Efficient Two-Dimensional Representation Evaluation via Dual Isogenies with Applications to Isogeny-Based Cryptography

Jianming Lin[1,2], Saiyu Wang[1], Chang-An Zhao[1,3], and Yuhao Zheng[1]

[1] School of Mathematics, Sun Yat-sen University, Guangzhou 510275, China
linjm76@mail.sysu.edu.cn
wangsy58@mail2.sysu.edu.cn
zhaochan3@mail.sysu.edu.cn
zhengyh57@mail2.sysu.edu.cn
[2] School of Mathematics, Jiaying University, Meizhou 514015, China
[3] Guangdong Key Laboratory of Information Security, Guangzhou 510006, China

Abstract. The evaluation of two-dimensional isogeny representations is a fundamental operation in many isogeny-based cryptographic schemes, notably the public-key encryption schemes POKÉ (Eurocrypt 2025) and PIKE (PKC 2026), as well as the digital signature schemes SQIsign2D-West (Asiacrypt 2024) and SQIsign2DPush (Eurocrypt 2026). However, existing two-dimensional evaluation algorithms rely on time-consuming discrete logarithm computations and torsion-basis generation, creating a severe performance bottleneck for the entire protocol. In this paper, we propose a novel algorithm for evaluating two-dimensional representations by leveraging the properties of dual isogenies, thereby eliminating the need for costly discrete logarithms. We apply this methodology to significantly accelerate some isogeny-based protocols, including POKÉ, SQIsign2D, and their variants. We provide optimized C implementations for the speed-optimized variants of POKÉ and SQIsign2D using our proposed method. Experimental benchmarks demonstrate that the fast variants of POKÉ combined with our new techniques achieve computational speedups of up to 3.47× in terms of CPU clock cycles compared to POKÉ at the NIST-I security level. Additionally, our explicit analysis and implementation results confirm that the signing phase of SQIsign2DPush (resp. the NIST post-quantum cryptography standardization candidate SQIsign-v3.0), when using our method, achieves a performance improvement of up to 11.24% (resp. 5.02%) across the three NIST security levels.

Keywords: Post-Quantum Cryptography · Two-Dimensional Representations · Theta Coordinates · POKÉ · SQIsign · Dual Isogenies

1 Introduction

Isogeny-based cryptography serves as a fundamental building block of post-quantum cryptography owing to its small public-key and signature sizes. SIDH (Supersingular Isogeny Diffie-Hellman), which was proposed by De Feo and Jao [27] in 2011, was the first practical isogeny-based protocol that offered both efficient performance and resistance to known quantum attacks at the time. In 2017, an SIDH-based key encapsulation mechanism named SIKE [12] became a candidate in NIST's post-quantum cryptography standardization project.

Nevertheless, in 2022, Castryck and Decru [13], Maino *et al.* [31], and Robert [42] successively proposed polynomial-time key-recovery attacks that completely broke SIDH.

Consequently, several SIDH-based cryptographic schemes, such as SIKE and B-SIDH [16], were rendered insecure. In response, various countermeasures were proposed to design secure protocols retaining an SIDH-like structure. By masking either the isogeny degrees or the auxiliary torsion points, Fouotsa, Moriya, and Petit [26] introduced M-SIDH and MD-SIDH to evade these attacks. Subsequently, Basso and Fouotsa [6] proposed binSIDH and terSIDH, which maintain resistance against the SIDH attacks while achieving improved efficiency over MD-SIDH and M-SIDH. However, the public-key encryption schemes and key encapsulation mechanisms built upon these protocols remain computationally inefficient compared to the original SIDH framework.

On the other hand, Kani's lemma [28], employed in the SIDH attacks, provides an efficient higher-dimensional representation of isogenies of non-smooth degrees between two given curves. Hence, it can be used to construct new higher-dimensional protocols, thereby opening a promising direction in isogeny-based cryptography. This novel technique has been utilized to achieve significant efficiency improvements for existing one-dimensional signature schemes (e.g., SQIsign [21]), and to establish corresponding higher-dimensional variants [19,3,4,5,39,24,47,38]. Among them, the SQIsign2D-West signature scheme has become a round-3 candidate of NIST's post-quantum cryptography standardization project, under the name SQIsign-v3.0 [1]. In the realm of public-key encryption, Basso, Maino, and Pope proposed FESTA [8] using two-dimensional isogenies, which was later accelerated by Nakagawa and Onuki via quaternion algebras through QFESTA [37].

Building on these developments in higher-dimensional public-key encryption, Basso and Maino [7] subsequently introduced POKÉ, a scheme that achieves even greater efficiency by integrating an SIDH-like structure with higher-dimensional isogeny representations. The core of this protocol involves two parties constructing a commutative diagram: one evaluates a non-smooth isogeny via a higher-dimensional representation, while the other computes a smooth-degree isogeny. The shared secret is ultimately derived from a pair of masked image torsion points. By virtue of these design choices, POKÉ substantially outperforms FESTA and QFESTA, positioning itself as one of the most competitive isogeny-based public-key encryption schemes.

Building upon this paradigm, several variants of POKÉ have since been proposed. Kim *et al.* [29] presented INKE, which utilizes the intermediate curve of the commutative diagram as the shared secret. Cai *et al.* [11] proposed PIKE, which leverages bilinear pairing values as the shared secret, thereby enhancing computational efficiency while simultaneously reducing the sizes of both public keys and ciphertexts.

