^{N_1}, \ldots, \lambda^{N_m}$ all. Thus, if we compute them one by one, the cost $\mathsf{P}_{\text{set}}(\{N_1, \ldots, N_m\})$ is $\mathsf{P}(N_1) + \cdots + \mathsf{P}(N_m)$. On the other hand, if we compute them by calculating by multiplying λ $(\max\{N_1, \ldots, N_m\} - 1)$ times; $\lambda^2, \lambda^3, \lambda^4, \ldots, \lambda^{\max\{N_1,\ldots,N_m\}}$, the cost $\mathsf{P}_{\text{set}}(\{N_1, \ldots, N_m\})$ is $(\max\{N_1, \ldots, N_m\} - 1)\mathsf{M}$. Although these methods may not be the most efficient, we consider P_{set} to be the minimum of the above costs.

Codomain Here, we calculate the theta-null point of the codomain using the Lubicz–Robert formula. As noted in Section 3.1, we give two algorithms `CodSq`, `CodOne` for computation of the theta-null point of a codomain.

`CodSq` is based on (10) using $\ell = \sum_{u=1}^{r} a_u^2$ and `CodOne` is based on (12) using $\ell = 1^2 + \cdots + 1^2$.

By contrast, in `CodSq`, we use equalities $m_1 a_u e_1 + m_2 a_u e_2 = s_{1,u} e_1 + s_{2,u} e_2$ if $m_1 a_u \equiv s_{1,u} \pmod{\ell}$ and $m_2 a_u \equiv s_{2,u} \pmod{\ell}$ as follows. The affine lift $\overline{s_1 e_1 + s_2 e_2}$ and $\overline{m_1 a_u e_1 + m_2 a_u e_2}$ correspond to the same projective coordinate but are not equal as affine lifts. Thus, by multiplying by an appropriate constant, we can compute $\overline{m_1 a_u e_1 + m_2 a_u e_2}$ from $\overline{s_{1,u} e_1 + s_{2,u} e_2}$. By this manner, we avoid computing linear combinations many times.

We summarize these two options `CodSq` and `CodOne` in Table 2.

	Formula	Normalization	Algorithm	Complexity
CodSq	(10)	(17) and (18)	1	$O_r(\ell^2)$
CodOne	(12)	(19)	2	$O(\ell^2 \log(\ell))$

Table 2. Two calculation methods of the *codomain*

Algorithm 1 CodSq

Input: Affine lifts $\overline{e_1}, \overline{e_2}, \overline{e_1+e_2}$ of the basis $\{e_1, e_2\}$ of the kernel.
Output: A projective theta-null point of the codomain.

1: Compute $\overline{m_1e_1+m_2e_2}$ for $(m_1,m_2) \in H_\ell$. $(\triangleright) C_{hlc}(\ell)$
2: Compute (α_1, d_1) such that $\frac{\alpha_1}{d_1} = \lambda_1^\ell$ using (16). $(\triangleright) O(1)$M
3: Compute (α_2, d_2) such that $\frac{\alpha_2}{d_2} = \lambda_2^\ell$ using (16). $(\triangleright) O(1)$M
4: Compute (α_{12}, d_{12}) such that $\frac{\alpha_{12}}{d_{12}} = \lambda_{12}^\ell$ using (16). $(\triangleright) O(1)$M
5: $(\beta, d') := (\alpha_{12} \cdot d_1 \cdot d_2, d_{12} \cdot \alpha_1 \cdot \alpha_2)$ where $\frac{\beta}{d'} = \frac{\lambda_{12}^\ell}{\lambda_1^\ell \lambda_2^\ell}$. $(\triangleright)$4M
6: $((\alpha_1, d), (\alpha_2, d), (\beta, d)) :=$ `Commondenom`$((\alpha_1, d_1), (\alpha_2, d_2), (\beta, d'))$.$(\triangleright) C_{cd}(3,1) = 7$M
7: Take a representation $\ell = \sum_{u=1}^r a_u^2$ based on Lemma 12 in Section 3.5.
8: Calculate $\alpha_1^s, \alpha_2^s, \beta^s, d^s$ for needed s in lines 9, 13. $(\triangleright) 6r\ell$M
9: Extend $\overline{m_1e_1+m_2e_2}$ from H_ℓ to $0 \le m_1, m_2 < \ell$ using (17). $(\triangleright)(3\ell^2 - 4\ell + 3)$M
10: $(\theta_i'(f(0)), d_i)_i := (\prod_{u=1}^r$ `Num`$((\tilde{\theta}_i(0))_i, a_u i), 1)_i$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$. $(\triangleright) 4(r-1)$M
11: **for** $(m_1, m_2) \in H_\ell$ **do**
12: $\quad a_u m_1 = \ell t_{1,u} + s_{1,u}, a_u m_2 = \ell t_{2,u} + s_{2,u}$ for $1 \le u \le r$.
13: $\quad (c_n, c_d) := (\alpha_1^{h_1} \cdot \alpha_2^{h_2} \cdot \beta^{h_{12}}, d^{h_1+h_2+h_{12}})$ where h_1, h_2, h_{12} are of (18). $(\triangleright)$2M
14: $\quad t_d := c_d \cdot \prod_{u=1}^r$ `Den`$(\overline{s_{1,u}e_1 + s_{2,u}e_2})$. $(\triangleright) r$M
15: $\quad$ **for** $i \in (\mathbb{Z}/2\mathbb{Z})^2$ **do**
16: $\quad\quad t_n := 2c_n \cdot \prod_{u=1}^r$ `Num`$(\overline{s_{1,u}e_1 + s_{2,u}e_2}, a_u i)$. $(\triangleright) r$M
17: $\quad\quad (\theta_i'(f(0)), d_i) := (\theta_i'(f(0)) \cdot t_d + d_i \cdot t_n, d_i \cdot t_d)$. $(\triangleright)$3M
18: $\quad$ **end for**
19: **end for**
20: $(\theta_i'(f(0)))_{i \in (\mathbb{Z}/2\mathbb{Z})^2} :=$ `Projcommondenom`$((\theta_i'(f(0)), d_i)_{i \in (\mathbb{Z}/2\mathbb{Z})^2})$.
$(\triangleright) C_{pcd}(4,1) = 10$M
21: **return** $(\theta_i(f(0)), 1)_i$.

In any case, since (in level 2) a projective theta coordinate of an element $e \in K$ is the same as that of the inverse element $-e$, we can reduce the complexity to half. To explain that, we use a subset $H_\ell \subseteq \mathbb{Z}^2$ of (15) in Section 3.2. Then, for a basis $\{e_1, e_2\}$ of K, we have $\{m_1e_1 + m_2e_2 \in K \mid (m_1, m_2) \in H_\ell\} \sqcup \{-(m_1e_1 + m_2e_2) \in K \mid (m_1, m_2) \in H_\ell\} = K \setminus \{0\}$.

For CodSq, note that affine lifts of $m_1e_1 + m_2e_2$ for $(m_1, m_2) \in H_\ell$ are not sufficient since we may not have $(s_{1,u}, s_{2,u}) \in H_\ell$. Thus, we have to extend affine lifts $\overline{m_1e_1+m_2e_2}$ for $(m_1, m_2) \in H_\ell$ to $0 \le m_1, m_2 < \ell$ using (17) of Lemma 8. Especially, it is not clear whether CodSq is more efficient.

Explicit algorithms of CodSq and CodOne are Algorithm 1 and 2, respectively. In Algorithm 2, $Excl(m_1, m_2)$ denotes the ℓ^{th} power of the excellent affine lift of $m_1e_1 + m_2e_2$ holding as 5 elements as stated at the beginning of Section 3.2.

Algorithm 2 `CodOne`

Input: Affine lifts $\overline{e_1}, \overline{e_2}, \overline{e_1 + e_2}$ of the basis $\{e_1, e_2\}$ of the kernel.
Output: A projective theta-null point of the codomain.

1: Compute $\overline{m_1e_1 + m_2e_2}$ for $(m_1, m_2) \in H_\ell$. $(\triangleright) C_{hlc}(\ell)$
2: Here is the same as lines 2 to 6 of Algorithm 1.
3: Calculate $\alpha_1^{m_1^2}$ for $0 \le m_1 < \ell$ $(\triangleright)\mathsf{P}_{\text{set}}(\{m_1^2 \mid 0 \le m_1 < \ell\})$
4: Calculate $\alpha_2^{m_2^2}$ for $0 \le m_2 < \ell$ $(\triangleright)\mathsf{P}_{\text{set}}(\{m_2^2 \mid 0 \le m_2 < \ell\})$
5: Calculate $\beta^{m_1m_2}$ for $(m_1, m_2) \in H_\ell$ $(\triangleright)\mathsf{P}_{\text{set}}(\{m_1m_2 \mid (m_1, m_2) \in H_\ell\})$
6: Calculate $d^{m_1^2+m_2^2+m_1m_2}$ for $(m_1, m_2) \in H_\ell$. $(\triangleright)\mathsf{P}_{\text{set}}(\{m_1^2 + m_2^2 + m_1m_2 \mid (m_1, m_2) \in H_\ell\})$
7: **for** $(m_1, m_2) \in H_\ell$ **do**
8: $\quad (c_n, c_d) := (\alpha_1^{m_1^2} \cdot \alpha_2^{m_2^2} \cdot \beta^{m_1m_2}, d^{m_1^2+m_2^2+m_1m_2})$. $(\triangleright)2\mathsf{M}$
9: $\quad$ Compute $\texttt{Num}(\overline{m_1e_1 + m_2e_2}, i)^\ell$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$. $(\triangleright)4\mathsf{P}(\ell)$
10: $\quad$ Compute $\texttt{Den}(\overline{m_1e_1 + m_2e_2})^\ell$. $(\triangleright)\mathsf{P}(\ell)$
11: $\quad \texttt{Num}(Excl(m_1, m_2), i) := c_n \cdot \texttt{Num}(\overline{m_1e_1 + m_2e_2}, i)^\ell$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$. $(\triangleright)4\mathsf{M}$
12: $\quad \texttt{Den}(Excl(m_1, m_2)) := c_d \cdot \texttt{Den}(\overline{m_1e_1 + m_2e_2})^\ell$. $(\triangleright)1\mathsf{M}$
13: **end for**
14: Calculate $\texttt{Num}(Excl(0, 0)) := \texttt{Num}(\theta_i(0), i)^\ell$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$. $(\triangleright)4\mathsf{P}(\ell)$
15: Calculate $\texttt{Den}(Excl(0, 0)) := 1$.
16: $(Excl(m_1, m_2))_{(m_1,m_2)} := \texttt{Projcommondenom}((Excl(m_1, m_2))_{(m_1,m_2)})$ w.r.t. $i \in (\mathbb{Z}/2\mathbb{Z})^2$ and $(m_1, m_2) \in H_\ell \sqcup \{(0, 0)\}$. $(\triangleright)C_{pcd}(\frac{\ell^2+1}{2}, 4) = (\frac{7}{2}\ell^2 + O(1))\mathsf{M}$
17: $\theta'_i(f(0)) := Excl(m_1, m_2)_i$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$.
18: **for** $(m_1, m_2) \in H_\ell$ **do**
19: $\quad$ **for** $i \in (\mathbb{Z}/2\mathbb{Z})^2$ **do**
20: $\quad\quad \theta'_i(f(0)) := \theta'_i(f(0)) + 2Excl(m_1, m_2)_i$.
21: $\quad$ **end for**
22: **end for**
23: **return** $(\theta'_i(f(0)), 1)_i$.

Remark 9. (Alg. 1, lines 2-4) For `CodSq` and `CodOne`, since we compute $\overline{m_1e_1 + m_2e_2}$ for $(m_1, m_2) \in H_\ell$ first, we have $\overline{\ell' e_1}$ and $\overline{\ell' e_2}$ and $\overline{\ell' e_1 + \ell' e_2}$. Thus, when computing $\lambda_1^\ell, \lambda_2^\ell, \lambda_{12}^\ell$ using (16), we only need $\overline{(\ell'+1)e_1}$ and $\overline{(\ell'+1)e_2}$ and $\overline{(\ell'+1)e_1 + (\ell'+1)e_2}$. They are computed from $\overline{m_1e_1 + m_2e_2}$ for $(m_1, m_2) \in H_\ell$ by Differential Addition. Moreover, we only need the i^{th}-coordinate for one $i \in (\mathbb{Z}/2\mathbb{Z})^2$. Especially, we can compute $\lambda_1^\ell, \lambda_2^\ell, \lambda_{12}^\ell$ in $O(1)\mathsf{M}$.

Here, we give complexities of `CodSq` and `CodOne`.

1. (Alg. 2, lines 3-4) When calculating $\alpha^{m_1^2}$ for $0 \le m_1 < \ell$, we compute individually. Thus, we can approximate $\mathsf{P}_{\text{set}}(\{m_1^2 \mid 0 \le m_1 < \ell\}) = O(\ell \log(\ell))\mathsf{M}$. The case for $\mathsf{P}_{\text{set}}(\{m_2^2 \mid 0 \le m_2 < \ell\})$ is similar.
2. (Alg. 2, lines 5-6) When we calculate $\beta^{m_1m_2}$ for $(m_1, m_2) \in H_\ell$, since $\max\{m_1m_2 \mid (m_1, m_2) \in H_\ell\} = \frac{\ell^2-1}{4}$, we calculate $\beta^2, \beta^3, \beta^4, \cdots, \beta^{\frac{\ell^2-1}{4}}$ straightforwardly. Thus, we have $\mathsf{P}_{\text{set}}(\{m_1m_2 \mid (m_1, m_2) \in H_\ell\}) = \frac{\ell^2-5}{4}\mathsf{M}$. Similarly, since $\max\{m_1^2 + m_2^2 + m_1m_2 \mid (m_1, m_2) \in H_\ell\} = \ell^2 - \ell + 1$, we have $\mathsf{P}_{\text{set}}(\{m_1^2 + m_2^2 + m_1m_2 \mid (m_1, m_2) \in H_\ell\}) = (\ell^2 - \ell)\mathsf{M}$. This computation is

not the dominant part of Algorithm 2, however there is likely a more efficient method to compute them.

Lemma 10. *From Algorithms 1 and 2, the costs are as follows:*

`CodSq`: $\frac{(5r+53)(\ell^2-1)}{2}\mathsf{M} + \frac{5(\ell^2-1)}{2}\mathsf{S} + 6r\ell\mathsf{M} + O(\ell)\mathsf{M}$.

`CodOne`: $\frac{99(\ell^2-1)}{4}\mathsf{M} + \frac{5(\ell^2-1)}{2}\mathsf{P}(\ell) + \frac{5(\ell^2-1)}{2}\mathsf{S} + O(\ell\log(\ell))\mathsf{M}$.

Especially, the complexity of `CodSq` is $O_r(\ell^2)\mathsf{M}$ and that of `CodOne` is $O(\ell^2\log(\ell))\mathsf{M}$ since $\mathsf{P}(\ell) = O(\log(\ell))$. Concrete counts of operations for each ℓ are written in Section 4.

Evaluation. We provide similar algorithms for general points, that is, for $x \in A$, we compute the theta coordinate of $f(x) \in B$ from some theta coordinates of $e_1, e_2, e_1+e_2, x, x+e_1, x+e_2$.

Note that we need to compute all linear combinations $x + m_1e_1 + m_2e_2$ for $0 \le m_1, m_2 < \ell$, not only for $(m_1, m_2) \in H_\ell$.

In the same notations as those for the codomain, let $\overline{e_1}, \overline{e_2}$ and $\overline{e_1+e_2}$ be affine lifts of e_1, e_2 and e_1+e_2 for a basis $\{e_1, e_2\}$ of K. For given any affine lifts $\tilde{x}, \overline{x+e_1}$, and $\overline{x+e_2}$ of $x, x+e_1$, and $x+e_2$, we give a projective theta coordinate of the image $f(x)$.

Now, we give two concrete algorithms `EvalSq` and `EvalOne`. `EvalSq` is based on (11) and `EvalOne` is based on (13). We summarize them in Table 3.

	Formula	Normalization	Algorithm	Complexity
`EvalSq`	(11)	(21)	3	$O_r(\ell^2)$
`EvalOne`	(13)	(22)	4	$O(\ell^2\log(\ell))$

Table 3. Two calculation methods of the *evaluation*

In advance, we calculate $(\lambda_1^\ell)^{m_1^2}(\lambda_2^\ell)^{m_2^2}(\frac{\lambda_{12}^\ell}{\lambda_1^\ell\lambda_2^\ell})^{m_1m_2}$ for all $0 \le m_1, m_2 < \ell$ and $\lambda_1^{\ell^2}, \lambda_2^{\ell^2}$ which are independent on x.

Their explicit algorithms of `EvalSq`,`EvalOne` are Algorithms 3, 4 respectively.

Remark 10. For `EvalOne`, we can use similar optimization as that in Remark 9. Concretely, since we first compute $\overline{x+m_1e_1+m_2e_2}$ for $0 \le m_1, m_2 < \ell$ we have $\overline{x+(\ell-1)e_1}$ and $\overline{x+(\ell-1)e_2}$. Hence, when we compute $(\frac{\mu_1}{\lambda_1})^\ell$ and $(\frac{\mu_2}{\lambda_2})^\ell$, we only need $\overline{x+\ell e_1}$ and $\overline{x+\ell e_2}$ which are computed from $\overline{x+m_1e_1+m_2e_2}$ by Differential Addition. Moreover, we only need the i^{th}-coordinate for one $i \in (\mathbb{Z}/2\mathbb{Z})^2$. Especially, we can compute $(\frac{\mu_1}{\lambda_1})^\ell$ and $(\frac{\mu_2}{\lambda_2})^\ell$ in $O(1)\mathsf{M}$. This is also valid for `EvalSq` if $a_u = 1$ for some u.

Algorithm 3 `EvalSq`

Input: Affine lifts $\overline{e_1}, \overline{e_2}, \overline{e_1+e_2}, \tilde{x}, \overline{x+e_1}, \overline{x+e_2}$ and $(\lambda_1^\ell)^{m_1^2}(\lambda_2^\ell)^{m_2^2}(\frac{\lambda_{12}^\ell}{\lambda_1^\ell\lambda_2^\ell})^{m_1m_2}$.

Output: A projective theta coordinate of $f(x)$.

```
Take a representation ℓ = Σ_{u=1}^r a_u^2.
Compute \overline{a_u e_1}, \overline{a_u e_2}, \overline{a_u(e_1+e_2)}, \overline{a_u x}, \overline{a_u(e_1+x)}, \overline{a_u(e_2+x)} for 1 ≤ u ≤ r.
                                                                    (▷)O(r log(ℓ))M
Compute \overline{a_u x + m_1 a_u e_1 + m_2 a_u e_2} for 0 ≤ m_1, m_2 < ℓ and for 1 ≤ u ≤ r.
Here, r' := #{a_1, ..., a_r}.                                       (▷)r' C_{lc+}(ℓ)
Compute (γ_1, d_1) such that γ_1/d_1 = (μ_1/λ_1)^ℓ using (20).      (▷)O(log(ℓ))M
Compute (γ_2, d_2) such that γ_2/d_2 = (μ_2/λ_2)^ℓ using (20).      (▷)O(log(ℓ))M
((γ_1, d), (γ_2, d)) := Commondenom((γ_1, d_1), (γ_2, d_2)).        (▷)C_{cd}(2,1) = 3M
Calculate numerators of γ_1^m, γ_2^m for 0 ≤ m ≤ ℓ − 1.             (▷)2(ℓ − 2)M
Calculate d^m for 0 ≤ m ≤ 2(ℓ − 1).                                 (▷)(2ℓ − 3)M
Take a representation ℓ = Σ_{u=1}^r a_u^2 based on Lemma 12 in Section 3.5.
(θ'_i(f(x)), d_i) := (0, 1) for i ∈ (Z/2Z)^2.
for 0 ≤ m_1, m_2 < ℓ do
    c_n := Num((λ_1^ℓ)^{m_1^2}(λ_2^ℓ)^{m_2^2}(λ_12^ℓ/(λ_1^ℓ λ_2^ℓ))^{m_1 m_2}) · γ_1^{m_1} · γ_2^{m_2}   (▷)2M
    c_d := Den((λ_1^ℓ)^{m_1^2}(λ_2^ℓ)^{m_2^2}(λ_12^ℓ/(λ_1^ℓ λ_2^ℓ))^{m_1 m_2}) · d^{m_1+m_2}.      (▷)1M
    t_d := c_d · Π_{u=1}^r Den(\overline{a_u x + m_1 a_u e_1 + m_2 a_u e_2}).                        (▷)rM
    for i ∈ (Z/2Z)^2 do
        t_n := c_n · Π_{u=1}^r Num(\overline{a_u x + m_1 a_u e_1 + m_2 a_u e_2}, a_u i).              (▷)rM
        (θ'_i(f(x)), d_i) := (θ'_i(f(x)) · t_d + d_i · t_n, d_i · t_d).                              (▷)3M
    end for
end for
(θ'_i(f(x)))_{i∈(Z/2Z)^2} := Projcommondenom((θ'_i(f(x)), d_i)_{i∈(Z/2Z)^2}).
                                                                    (▷)C_{pcd}(4,1) = 10M
return (θ'_i(f(x)), 1)_i.
```

Lemma 11. *By Algorithms 3 and 4, we give concrete costs of* `EvalSq` *and* `EvalOne` *as follows. Here,* $r' := \#\{a_1, \dots, a_r\} \le r$ *for a representation* $\ell = \sum_{u=1}^r a_u^2$.

`EvalSq`: $(5r + 33r' + 15)\ell^2\mathsf{M} + 5r'\ell^2\mathsf{S} + O(\ell)\mathsf{M} + O_r(\log(\ell))\mathsf{M}$.

`EvalOne`: $51\ell^2\mathsf{M} + 5\ell^2\mathsf{P}(\ell) + 5\ell^2\mathsf{S} + O(\ell)\mathsf{M}$.

In particular, the complexity of `EvalSq` is $O_r(\ell^2)$ and that of `EvalOne` is $O(\ell^2\log(\ell))$. The concrete counts of operations for each ℓ are presented in Section 4.

3.5 Representation $\ell = \sum_{u=1}^r a_u^2$

In this subsection, for simplicity, we only consider odd prime numbers ℓ. When using Algorithm `CodSq` or `EvalSq`, we take a representation of ℓ by the sum of squares of positive integers: $\ell = \sum_{u=1}^r a_u^2$. If $\ell = 3$, such a representation is only $3 = 1^2 + 1^2 + 1^2$. Otherwise, that is $\ell \ge 5$, what kind of representation is efficient

Algorithm 4 `EvalOne`

Input: Affine lifts $\overline{e_1}, \overline{e_2}, \overline{e_1+e_2}$, $\tilde{x}, \overline{x+e_1}, \overline{x+e_2}$ and $(\lambda_1^\ell)^{m_1^2}(\lambda_2^\ell)^{m_2^2}(\frac{\lambda_{12}^\ell}{\lambda_1^\ell\lambda_2^\ell})^{m_1m_2}$ and $(\theta_i(x))_i$.
Output: A projective theta coordinate of $f(x)$.

Compute $\overline{x+m_1e_1+m_2e_2}$ for $0 \le m_1, m_2 < \ell$. $(\triangleright) C_{lc+}(\ell)$
Here is the same as lines 4 to 8 of Algorithm 3.
for $0 \le m_1, m_2 < \ell$ **do**
    $c_n := \texttt{Num}((\lambda_1^\ell)^{m_1^2}(\lambda_2^\ell)^{m_2^2}(\frac{\lambda_{12}^\ell}{\lambda_1^\ell\lambda_2^\ell})^{m_1m_2}) \cdot \gamma_1^{m_1} \cdot \gamma_2^{m_2}$ $(\triangleright)2\mathsf{M}$
    $c_d := \texttt{Den}((\lambda_1^\ell)^{m_1^2}(\lambda_2^\ell)^{m_2^2}(\frac{\lambda_{12}^\ell}{\lambda_1^\ell\lambda_2^\ell})^{m_1m_2}) \cdot d^{m_1+m_2}$. $(\triangleright)1\mathsf{M}$
    Compute $\texttt{Num}(\overline{x+m_1e_1+m_2e_2}, i)^\ell$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$. $(\triangleright)4\mathsf{P}(\ell)$
    Compute $\texttt{Den}(\overline{x+m_1e_1+m_2e_2})^\ell$. $(\triangleright)\mathsf{P}(\ell)$
    $\texttt{Num}(Excl(m_1,m_2), i) := c_n \cdot \texttt{Num}(\overline{x+m_1e_1+m_2e_2}, i)^\ell$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$. $(\triangleright)4\mathsf{M}$
    $\texttt{Den}(Excl(m_1,m_2), i) := c_d \cdot \texttt{Den}(\overline{x+m_1e_1+m_2e_2}, i)^\ell$. $(\triangleright)4\mathsf{M}$
end for
$(Excl(m_1,m_2))_{(m_1,m_2)} := \texttt{Projcommondenom}((Excl(m_1,m_2))_{(m_1,m_2)})$ w.r.t. $i \in (\mathbb{Z}/2\mathbb{Z})^2$ and $0 \le m_1, m_2 < \ell$. $(\triangleright) C_{pcd}(\ell^2, 4) = (7\ell^2 + O(1))\mathsf{M}$
$\theta_i'(f(x)) := 0$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$.
for $0 \le m_1, m_2 < \ell$ **do**
    for $i \in (\mathbb{Z}/2\mathbb{Z})^2$ **do**
        $\theta_i'(f(x)) := \theta_i'(f(x)) + Excl(m_1,m_2)_i$.
    end for
end for
return $(\theta_i'(f(x)), 1)_i$.

for each algorithm? In the following, for $\ell \ge 5$, we exclude the representation $\ell = 1^2 + \cdots + 1^2$ since the case is just `CodOne` and `EvalOne`.

For `CodSq`, by the asymptotic complexity in Lemma 10, we should take a representation such that r is minimized. By contrast, for `EvalSq`, by their asymptotic complexities in Lemmas 10, 11, we should take a representation such that $(5r + 33r')\mathsf{M} + 5r'\mathsf{S}$ is minimized, where $r' := \#\{a_1, \ldots, a_r\} (\ge 2)$.

Lemma 12. *Let ℓ be an odd prime number such that $\ell \ge 5$. For each of* `CodSq` *and* `EvalSq`*, and for each ℓ, we should take a representation $\ell = \sum_{u=1}^r a_u^2$ (instead of $\ell = 1^2 + \cdots + 1^2$) as follows:*

1. *If $\ell \not\equiv -1 \pmod{24}$, the minimum value r_{min} of r for each ℓ is as follows. When $\ell \equiv 1 \pmod 4$, $r_{min} = 2$, when $\ell \equiv 3 \pmod 8$, $r_{min} = 3$, and when $\ell \equiv 7 \pmod{24}$, $r_{min} = 4$. Moreover, there exists a representation satisfying $r = r_{min}$ and $r' = 2$. Thus, for any* `CodSq` *and* `EvalSq`*, we should take any such representation with $r = r_{min}$ and $r' = 2$.*
2. *If $\ell \equiv -1 \pmod{24}$, the minimum value r_{min} of r is 4. Thus, for* `CodSq`*, we take a representation with $r = 4$. For* `EvalSq`*, for each $\ell < 200$, under the*

assumption $\mathsf{M} : \mathsf{S} = 3 : 2$[‡], *we should take a representation in Table 4 which minimizes* $(5r + 33r')\mathsf{M} + 5r'\mathsf{S}$.

ℓ	$\sum_{u=1}^{r} a_r^2$	r	r'
23	$5 \cdot 1^2 + 2 \cdot 3^2$	7	2
47	$2 \cdot 1^2 + 5 \cdot 3^2$	5	2
71	$2^2 + 2 \cdot 3^2 + 7^2$	4	3
167	$5 \cdot 1^2 + 2 \cdot 9^2$	7	2
191	$4^2 + 7 \cdot 5^2$	8	2

Table 4. The most efficient representation for ℓ such that $\ell \equiv -1 \pmod{24}$ for `EvalSq`. (for 2 of Lemma 12)

Proof. 1. By Fermat's theorem on sums of two squares, there exists a representation such that $r = 2$, if and only if $\ell \equiv 1 \pmod 4$. In this case, clearly, $r' = 2$. Next, we consider the other case, $\ell \equiv 3 \pmod 4$. By Legendre's three-square theorem, there exists a representation such that $r = 3$, if and only if $\ell \equiv 3 \pmod 8$. In addition, in this case, it is known that there exists a representation such that $r = 3$ and $r' = 2$ [16, Section 1(1.1)]. When $\ell \equiv 7 \pmod 8$, by Lagrange's four-square theorem, there exists a representation such that $r = 4$. Then, there exists a representation such that $r = 4, r' = 2$ if and only if $\ell \equiv 1 \pmod 3$ [16, Section 1(1.1)]. This condition is equivalent to $\ell \equiv 7 \pmod{24}$. In any case, since $r' = 2$, r and $(5r + 33r')\mathsf{M} + 5r'\mathsf{S}$ are minimized.

2. By Lagrange's four-square theorem, we have $r_{min} = 4$. Now, $(5r + 33r')\mathsf{M} + 5r'\mathsf{S}$ is minimized if and only if $(5r+33r') \cdot 3 + 5r' \cdot 2 = 15r + 109r'$ is minimized. For each $\ell < 200$, by comparing $15r + 109r'$ for all representations, we have the result of Table 4. □

Table 5 summarizes Lemma 12.

ℓ (≥ 5)	r	r'	example
$\ell \equiv 1 \pmod 4$	2	2	$5 = 1^2 + 2^2$
$\ell \equiv 3 \pmod 8$	3	2	$11 = 1^2 + 1^2 + 3^2$
$\ell \equiv 7 \pmod{24}$	4	2	$7 = 1^2 + 1^2 + 1^2 + 2^2$
$\ell \equiv -1 \pmod{24}$	$r \geq 4$	–	See Table 4

Table 5. An efficient representation $\ell = \sum_{u=1}^{r} a_u^2$ where $r' = \#\{a_1, \ldots, a_r\}$. (for Lemma 12)

‡ The reason considering for the ratio $\mathsf{M} : \mathsf{S} = 3 : 2$ will be explained at the beginning of Section 4.

ℓ	CodSq	CodOne	EvalSq	EvalOne
	$O_r(\ell^2)$	$O(\ell^2 \log \ell)$	$O_r(\ell^2)$	$O(\ell^2 \log \ell)$
3	1071	771	2118	1823
5	2711	2452	8270	5164
7	6740	5282	18034	10619
11	14924	13876	41573	27053
13	18579	19466	54745	37749
17	31829	32740	91496	63336
19	44376	43676	121861	83431
23	71692	68158	228845	128574
29	92627	108580	265217	204366
31	129896	131354	347335	245053
37	150807	173612	428398	327161
41	185105	213334	526691	401713
43	226500	248600	623287	464048
47	297904	313726	952613	580903
53	309335	378028	878633	704968
59	426020	494794	1169783	915391
61	409683	528968	1161502	978503
67	549240	593390	1509055	1108635
71	678412	704272	2363562	1305458
73	586713	704588	1663126	1316091
79	839528	919016	2244289	1691121
83	842600	962854	2311751	1784042
89	871961	1107298	2469014	2051306
97	1035741	1244930	2932321	2323755
101	1122983	1426324	3176858	2641778
103	1426148	1563008	3812926	2874753
107	1399436	1686874	3840095	3102377
109	1307811	1750586	3701596	3219441
113	1405469	1785742	3976406	3306842
127	2166776	2618972	5796664	4757679
131	2096960	2357542	5754983	4375653
137	2065769	2578540	5844425	4785669
139	2360796	2799332	6479773	5158276
149	2443367	3216862	6910958	5927196
151	3061844	3474890	8190025	6361003
157	2712747	3756698	7672825	6876571
163	3245712	3850136	8903431	7093396
167	3293819	4250704	12022895	7780491
173	3744484	4561804	9315983	8349627
179	3913640	4883890	10739111	8938851
181	3605295	4993700	10194670	9139723
191	4896904	6108136	16601459	11053113
193	4099125	5119244	11591572	9497845
197	4270919	5624758	12075206	10361340
199	5315588	6036608	14222314	11048011

Table 6. Values of $3m + 2s$ where $m\mathsf{M} + s\mathsf{S}$ is the count of operations of (ℓ, ℓ)-isogeny. (for Section 4)

4 Counting the Number of Operations

In this section, for each odd prime number ℓ, we count the number of operations on k of algorithms `CodSq`, `CodOne`, `EvalSq`, and `EvalOne` of Section 3.4. Here, we consider that the base field is $\mathbb{F}_{p^2}$ for the sake of application to isogeny-based cryptography. Note that characteristic p does not affect the number of operations. Here, a multiplication on $\mathbb{F}_{p^2}$ requires three times multiplications on $\mathbb{F}_p$, and a square on $\mathbb{F}_{p^2}$ requires two times multiplications on $\mathbb{F}_p$. Thus, we consider the cost as $\mathsf{M} : \mathsf{S} = 3 : 2$ on $\mathbb{F}_{p^2}$, and express the total cost of $m\mathsf{M} + s\mathsf{S}$ using the integer $3m+2s$ to facilitate algorithm cost comparisons. Table 6 shows the values of $3m + 2s$ for each algorithm and for each ℓ. The underlined values in red font are the minimum values for each ℓ.

Codomain For $3 \leq \ell \leq 11$ and $\ell = 19, 23$, `CodOne` is the most efficient, and for $\ell = 13, 17$ and $\ell \geq 29$, `CodSq` is the most efficient. Indeed, the asymptotic complexity of `CodOne` is $O(\ell^2 \log(\ell))$, but that of `CodSq` is $O_r(\ell^2)$. Here, O_r denotes the big O notation under the assumption that $r = O(1)$.

The cost of `CodSq` depends on r which is determined by $\ell \pmod 8$.

The cost of `CodOne` depends on the Hamming weight of ℓ, since we calculate the ℓ^{th} power several times in the algorithm. In fact, for example, the cost of `CodOne` is large when $\ell = 127 = (1111111)_2$ and $\ell = 191 = (10111111)_2$.

Evaluation For $3 \leq \ell < 200$, `EvalOne` is more efficient, although the asymptotic complexity of `EvalOne` is $O(\ell^2 \log(\ell))$ and that of `EvalSq` is $O_r(\ell^2)$. For sufficiently large ℓ, `EvalSq` would be more efficient. The smallest ℓ that `EvalSq` is more efficient is 509.

As for the codomain, the cost of `EvalSq` depends on r and r' which are determined by $\ell \pmod{24}$. See Table 5 and Table 4. The cost of `EvalOne` depends on the Hamming weight of ℓ.

5 Application to Attack on B-SIDH

In this section, we implement SIDH attack on key exchange protocol B-SIDH. In the attack, we calculate (ℓ, ℓ)-isogenies between Kummer surfaces. Then, we will use the results in Sections 3 and 4.

5.1 SIDH (B-SIDH) attacks

In this subsection, we explain SIDH (B-SIDH) attacks briefly.

B-SIDH is key exchange protocol given by Costello [14], which is based on the same problem as SIDH [21]. However, by using quadratic twist of elliptic curve we can use smaller characteristic than one of SIDH.

The security of both SIDH and B-SIDH depends on the hardness of the *supersingular isogeny with torsion problem* below. Here, p is a prime number and k is a finite field of characteristic p.

Problem 1 (Supersingular Isogeny with Torsion). Let N_A and N_B be coprime integers, E_0/k and E_B/k be elliptic curves, $\varphi_B : E_0 \to E_B$ be N_B-isogeny, and $\{P_A, Q_A\}$ be a basis of $E_0[N_A]$.

Then, given $N_A, N_B, E_0, E_B, P_A, Q_A, \varphi_B(P_A), \varphi_B(P_B)$, construct φ_B.

SIDH attacks However, in 2022, Castryck, Decru [7] and Maino, Martindale, Panny, Pope, Wesolowski [30] and Robert [38] gave a polynomial-time attack on SIDH by solving the above problem. Thus, as noted in [7], the security of B-SIDH was also broken.

In the attack, the following lemma based on a criterion by Kani [23] is essential. Here, we consider the case of dimension one, even though it holds for a general dimension, see [38, Lemma 3.4].

Lemma 13 ([38, Lemma 3.4]). *Let E, E_1, E_2, and E' be elliptic curves. For coprime d_1, d_2, let f_1, g_1 be d_1-isogenies and f_2, g_2 be d_2-isogenies such that the* *following diagram is commutative:*

$$\begin{CD} E @>{f_1}>> E_1 \\ @V{f_2}VV @VV{g_2}V \\ E_2 @>>{g_1}> E' \end{CD}$$

Then, an isogeny $F : E \times E' \to E_1 \times E_2$ defined by a matrix $\begin{pmatrix} f_1 & \hat{g_2} \\ -f_2 & \hat{g_1} \end{pmatrix}$ is (d, d)-isogeny where $d := d_1 + d_2$ with respect to the natural product polarizations on $E \times E'$ and $E_1 \times E_2$. In addition, the kernel of F is represented by

$$\operatorname{Ker} F = \{(\hat{f}_1(P), g_2(P)) \in E \times E' \mid P \in E_1[d]\} \ .$$

Now, we construct an attack on SIDH for the following case: $p \equiv 3 \pmod 4$ and E_0 is a supersingular elliptic curve $E_0/\mathbb{F}_{p^2} : y^2 = x^3 + x$. In addition, we can assume $N_A > N_B$ if necessary, by changing Alice and Bob in the SIDH protocol. Then, for $a := N_A - N_B$, as given in Section 5.2 below, there exists a manner to construct $\alpha(P_A), \alpha(Q_A)$ for some a-isogeny $\alpha : E_0 \to E'$ using the information of $\operatorname{End}(E_0)$, Then, we have the left-hand side diagram below by taking the pushout of φ_B and α. Hence, we have the right-hand side commutative diagram below:

$$\begin{CD} E_0 @>{\varphi_B}>> E_B \\ @V{\alpha}VV @VV{\alpha'}V \\ E' @>>{\varphi'_B}> E'_B. \end{CD} \qquad\qquad \begin{CD} E' @>{\hat{\alpha}}>> E_0 \\ @V{\varphi'_B}VV @VV{\varphi_B}V \\ E'_B @>>{\hat{\alpha'}}> E_B. \end{CD}$$

We apply Lemma 13 to the above right-hand side diagram; specifically, let $F : E' \times E_B \to E_0 \times E'_B$ be the (N_A, N_A)-isogeny given by a matrix $\begin{pmatrix} \hat{\alpha} & \hat{\varphi_B} \\ -\varphi'_B & \alpha' \end{pmatrix}$. Then, we have

$$\operatorname{Ker} F = \{(\alpha(P), \varphi_B(P)) \in E' \times E_B \mid P \in E_0[N_A]\} \ .$$

Since the attacker has $(\alpha(P_A), \varphi_B(P_A)), (\alpha(Q_A), \varphi_B(Q_A))$ which generate $\operatorname{Ker} F$, the attacker can calculate F. Then, the attacker takes a basis $\{S_1, S_2\}$ of $E_B[N_B]$ and computes $F((0, S_i)) = (\hat{\varphi_B}(S_i), \alpha'(S_i))$ for $i = 1, 2$. Since $\operatorname{Ker} \varphi_B = \langle \hat{\varphi_B}(S_1), \hat{\varphi_B}(S_2) \rangle$, the attacker gets the generator of $\operatorname{Ker} \varphi_B$.

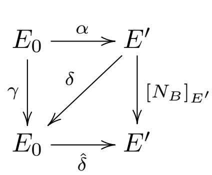
Difference between attacks on SIDH and attacks on B-SIDH. One of the differences between SIDH and B-SIDH is the number of prime factors of N_A and N_B. In SIDH, N_A and N_B are the form of 2^a or 3^b. By contrast, in B-SIDH, N_A and N_B have several prime factors. At the point of attacks, since attackers need to compute to (N_A, N_A)-isogeny F, for SIDH they compute the composition of $(2, 2)$-isogenies or $(3, 3)$-isogenies. By contrast, for B-SIDH they compute the composition high degree isogenies; that is, if $N_A = \ell_1 \cdots \ell_m$ is the prime factorization, they compute the composition of (ℓ_i, ℓ_i)-isogenies. In fact, although an implementation of attack on SIDH is given by Castryck–Decru [7, Section 9], for B-SIDH it is not given. However, we note that an attack on SIDH using higher-degree isogenies was recently provided by Jao–Laflamme [22].

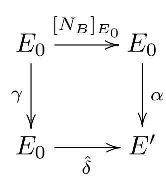
5.2 Concrete construction of attack on B-SIDH

Computation of images of some *a*-isogeny. The notation is the same as Section 5.1, that is, p is a prime number such that $p \equiv 3 \pmod 4$, E_0 is a supersingular elliptic curve $E_0/\mathbb{F}_{p^2} : y^2 = x^3 + x$, $N_A > N_B$ are coprime integers, and $\{P_A, Q_A\}$ is a basis of $E_0[N_A]$. As noted in the previous section, we can construct $\alpha(P_A), \alpha(Q_A)$ for some a-isogeny $\alpha : E_0 \to E'$ where $a := N_A - N_B$. Here, we give the construction.

We use a theory about quaternion algebra and refer to [19, Section 2]. The endomorphism ring $\mathrm{End}(E_0)$ is isomorphic to the maximal order $\mathcal{O}_0 = \langle 1, i, \frac{i+j}{2}, \frac{1+k}{2} \rangle$ with $i^2 = -1, j^2 = -p, k = ij$ of $H(-1, -p)$. Concretely, we have isomorphism by $\iota \mapsto i$ and $\pi \mapsto j$, where $\iota : E_0 \to E_0$ is $(x, y) \mapsto (-x, \sqrt{-1}y)$ and $\pi : E_0 \to E_0$ is $(x, y) \mapsto (x^p, y^p)$. In addition, we use $\texttt{FullRepresentInteger}_{\mathcal{O}_0}(M)$ of [20, Algorithm 1] which gives an element of $\mathcal{O}_0$ of norm M for a given integer $M > p$.

First, applying $\texttt{FullRepresentInteger}_{\mathcal{O}_0}(aN_B)$, we obtain an (aN_B)-isogeny $\gamma : E_0 \to E_0$. Then, we decompose γ to an a-isogeny $\alpha : E_0 \to E'$ and an N_B-isogeny $\delta : E' \to E_0$ with $\delta \circ \alpha = \gamma$. Then, since $\hat{\delta} \circ \delta = [N_B]_{E'}$, the left-hand side diagram below is commutative. Since $[N_B]_{E'} \circ \alpha = \alpha \circ [N_B]_{E_0}$, we have the right-hand side commutative diagram. Here, since $\gcd(a, N_B) = 1$ and we have $\mathrm{Ker}\, \hat{\delta} = \gamma(E_0[N_B])$, we can calculate $\hat{\delta} : E_0 \to E'$. Then, by the right-hand side commutative diagram, we have $\alpha(P_A) = \hat{\delta}(\gamma(\frac{P_A}{N_B}))$ and $\alpha(Q_A) = \hat{\delta}(\gamma(\frac{Q_A}{N_B}))$.





Composition of isogeny As discussed, when attacking B-SIDH, we calculate an isogeny of high degree. Thus, we decompose the isogeny to prime-degree isogenies. We generalize the situation slightly.

Let $K \subset A[N]$ be a maximal isotropic subgroup and $F : A \to B$ be the (N, N)-isogeny. When $N = \ell_1 \cdots \ell_m$ is the prime factorization, we have a decomposition $F = \varphi_m \circ \cdots \circ \varphi_1$ where $\varphi_i : A_i \to A_{i+1}$ is an (ℓ_i, ℓ_i)-isogeny with $A_1 = A$ and $A_{m+1} = B$.

For a basis $\{f_1, f_2\}$ of $K \simeq (\mathbb{Z}/N\mathbb{Z})^2$, from theta coordinates $\overline{f_1}, \overline{f_2}$, we calculate a theta-null point of B. To do this, we first calculate $\overline{f_1 + f_2}$ by Normal Addition. Then, we multiply $\overline{f_1}, \overline{f_2}, \overline{f_1 + f_2}$ by $\texttt{Mult}(\ell_2 \cdots \ell_m, *)$ and call them

$\overline{e_1}, \overline{e_2}, \overline{e_1+e_2}$. Since $\overline{e_1}, \overline{e_2}$ are affine lifts of a basis of $\mathrm{Ker}\,\varphi_1$, we can calculate the theta-null point of A_2. Then, we compute affine lifts $\texttt{Mult}(\ell_2 \cdots \ell_m + 1, f_1)$ of $f_1 + e_1$ and $\texttt{Multadd}(\ell_2 \cdots \ell_m, f_2, f_1, f_1 + f_2)$ of $f_1 + e_2$. From them, we calculate the theta coordinate of the image $\varphi_1(f_1) \in A_2$. Similarly, we calculate a theta coordinate of $\varphi_2(f_2) \in A_2$. By iterating this calculation m times, we have a theta-null point of B. Thus, the total cost is $5m$ times scalar multiplications and m times Normal Addition and m times codomain calculation and $2m$ times evaluations.

For $x \in A$, from an affine lift $\tilde{x}$, we calculate a theta coordinate of $F(x)$. We calculate $\overline{x+e_1}$ by Normal Addition. Then, we calculate $\overline{x+e_2}$ by Compatible Addition. Thus, we can calculate the theta coordinate of the image of $\varphi_1(x) \in A_2$.

Remark 11. In the above, when we compute F, we need $3m$ times calculations of square roots since we use two times Normal Addition and one time Compatible Addition at every step. Here, the computation of the square root on $\mathbb{F}_{p^4}$ can be performed in $O(\log p)$ operations on $\mathbb{F}_p$ as described in [39,1]. Thus, when p is extremely large, as in this case, the cost of computing square root is significantly higher compared to multiplication. As an alternative method to avoid square root computations, we may evaluate $f_1 + f_2, x + f_1$, and $x + f_2$ under φ_i at every step. Determining which method is more efficient is left as future work.

Remark 12. In the above computation, we calculate excellent lifts of $e_1, e_2, e_1 + e_2$ each time for each of $\varphi_1, \ldots, \varphi_m$, more precisely we compute normalization factors $\lambda_1^\ell, \lambda_2^\ell, \lambda_{12}^\ell$ each time. On the other hand, there is another normalization method which we compute excellent lifts of $f_1, f_2, f_1 + f_2$, more precisely we compute the normalization factors of $f_1, f_2, f_1 + f_2$ only in the first step. For the former method, as mentioned in Remark 9, it requires $O(m)\mathsf{M}$ operations. On the other hand, for the latter method, it requires $O(\log N)\mathsf{M} = O(\sum_{i=1}^{m} \log(\ell_i))\mathsf{M}$ operations. Hence, the asymptotic complexity is better for the former method. However, determining precisely which is superior in eahc N is left as future work.

Remark 13. For applying this argument to the attack on B-SIDH, since the domain A is a product of elliptic curves, A does not satisfy the assumption of non-zeroness of even theta-null points. Thus, on A, we prepare needed affine lifts by using additions of elliptic curves. Since for $A_2, \ldots, A_m$ the probability that each A_i is a product of elliptic curves is $O(\frac{10}{p})$, we consider that does not happen for sufficiently large p such as the parameter of B-SIDH.

5.3 Implementation of the attack

We implemented the attack on B-SIDH for the following parameter based on [19, Appendix.C]:

$p = \texttt{0x1E409D8D53CF3BEB65B5F41FB53B25EBEAF37761CD8BA996684150A40FFFFFFFF}$,

$N_A = 3^{56} \cdot 31 \cdot 43 \cdot 59 \cdot 271 \cdot 311 \cdot 353 \cdot 461 \cdot 593 \cdot 607 \cdot 647 \cdot 691 \cdot 743 \cdot 769 \cdot 877 \cdot 1549,$

$N_B = 2^{32} \cdot 5^{21} \cdot 7 \cdot 11 \cdot 163 \cdot 1181 \cdot 2389 \cdot 5233 \cdot 8353 \cdot 10139 \cdot 11939 \cdot 22003 \cdot 25391 \cdot 41843.$

Here, p is 257-bit and N_A is 216-bit and N_B is 213-bit. In addition, $p \equiv 3 \pmod 4$, $N_A \mid (p-1)$, $N_B \mid (p+1)$, and $N_A > N_B$. We used this parameter since (2,2)-isogeny and $2 \nmid N_A$ are not main point of concern in this paper.

As discussed in Section 5.1, we calculate the image of two points for (N_A, N_A)-isogeny $F : E' \times E_B \to E_0 \times E'_B$. The dominant step in this calculation is the computation of higher degree isogenies, such as a $(1549, 1549)$-isogeny.

We implemented this attack using our algorithms in the computer algebra system `SageMath` [41]. Then, we conducted the attack in approximately 40500 s (11.25 h) on an Apple M1 3200MHz CPU. The implementation can be found in

https://github.com/Yoshizumi-Ryo/ellell-isogeny_sage.

Remark 14. In the computation of the (N_A, N_A)-isogeny, we compute a $(3^{56}, 3^{56})$-isogeny. Then, even though it would have been possible to apply the strategy of [10, Section 3], we did not implement the strategy since the part of the $(3^{56}, 3^{56})$-isogeny does not dominate in the computation of the (N_A, N_A)-isogeny.

6 Conclusion

In this paper, we provide explicit inversion-free algorithms of (ℓ, ℓ)-isogeny between Kummer surfaces based on the Lubicz–Robert formula for an odd number ℓ.

Specifically, we proposed two algorithms that use two representations $\ell = \sum_{u=1}^{r} a_u^2$ and $\ell = 1^2 + \cdots + 1^2$ for the codomain and evaluation each. Then, we made several improvements. First, for the codomain, we reduced the complexity of computing affine lifts to half. Second, for representations $\ell = \sum_{u=1}^{r} a_u^2$, we determined the most efficient representation for each ℓ. Third, we constructed relations to compute excellent lifts from affine lifts used in the Lubicz–Robert formula. Then, we provided some improvements based on the relations. Fourth, in our algorithms, we avoided computing multiplicative inversions, which are expensive for cryptographic situations. Finally, by counting and comparing the number of arithmetic operations, we determined the most efficient algorithm for each ℓ from each of the two algorithms.

In addition, using the most efficient algorithm, we implemented the SIDH attack on B-SIDH in `SageMath`. In a setting that originally claimed 128-bit security, we were able to recover 128-bit secure B-SIDH in approximately 11 h.

References

1. Adj, G., Rodríguez-Henríquez, F.: Square Root Computation over Even Extension Fields. IEEE Transactions on Computers **63**(11), 2829–2841 (2014). https://doi.org/10.1109/TC.2013.145
2. Basso, A., Maino, L., Pope, G.: FESTA: Fast Encryption from Supersingular Torsion Attacks. In: ASIACRYPT 2023, Part VII. LNCS. vol. 14444, pp. 98–126. Springer Nature Singapore, Singapore (2023). https://doi.org/10.1007/978-981-99-8739-9_4
3. Bernstein, D.J., De Feo, L., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. In: ANTS XIV—Proceedings of the Fourteenth Algorithmic Number Theory Symposium. Open Book Ser., vol. 4, pp. 39–55. Math. Sci. Publ., Berkeley, CA (2020). https://doi.org/10.2140/obs.2020.4.39
4. Birkenhake, C., Lange, H.: Complex abelian varieties, Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 302. Springer-Verlag, Berlin, second edn. (2004), https://doi.org/10.1007/978-3-662-06307-1
5. Bisson, G., Cosset, R., Robert, D.: AVIsogenies v0.7 (Abelian Varieties and Isogenies), Magma package for explicit isogenies between abelian varieties (2021), https://www.math.u-bordeaux.fr/~damienrobert/avisogenies/
6. Bosma, W., Cannon, J., Playoust, C.: The Magma Algebra System I: The User Language. Journal of Symbolic Computation **24**(3), 235–265 (1997). https://doi.org/10.1006/jsco.1996.0125
7. Castryck, W., Decru, T.: An Efficient Key Recovery Attack on SIDH. In: EUROCRYPT 2023, Part V. LNCS. vol. 14008, pp. 423–447. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30589-4_15
8. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An Efficient Post-Quantum Commutative Group Action. In: ASIACRYPT 2018, Part III. LNCS. vol. 11274, pp. 395–427. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-030-03332-3_15
9. Chen, M., Leroux, A., Panny, L.: SCALLOP-HD: Group Action from 2-Dimensional Isogenies. In: Public-Key Cryptography – PKC 2024. pp. 190–216. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-57725-3_7
10. Chi-Domínguez, J.J., Pizarro-Madariaga, A., Riquelme, E.: Computing Isogenies of Power-Smooth Degrees Between PPAVs. Cryptology ePrint Archive, Paper 2023/508 (2023), https://eprint.iacr.org/2023/508
11. Corte-Real Santos, M., Costello, C., Smith, B.: Efficient (3,3)-isogenies on fast Kummer surfaces. Cryptology ePrint Archive, Paper 2024/144 (2024), https://eprint.iacr.org/2024/144
12. Corte-Real Santos, M., Flynn, E.V.: Isogenies on Kummer Surfaces (2024), https://arxiv.org/abs/2409.14819
13. Cosset, R., Robert, D.: Computing (ℓ, ℓ)-isogenies in polynomial time on Jacobians of genus 2 curves. Mathematics of Computation **84**(294), 1953–1975 (2015), http://www.jstor.org/stable/24489183
14. Costello, C.: B-SIDH: Supersingular Isogeny Diffie-Hellman Using Twisted Torsion. In: ASIACRYPT 2020, Part II. LNCS. vol. 12492, pp. 440–463. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_15

15. Costello, C., Hisil, H.: A Simple and Compact Algorithm for SIDH with Arbitrary Degree Isogenies. In: ASIACRYPT 2017, Part II. LNCS. vol. 10625, pp. 303–329. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_11
16. Cox, D.A.: Primes of the form $x^2 + ny^2$. A Wiley-Interscience Publication, John Wiley & Sons, Inc., New York (1989). https://doi.org/10.1002/9781118400722
17. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: SQIsignHD: New Dimensions in Cryptography. In: EUROCRYPT 2024, Part I. LNCS. vol. 14651, pp. 3–32. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-58716-0_1
18. 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. In: Advances in Cryptology – ASIACRYPT 2024. pp. 304–338. Springer Nature Singapore, Singapore (2025). https://doi.org/10.1007/978-981-96-0891-1_10
19. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In: ASIACRYPT 2020, Part I. LNCS. pp. 64–93. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_3
20. De Feo, L., Leroux, A., Longa, P., Wesolowski, B.: New Algorithms for the Deuring Correspondence. In: EUROCRYPT 2023, Part V. LNCS. vol. 14008, pp. 659–690. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30589-4_23
21. Jao, D., De Feo, L.: Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In: Post-Quantum Cryptography. pp. 19–34. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_2
22. Jao, D., Laflamme, J.: Isogeny Interpolation and the Computation of Isogenies from Higher Dimensional Representations. In: Progress in Cryptology – INDOCRYPT 2024. pp. 159–179. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-80311-6_8
23. Kani, E.: The number of curves of genus two with elliptic differentials. J. Reine Angew. Math. **485**, 93–121 (1997). https://doi.org/10.1515/crll.1997.485.93
24. Koizumi, S.: Theta Relations and Projective Normality of Abelian Varieties. American Journal of Mathematics **98**(4), 865–889 (1976), http://www.jstor.org/stable/2374034
25. Lubicz, D., Robert, D.: Efficient pairing computation with theta functions. In: Algorithmic number theory, Lecture Notes in Comput. Sci., vol. 6197, pp. 251–269. Springer, Berlin (2010), https://doi.org/10.1007/978-3-642-14518-6_21
26. Lubicz, D., Robert, D.: Computing isogenies between abelian varieties. Compos. Math. **148**(5), 1483–1515 (2012), https://doi.org/10.1112/S0010437X12000243
27. Lubicz, D., Robert, D.: A generalisation of Miller's algorithm and applications to pairing computations on abelian varieties. J. Symbolic Comput. **67**, 68–92 (2015), https://doi.org/10.1016/j.jsc.2014.08.001
28. Lubicz, D., Robert, D.: Arithmetic on abelian and Kummer varieties. Finite Fields Appl. **39**, 130–158 (2016), https://doi.org/10.1016/j.ffa.2016.01.009
29. Lubicz, D., Robert, D.: Fast change of level and applications to isogenies. Res. Number Theory **9**(1), Paper No. 7, 28 (2023), https://doi.org/10.1007/s40993-022-00407-9
30. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A Direct Key Recovery Attack on SIDH. In: EUROCRYPT 2023, Part V. LNCS. vol. 14008, pp.

448–471. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30589-4_16
31. Meyer, M., Reith, S.: A Faster Way to the CSIDH. In: INDOCRYPT 2018. vol. 11356, pp. 137–152. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-030-05378-9_8
32. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation **48**, 243–264 (1987), https://api.semanticscholar.org/CorpusID:4262792
33. Mumford, D.: On the equations defining abelian varieties. I. Inventiones mathematicae **1**(4), 287–354 (1966). https://doi.org/10.1007/BF01389737
34. Mumford, D.: Abelian Varieties Tata Institute of Fundamental Research (1970), https://api.semanticscholar.org/CorpusID:115766011
35. Mumford, D.: Tata lectures on theta. II, Progress in Mathematics, vol. 43. Birkhäuser Boston, Inc., Boston, MA (1984), https://doi.org/10.1007/978-0-8176-4578-6, jacobian theta functions and differential equations, With the collaboration of C. Musili, M. Nori, E. Previato, M. Stillman and H. Umemura
36. Nakagawa, K., Onuki, H.: QFESTA: Efficient Algorithms and Parameters for FESTA Using Quaternion Algebras. In: CRYPTO 2024, Part V. LNCS. vol. 14924, pp. 75–106. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-68388-6_4
37. Robert, D.: Efficient algorithms for abelian varieties and their moduli spaces. Habilitation à diriger des recherches, Université de Bordeaux (UB) (Mar 2021), https://hal.science/tel-03498268
38. Robert, D.: Breaking SIDH in Polynomial Time. In: EUROCRYPT 2023, Part V. LNCS. vol. 14008, pp. 472–503. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30589-4_17
39. Scott, M.: A note on the calculation of some functions in finite fields: Tricks of the Trade. Cryptology ePrint Archive, Paper 2020/1497 (2020), https://eprint.iacr.org/2020/1497
40. Stange, K.E.: Elliptic Nets and Elliptic Curves . Ph.D. thesis, Brown University (2008)
41. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 10.3) (2024), https://www.sagemath.org
42. Vélu, J.: Isogénies entre courbes elliptiques. C. R. Acad. Sci. Paris Sér. A-B **273**, A238–A241 (1971)

A Comparison with Corte-Real Santos and Flynn's algorithm

In this section, we compare the performance of our algorithms with those of Corte-Real Santos and Flynn's ones [12]. Here, we compare the algorithms by measuring their computational times in `Magma` [6]. To compare them fairly, we should compare the number of operations of those algorithms. However, we compare using the computational times since Corte-Real Santos and Flynn's algorithm lacks a precise operation count, only providing an upper bound.

Our implementation in `Magma` is found at

https://github.com/Yoshizumi-Ryo/ellell_isogeny_Magma,

and Corte-Real Santos and Flynn's one is found at

https://github.com/mariascrs/NN_isogenies.

We use the following prime p whose bit length is 256:

$p =$ `0xe221f97c30e94e1d00000000000000000000000000000003f998e2aedc19df827`.

Then, $\ell|(p+1)$ for all primes ℓ such that $2 \leq \ell \leq 53$, and $p \equiv 3 \pmod 4$. Let $f : A \to B$ be an (ℓ, ℓ)-isogeny between abelian surfaces over $\mathbb{F}_{p^2}$. Then, there is the induced isogeny $f : K_A \to K_B$ between their Kummer surfaces.

Let $\{e_1, e_2\}$ be a basis of the kernel of f and suppose theta coordinates of $0, e_1, e_2, e_1 + e_2$ are given. Which of our algorithms and Corte-Real Santos and Flynn's algorithms is the most efficient to compute to the theta-null point of B?

Here, we use Corte-Real Santos and Flynn's algorithm `GetIsogeny`. More precisely, `GetIsogeny` has two types "GE" and "sqrt" for $\ell \geq 7$, which differ only in the computational method; see [12]. The algorithm provides the equation of the isogeny in theta coordinates. Thus, after we compute the equation using `GetIsogeny`, by substituting the theta-null point of A, we get the theta-null point of B.

Hence, we have four methods to solve the above question: `CodSq`, `CodOne` in Section 3.4, and the above ways for GE and sqrt. Table 7 shows the average time (in millisecond) over a number of trials to compute them for $\ell = 5, 7, 11, 13$. For both our cases and Corte-Real Santos and Flynn's ones, domain abelian surfaces are Jacobians of genus 2 curves, however, due to implementation challenges, different abelian surfaces are used. In addition, in our method, the same Jacobian is used for all trials. However, the number of operations does not depend on the Jacobian, so we consider it not to be an issue. For $\ell = 5, 7, 11, 13$ in our method and for $\ell = 5$ and 7 in Corte-Real Santos and Flynn's method, we take the average over 100 runs. For $\ell = 11$ and 13 in Corte-Real Santos and Flynn's method, we take the average over 20 runs since the computation time is large and detailed computation time is not required, which significantly differs from the time of our methods. For $\ell > 13$, we conclude that our method is more efficient than the Corte-Real Santos and Flynn's method since the complexity

ℓ	CodSq	CodOne	GetIsogeny(GE) +a substitution	GetIsogeny(sqrt) +a substitution
5	3	2	6	-
7	5	4	30	35
11	13	12	716	649
13	16	17	3375	3126

Table 7. Average times (ms) to compute theta-null point of a codomain. We take the average over 100 (resp. 20) runs for $\ell = 5, 7, 11, 13$ in our methods and for $\ell = 5, 7$ in the Corte-Real Santos and Flynn's (resp. $\ell = 11, 13$ in the Corte-Real Santos and Flynn's) method.

of GetIsogeny is considerably large for such ℓ [12, Table 1]. The red underlined times are the most efficient method from them for each ℓ.

From Table 7, it can be concluded that our method is more efficient for $\ell \geq 7$. For $\ell = 5$, we consider that there may be a margin of error of the computation time. This result matches the predictions based on the operation counts of our algorithms and the complexities of Corte-Real Santos and Flynn's ones.

Next, we consider an isogeny chain $F : A_1 \to A_2 \to \cdots \to A_m$. Then, we consider computing images of n points under the chain for $n = 0, 1, 2, 3$. The case of $n = 0$ is just computing the theta-null point of the codomain A_m. Let $x_1, \ldots, x_n$ be n points of A_1 and $\{e_1, e_2\}$ be a basis of the kernel of F. Then, for our method, except for a theta-null point, we need to send $(3 + 3n)$ points $e_1, e_2, e_1 + e_2, x_i, x_i + e_1, x_i + e_2$ for $1 \leq i \leq n$ for each step of the chain. By contrast, for Corte-Real Santos and Flynn's method, we need to send $(2 + n)$ points e_1, e_2, x_i for $1 \leq i \leq n$ for each step. Note that in both methods, it is not necessary to calculate the image of elements other than x in the last isogeny $A_{m-1} \to A_m$. Thus, we compare computational time to compute images of $(3 + 3n)$ points for an (ℓ, ℓ)-isogeny for our method with to compute images of $(2 + n)$ points for the Corte-Real Santos and Flynn's method. Here, note that Corte-Real Santos and Flynn's method is efficient in terms of requiring fewer points to be prepared.

When using our method, we can compute them using one CodOne or CodSq, and $(3 + 3n)$ times EvalOne. From CodOne and CodSq, we use more efficient method concluded from Table 7 for each ℓ. The reason to use EvalOne not EvalSq is the operation count of EvalOne is more efficient for small ℓ, as shown in Table 6 in Section 4. By contrast, when using the Corte-Real Santos and the Flynn's method, we can compute them using GetIsogeny and $(2 + n)$ times substitutions. Then, for GetIsogeny, we use more efficient method from GE and sqrt concluded from Table 7.

Table 8 shows the average times (in millisecond) over a number of trials to compute them to implement them for $\ell = 5, 7, 11, 13$ and $n = 0, 1, 2, 3$. As well as Table 7, for both cases, the domain abelian surfaces are Jacobians of genus 2 curves, however, different abelian surfaces are used. For $\ell = 5, 7, 11, 13$ in our method and for $\ell = 5$ and 7 in Corte-Real Santos and Flynn's method, we take the average over 100 runs. For $\ell = 11$ and 13 in Corte-Real Santos and Flynn's method, we take the average over 20 runs. For the same reason as the first question, we consider $\ell \leq 13$. In the same method as Table 7, the red underlined times are the most efficient method for each ℓ and n.

$\ell \backslash n$	0	1	2	3
5 (our)	17	33	47	62
5 (CF)	7	7	7	7
7 (our)	36	65	95	124
7 (CF)	32	32	33	33
11 (our)	85	156	229	301
11 (CF)	648	650	650	651
13 (our)	126	232	341	447
13 (CF)	3131	3133	3135	3136

Table 8. Average times (ms) to compute the theta-null point and evaluation of $(3+3n)$ (resp. $(2+n)$) points for an (ℓ, ℓ)-isogeny for our (resp. Corte-Real Santos and Flynn's (CF)) method. We take the average over 100 (resp. 20) runs for $\ell = 5, 7, 11, 13$ in our methods and for $\ell = 5, 7$ in the Corte-Real Santos and Flynn's (resp. $\ell = 11, 13$ in the Corte-Real Santos and Flynn's) ones.

From Table 8, we should use the red underlined method for ℓ and n. The Corte-Real Santos and Flynn's method is barely affected by n, since the dominant sub-algorithm `GetIsogeny` is just one computation for any n. In summary, for $\ell = 5$ and 7, the Corte-Real Santos and Flynn's method is always more efficient than ours. For $\ell \geq 11$, our method is more efficient than the Corte-Real Santos and Flynn's one.

Remark 15. As mentioned above, the advantage of the Corte-Real Santos and Flynn's method is that the computational complexity does not increase as the number of points increases, as seen in Table 8, since it is used to compute the equation of the isogeny. On the other hand, with our method, if there are enough point computations, we should be able to compute the equation of isogeny via Lagrange interpolation. The exact number of points required and a comparison of efficiency is left as future work.

B Additional arithmetic on Kummer surfaces of Section 2.2

As described in Remark 1 in Section 2.2, in this section, we introduce Normal Addition and Compatible Addition. They are used attack on B-SIDH.

Normal Addition We consider the case where $(\tilde{\theta}_i(x-y))_i$ is not given as a part of input. Then, note that we cannot distinguish $(\tilde{\theta}_i(x+y))_i$ from $(\tilde{\theta}_i(x-y))_i$ using

$(\tilde{\theta}_i(x))_i, (\tilde{\theta}_i(y))_i$ only, since $(\tilde{\theta}_i(y))_i = (\tilde{\theta}_i(-y))_i$. Nonetheless, we can compute a set (unordered pair) $\{(X_i)_i, (Y_i)_i\} := \{(\tilde{\theta}_i(x+y))_i, (\tilde{\theta}_i(x-y))_i\}$ as follows. First, note that for any pair of $(\tilde{\theta}_i(x+y))_i$ and $(\tilde{\theta}_i(x-y))_i$ which satisfies (3) and for any $\lambda \in \mathbb{C}^*$, a pair of $\lambda * (\tilde{\theta}_i(x+y))_i$ and $\frac{1}{\lambda} * (\tilde{\theta}_i(x-y))_i$ also satisfies (3). Thus, we may fix $X_0 := 1$, and then we have $Y_0 = \kappa_{00}$ by (7). Here, we assume that $\kappa_{00} \neq 0$. If necessary, we replace by another i with $\kappa_{ii} \neq 0$.

In the above notation, equality (6) becomes $\kappa_{ij} = \frac{1}{2}(X_iY_j + X_jY_i)$. Thus, for all $i \in (\mathbb{Z}/2\mathbb{Z})^2$, we have $\frac{X_i}{X_0} \cdot \frac{Y_i}{Y_0} = \frac{\kappa_{ii}}{\kappa_{00}}$ and $\frac{X_i}{X_0} + \frac{Y_i}{Y_0} = \frac{2\kappa_{i0}}{\kappa_{00}}$. Hence, $\frac{X_i}{X_0}$ and $\frac{Y_i}{Y_0}$ are solutions of the following quadratic equation:

$$\kappa_{00}t^2 - 2\kappa_{i0}t + \kappa_{ii} = 0 \ . \tag{23}$$

If x or y is a 2-torsion point, we have $x - y \in \{\pm(x+y)\}$ and hence $(\theta_i(x+y))_i = (\theta_i(x-y))_i$ as projective theta coordinates. Therefore, $\frac{X_i}{X_0} = \frac{Y_i}{Y_0} = \frac{\kappa_{i0}}{\kappa_{00}}$. Thus, in this case, we can compute the set $\{(X_i)_i, (Y_i)_i\}$ from κ_{ij}.

Otherwise, we have $x - y \notin \{\pm(x+y)\}$ and hence $(X_i)_i \neq (Y_i)_i$ as projective theta coordinates. Therefore, there exists $\alpha \in (\mathbb{Z}/2\mathbb{Z})^2$ such that $\frac{X_\alpha}{X_0} \neq \frac{Y_\alpha}{Y_0}$. Then, the quadratic equation (23) with $i = \alpha$ has two distinct solutions; among them, we can set $\frac{X_\alpha}{X_0} = \frac{\kappa_{\alpha 0} + \sqrt{D_\alpha}}{\kappa_{00}}$ by symmetry where $D_\alpha := \kappa_{\alpha 0}^2 - \kappa_{\alpha\alpha}\kappa_{00}$ (note that now $D_\alpha \neq 0$). Since we fixed $X_0 = 1$, we have

$$X_\alpha = \frac{\kappa_{\alpha 0} + \sqrt{D_\alpha}}{\kappa_{00}} \ .$$

Moreover, for the remaining $i \in (\mathbb{Z}/2\mathbb{Z})^2 \smallsetminus \{0, \alpha\}$, we have the following linear equation:

$$\begin{pmatrix} 1 & 1 \\ \frac{X_\alpha}{X_0} & \frac{Y_\alpha}{Y_0} \end{pmatrix} \begin{pmatrix} \frac{Y_i}{Y_0} \\ \frac{X_i}{X_0} \end{pmatrix} = \begin{pmatrix} \frac{2\kappa_{i0}}{\kappa_{00}} \\ \frac{2\kappa_{i\alpha}}{\kappa_{00}} \end{pmatrix} \ .$$

Since $\det \begin{pmatrix} 1 & 1 \\ \frac{X_\alpha}{X_0} & \frac{Y_\alpha}{Y_0} \end{pmatrix} = \frac{-2\sqrt{D_\alpha}}{\kappa_{00}} \neq 0$, By solving the above linear equation, we can calculate X_i as follows:

$$X_i = \frac{X_\alpha \kappa_{i0} - \kappa_{i\alpha}}{\sqrt{D_\alpha}} \ .$$

Moreover, since $\kappa_{ii} = X_iY_i$, we have $Y_i = \frac{\kappa_{ii}}{X_i}$ if $X_i \neq 0$. Even if $X_i = 0$, we can compute Y_i in the same manner as that for X_i.

Thus, from affine lifts $(\tilde{\theta}_i(x))_i, (\tilde{\theta}_i(y))_i$, we obtained the set $\{(\tilde{\theta}_i(x+y))_i, (\tilde{\theta}_i(x-y))_i\}$. We call this algorithm *Normal Addition* (cf. [28, Section 5.2]). Remark that this operation requires one square root computation.

Compatible Addition For given $(\tilde{\theta}_i(y))_i, (\tilde{\theta}_i(z))_i, (\tilde{\theta}_i(x+y))_i, (\tilde{\theta}_i(x+z))_i$, we can compute $(\tilde{\theta}_i(y+z))_i$ as follows. If y or z is a 2-torsion point, since $(\tilde{\theta}_i(y+z))_i = (\tilde{\theta}_i(y-z))_i$ as projective theta coordinates, it suffices to compute the Normal Addition of $(\tilde{\theta}_i(y))_i$ and $(\tilde{\theta}_i(z))_i$. Otherwise, we can compute $(\tilde{\theta}_i(y+z))_i$ by using Normal Addition twice as follows. First, we calculate the set $\{Y, Z\} :=$

$\{(\tilde{\theta}_i(y+z))_i, (\tilde{\theta}_i(y-z))_i\}$ from $(\tilde{\theta}_i(y))_i, (\tilde{\theta}_i(z))_i$ using Normal Addition. Then, we compute the set S of Normal Addition of Y and $(\tilde{\theta}_i(x+y))_i$. If $Y = (\tilde{\theta}_i(y+z))_i$, we have $S = \{(\tilde{\theta}_i(x+2y+z))_i, (\tilde{\theta}_i(x-z))_i\}$. Thus, in this case, $(\tilde{\theta}_i(x+z))_i$ is not contained in S since neither y nor z is a 2-torsion point. By contrast, if $Y = (\tilde{\theta}_i(y-z))_i$, we have $S = \{(\tilde{\theta}_i(x+2y-z))_i, (\tilde{\theta}_i(x+z))_i\}$. Thus, if S contains the projective theta coordinate $(\tilde{\theta}_i(x+z))_i$, we have $Z = (\tilde{\theta}_i(y+z))_i$. Otherwise, we have $Y = (\tilde{\theta}_i(y+z))_i$. We call this algorithm *Compatible Addition* (cf. [26, Section 3.2.1]).

C Explicit algorithms of Section 3.2

In this appendix, as described in Section 3.2, we provide concrete algorithms of arithmetic on Kummer surfaces.

C.1 Batch inversion

First, to unify the denominators of some given fractions (Lemma 2), we give an algorithm to compute some products from given elements of k (Algorithm 5) and an evaluation of its cost (Lemma 14). For any integer $M \geq 0$, we write the binary expansion as $M = (d_{n-1}, \ldots, d_0)_2$, where $M = \sum_{i=0}^{n-1} d_i 2^i$ for $d_i \in \{0, 1\}$. Here, we do *not* require $d_{n-1} = 1$.

Lemma 14. *Let $N \geq 2$ and $a_0, \ldots, a_{N-1} \in k$. Then the output of Algorithm 5 satisfies that $\widetilde{\alpha} = \alpha := a_0 \cdots a_{N-1}$ and $\widetilde{\alpha}_M = \alpha_M := a_0 \cdots a_{M-1} a_{M+1} \cdots a_{N-1}$ for any $M = 0, \ldots, N-1$, and Algorithm 5 requires* $(3N-5)\mathsf{M}$. *If the part α of the output is not needed, then the cost reduces to* $(3N-6)\mathsf{M}$.

Proof. As for line 2 in the algorithm, let L denote the set of all leaves of the binary tree T; and for each node v of T, let $L(v)$ denote the set of all $v' \in L$ that is covered by v, that is, the upward path from v' to the root of T involves the node v. Then, we have $\alpha = \prod_{v \in L} \mathtt{a}[v]$ and $\alpha_M = \prod_{v \in L \smallsetminus \{v[n;M]\}} \mathtt{a}[v]$ for any $M \in \{0, \ldots, N-1\}$. Now a recursive argument implies that $\mathtt{a}[v] = \prod_{w \in L(v)} \mathtt{a}[w]$ for any node v of T; this follows from the fact that for each non-leaf node v, $L(v)$ is the disjoint union of $L(v'_1)$ and $L(v'_2)$ if v has two child nodes v'_1 and v'_2, and $L(v) = L(v')$ otherwise where v' is the unique child node of v. The former case occurs $N-1$ times in total by the argument of "counting losers in knockout tournament", therefore $N-1$ multiplications on k are performed during this process. Finally, for the root $v[0;0]$ of T, we have

$$\widetilde{\alpha} = \mathtt{a}[v[0;0]] = \prod_{w \in L(v[0;0])} \mathtt{a}[w] = \prod_{w \in L} \mathtt{a}[w] = \alpha$$

since $L(v[0;0]) = L$. Hence, the part $\widetilde{\alpha}$ of the output is correct.

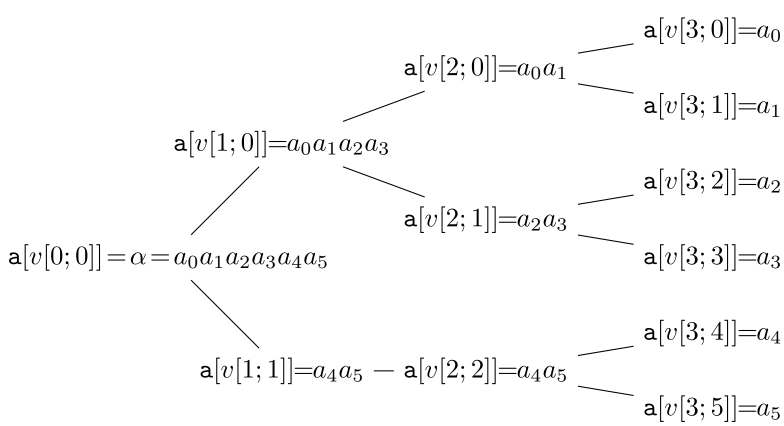
Secondly, a recursive argument also implies that $\mathtt{b}[v] = \prod_{w \in L \smallsetminus L(v)} \mathtt{a}[w]$ for any node v of T. Indeed, this follows from the fact that for each non-leaf node v, if v has two child nodes v'_1 and v'_2, then $L \smallsetminus L(v'_1)$ is the disjoint union of $L \smallsetminus L(v)$ and $L(v_2')$ and vice versa. If v has a single child node v', then $L \smallsetminus L(v') = L \smallsetminus L(v)$. (Note that the relations $\mathtt{b}[v[1;0]] = \mathtt{b}[v[0;0]] \cdot \mathtt{a}[v[1;1]]$ and $\mathtt{b}[v[1;1]] = \mathtt{b}[v[0;0]] \cdot \mathtt{a}[v[1;0]]$ also hold for nodes at level 1.) Multiplication on k occurs only when a non-leaf node v is of the former type (except for the case of the root $v = v[0;0]$); now two multiplications on k are performed in the calculation at the two child nodes. By the argument at the previous paragraph, there are $(N-1)-1 = N-2$ such nodes v, hence there are $2(N-2)$ multiplications in total. Finally, for each leaf $v[n;M]$ with $0 \le M \le N-1$, we have

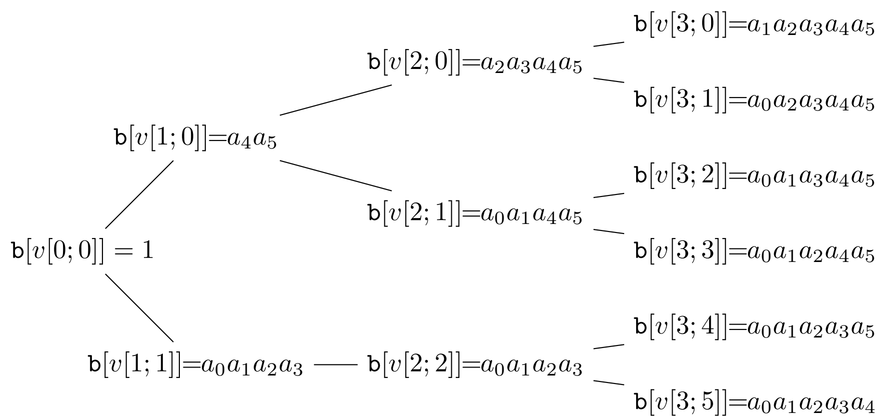
$$\begin{aligned}\widetilde{\alpha}_M = \mathtt{b}[v[n;M]] = \prod_{w \in L \smallsetminus L(v[n;M])} \mathtt{a}[w] &= \prod_{w \in L \smallsetminus \{v[n;M]\}} \mathtt{a}[w] \\ &= a_0 \cdots a_{M-1} a_{M+1} \cdots a_{N-1} = \alpha_M\end{aligned}$$

since $L(v[n;M]) = \{v[n;M]\}$. Hence, the part $\widetilde{\alpha}_M$ of the output is correct.

The total number of multiplications is $(N-1)+2(N-2) = 3N-5$. Now, if the part α of the output is not required, the calculation of $\mathtt{a}[v[0;0]]$ in the algorithm can be removed, decreasing the number of multiplications by one; that is, $3N-6$ multiplications in total. □

Example 1. We give an example of Lemma 14 in the case of $N = 6$. Then, we use the following two binary trees:





C.2 Concrete algorithms

Here, we present the concrete algorithms used in Lemma 3, Lemma 4, and Lemma 5:

Content	Lemma	Algorithm
Computing κ_{ii}	Lemma 3	Algorithm 6
Differential Addition, Doubling	Lemma 4	Algorithm 7
Three-way Addition	Lemma 5	Algorithm 8

Table 9. Correspondence between lemmas and algorithms

Algorithm 5 Algorithm to compute some products

Input: N elements $a_0, \ldots, a_{N-1} \in k$ $(N \geq 2)$

Output: N products $\alpha_M := a_0 \cdots a_{M-1} a_{M+1} \cdots a_{N-1}$ for $0 \leq M \leq N-1$ and a product $\alpha := a_0 \cdots a_{N-1}$

Write $N - 1 = (d_{n-1}, \ldots, d_0)_2$ where $d_{n-1} = 1$ # possible since $N \geq 2$

Generate a binary tree T with 0-th level (root) to n-th level (leaves), where
- for $\ell = 0, \ldots, n$, the ℓ-th level consists of nodes $v[\ell; 0], v[\ell; 1], \ldots, v[\ell; N'_\ell]$ where $N'_\ell = \lfloor (N-1)/2^{n-\ell} \rfloor = (d_{n-1}, \ldots, d_{n-\ell})_2$ (now $N'_0 = 0$);
- for $\ell = 0, \ldots, n-1$, node $v[\ell; c]$ has child node(s) $v[\ell+1; 2c]$ and $v[\ell+1; 2c+1]$ (if it exists); we call $v[\ell+1; 2c+1]$ *the sibling node* of $v[\ell+1; 2c]$ and vice versa

\# Multiply from leaves to the root

for each leaf $v := v[n; c]$ **do**
    $\mathtt{a}[v] := a_c$
end for
for $\ell = n-1$ **downto** 0 **do**
    for each node $v := v[\ell; c]$ **do**
        if v has two child nodes v'_1 and v'_2 **then**
            $\mathtt{a}[v] := \mathtt{a}[v'_1] \cdot \mathtt{a}[v'_2]$ (▷) 1M
        else
            $\mathtt{a}[v] := \mathtt{a}[v']$ for the unique child node v' of v
        end if
    end for
end for

\# Multiply from root to leaves

$\mathtt{b}[v[0;0]] := 1 \in k$ # $v[0;0]$ is the root of T

$\mathtt{b}[v[1;0]] := \mathtt{a}[v[1;1]]$ and $\mathtt{b}[v[1;1]] := \mathtt{a}[v[1;0]]$ # $N'_1 = 1$ since $N \geq 2$

for $\ell = 2$ **to** n **do**
    for each node $v := v[\ell; c]$ (with its parent node $\hat{v}$) **do**
        if v has the sibling node v' **then**
            $\mathtt{b}[v] := \mathtt{b}[\hat{v}] \cdot \mathtt{a}[v']$ (▷) 1M
        else
            $\mathtt{b}[v] := \mathtt{b}[\hat{v}]$
        end if
    end for
end for

return $\tilde{\alpha}_M := \mathtt{b}[v[n; M]]$ for $M = 0, \ldots, N-1$ (as α_M) and $\tilde{\alpha} := \mathtt{a}[v[0;0]]$ (as α)

(▷) total: $(3N-5)$M

Algorithm 6 Algorithm to calculate κ_{ii} in Lemma 3 (i) (resp. (ii))

Input: Affine lifts $(\theta'_i(x), d_x)_i$, $(\theta'_i(y), d_y)_i$.

Output: κ_{ii} for all $i \in (\mathbb{Z}/2\mathbb{Z})^2$.

Calculate $\theta'_i(0)^2, \theta'_i(x)^2, \theta'_i(y)^2$ for all $i \in (\mathbb{Z}/2\mathbb{Z})^2$. ($\triangleright$)$4\mathsf{S}_0 + 8\mathsf{S}$ (resp. $4\mathsf{S}_0 + 4\mathsf{S}$)

for $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})^2}$ (4 elements in total) **do**

  $z'^{\chi}_0 := (\sum_{t \in (\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta'_t(x)^2) \cdot (\sum_{t \in (\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta'_t(y)^2)$. ($\triangleright$)$1\mathsf{M}$ (resp. $1\mathsf{S}$)

  $d_\chi := \sum_{t \in (\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta'_t(0)^2$.

end for

$(z'^{\chi}_0, d)_{\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})^2}} :=$ `Commondenom`$((z'^{\chi}_0, d_\chi)_{\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})^2}})$. ($\triangleright$)$C_{cd}(4,1) = 11\mathsf{M}$

$\kappa'_{ii} := \sum_{\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})^2}} \chi(i) z'^{\chi}_0$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$.

$d_\kappa := 4d \cdot (d_x \cdot d_y)^2$. ($\triangleright$)$1\mathsf{S} + 2\mathsf{M}$ (resp. $2\mathsf{S} + 1\mathsf{M}$)

return $(\kappa'_{ii}, d_\kappa)_i$. ($\triangleright$) total : $4\mathsf{S}_0 + 9\mathsf{S} + 17\mathsf{M}$ (resp. $4\mathsf{S}_0 + 10\mathsf{S} + 12\mathsf{M}$)

Algorithm 7 Differential Addition (resp. Doubling)

Input: Affine lifts $(\theta'_i(x), d_x)_i$, $(\theta'_i(y), d_y)_i$, and $(\theta'_i(x-y), d_{x-y})_i$.

Output: The affine lift $(\tilde{\theta}_i(x+y))_i$.

Compute $(\kappa'_{ii}, d_\kappa)_i$ for $i \in (\mathbb{Z}/2\mathbb{Z})^2$ by Algorithm 6.
($\triangleright$)$4\mathsf{S}_0 + 9\mathsf{S} + 17\mathsf{M}$ (resp. $4\mathsf{S}_0 + 10\mathsf{S} + 12\mathsf{M}$)

$(\theta'_i(x+y), d_{x+y})_{i \in (\mathbb{Z}/2\mathbb{Z})^2} :=$ `Commondenom`$((\kappa'_{ii}, \theta_i(x-y))_{i \in (\mathbb{Z}/2\mathbb{Z})^2})$.
($\triangleright$)$C_{cd}(4,1) = 11\mathsf{M}$

$\theta'_i(x+y) := d_{x-y} \cdot \theta'_i(x+y)$ for all $i \in (\mathbb{Z}/2\mathbb{Z})^2$. ($\triangleright$)$4\mathsf{M}$

$d_{x+y} := d_\kappa \cdot d_{x+y}$. ($\triangleright$)$1\mathsf{M}$

return $(\theta'_i(x+y), d_{x+y})_i$. ($\triangleright$)total: $4\mathsf{S}_0 + 9\mathsf{S} + 33\mathsf{M}$ (resp. $4\mathsf{S}_0 + 10\mathsf{S} + 28\mathsf{M}$)

Algorithm 8 Three-way Addition

Input: Affine lifts $(\theta'_i(x), d_x)_i$, $(\theta'_i(y), d_y)_i$, $(\theta'_i(z), d_z)_i$, $(\theta'_i(x+y), d_{x+y})_i$, $(\theta'_i(y+z), d_{y+z})_i$, and $(\theta'_i(z+x), d_{z+x})_i$.

Output: The affine lift $(\tilde{\theta}_i(x+y+z))_i$.

$R_1^\chi := \sum_{t \in (\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta'_t(0)\theta'_t(y+z)$ for all $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})}^2$. (▷)4M

$R_2^\chi := \sum_{t \in (\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta'_t(z+x)\theta'_t(x+y)$ for all $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})}^2$. (▷)4M

$L_2^\chi := \sum_{t \in (\mathbb{Z}/2\mathbb{Z})^2} \chi(t)\theta'_t(y)\theta'_t(z)$ for all $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})}^2$. (▷)4M

for $\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})}^2$ **do**

  $E'^\chi := R_1^\chi \cdot R_2^\chi$. (▷)1M

  $d_\chi := L_2^\chi$.

end for

$(\tilde{E}^\chi, d)_{\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})}^2} :=$ `Commondenom`$((E'^\chi, d_\chi)_{\chi \in \widehat{(\mathbb{Z}/2\mathbb{Z})}^2})$. (▷)$C_{cd}(4,1) = 11$M

for $i \in (\mathbb{Z}/2\mathbb{Z})^2$ **do**

  $\theta'_i(x+y+z) := \sum_\chi \chi(i)\tilde{E}^\chi$.

  $d_i := 4\theta'_i(x)$.

end for

$(\theta'_i(x+y+z), d_{x+y+z})_{i \in (\mathbb{Z}/2\mathbb{Z})^2} :=$ `Commondenom`$((\theta'_i(x+y+z), d_i)_{i \in (\mathbb{Z}/2\mathbb{Z})^2})$. (▷)$C_{cd}(4,1) = 11$M

$d_{x+y+z} := d_{x+y+z} \cdot d_{y+z} \cdot d_{z+x} \cdot d_{x+y} \cdot d$. (▷)4M

$d_{xyz} := d_x \cdot d_y \cdot d_z$. (▷)2M

Calculate $\theta'_i(x+y+z) := \theta'_i(x+y+z) \cdot d_{xyz}$ for all $i \in (\mathbb{Z}/2\mathbb{Z})^2$. (▷)4M

return $(\theta'_i(x+y+z), d_{x+y+z})_i$. (▷) total: 48M