For many isogeny-based cryptographic schemes, such as POKÉ, SQIsign2D, and their variants, the evaluation of *two-dimensional isogeny representations* constitutes the core computational module, whose efficiency directly affects overall protocol performance. The existing algorithm [29, Algorithm 1] for evaluating two-dimensional representations remains inefficient because it requires expensive two-dimensional discrete logarithm computations and torsion-basis generation. To address this bottleneck, this paper aims to develop a fast algorithm for evaluating two-dimensional representations, which avoids these costly operations. This algorithm not only enhances the performance of POKÉ and SQIsign, but also relaxes the constraints on the selection of smaller parameters, especially for POKÉ and its variants.

1.1 Our Contributions

Our contributions are summarized as follows.

– **A new algorithm for the evaluation of two-dimensional representations.** We present an efficient algorithm for the evaluation of two-dimensional representations using the properties of dual isogenies. The central idea of our approach is to

circumvent costly elliptic-curve operations, such as torsion-basis generation and discrete logarithm computations, by exploiting the matrix structure arising from Kani's lemma [28] and the duality. Moreover, the intermediate values required for dual evaluation can be recovered during the $(2,2)$-isogeny chain computation built upon the existing framework in the theta model proposed by Dartois [18, Section 4.4], thereby facilitating extensive data reuse. However, this framework requires the absence of zero components, which cannot be directly applied to the crucial dual gluing step within the two-dimensional representation. To overcome this critical obstruction, we establish Proposition 2 and a specialized gluing evaluation procedure for the points arising in our applications (detailed in Section 3.2). For the generic and splitting cases, we adapt this general theoretical framework from [18] to our two-dimensional representation setting and obtain concrete algorithms. Through these accelerations, the overall computational cost is reduced to approximately that of a single $(2,2)$-isogeny chain, yielding a 67.2% reduction in multiplications over the finite field $\mathbb{F}_p$ compared with previous approaches.

– **Acceleration of isogeny-based cryptographic protocols.** We apply our new algorithm for evaluating two-dimensional representations to isogeny-based cryptosystems, including POKÉ, SQIsign2D, and their variants, achieving efficiency improvements. Specifically, by integrating the *Superglue* [23], we can leverage the twist trick [16]. This naturally relaxes parameter selection constraints and allows our methodology to work well for POKÉ variants with a single point as the shared secret. Furthermore, this optimization technique is also extended to PIKE. We denote the two fast variants of POKÉ based on our proposed approach by POKÉD and PIKED (POKÉ and PIKE using Dual isogenies). For the SQIsign2D variants, we integrate the proposed techniques into the NIST candidate SQIsign-v3.0 and SQIsign2DPush at the algorithmic level, improving the corresponding signing phases.
– **Efficient C implementations of POKÉ and SQIsign2D variants.** We provide optimized C implementations for the POKÉ variants leveraging our new two-dimensional representation evaluation method, alongside concrete parameters targeting the standard NIST I security level. Experimental benchmarks demonstrate that the variants integrating our methodology, namely POKÉD and PIKED, achieve substantial performance improvements. Specifically, at the NIST-I security level, they achieve overall speedups of approximately 3.47× and 3.14× in terms of CPU clock cycles relative to POKÉ and PIKE, respectively. For SQIsign2DPush (resp. SQIsign-v3.0) at the NIST-I, -III, and -V security levels, we report signing-phase improvements of 11.24% (resp. 5.02%), 10.44% (resp. 4.18%), and 10.11% (resp. 4.44%), respectively. Detailed performance results are presented in Section 5. Our optimized C implementation is available at

https://github.com/MicroMicrocosm/2D-Dual-Isogeny.

1.2 Related Work

Dartois [18, Section 4.4] establishes a general framework for evaluating dual isogenies in level-2 theta coordinates using data obtained during the forward computation. The coordinatewise inversion required by these formulas, nevertheless, does not directly recover image coordinates corresponding to vanishing theta-null components. The public-key encryption scheme LIT-SiGamal [34] and the prime-degree PRISM [3,4] variant ECLIPSE [40, Section 4] also reuse intermediate theta-null points to evaluate a two-dimensional dual chain, combining a forward chain, a change of theta structure, and a pull-back. Their pull-back starts from an intermediate abelian surface and ends with the dual of the initial gluing, rather than addressing the dual gluing encountered when reversing a chain between elliptic products.

In contrast to the aforementioned literature, our work concentrates on the two-dimensional representation evaluations building on the framework of dual-evaluation in the theta model, resulting in fast forward–dual and dual–forward algorithms without the time-consuming intermediate torsion-basis generation and discrete logarithms required by previous evaluation approaches. Additionally, we address the additional obstruction mentioned above: for the inputs of the form $(P, \mathcal{O})$ or $(\mathcal{O}, Q)$ arising in our applications, we prove that the otherwise undetermined image coordinate vanishes, thereby obtaining a specialized dual-gluing procedure together with explicit dual generic and splitting algorithms. Furthermore, we integrate these algorithms with Superglue and apply them to POKÉ, PIKE, SQIsign-v3.0, and SQIsign2DPush, including a separate one-dimensional dual optimization of FastCommit. The scope of this paper is therefore orthogonal to the previous work.

1.3 Organization of This Paper

The mathematical preliminaries and definitions are presented in Section 2, including Kani's Lemma, the two-dimensional representation, theta coordinates, the POKÉ scheme, and the SQIsign2D variants. Section 3 introduces our new algorithm for the evaluation of two-dimensional representations. Section 4 applies this method to POKÉ with its variants, SQIsign-v3.0, as well as SQIsign2DPush. The performance results are presented in Section 5. Finally, our conclusions and future work are discussed in Section 6.

2 Preliminaries

This section introduces the mathematical preliminaries required in this paper, including Kani's lemma, theta coordinates, the POKÉ protocol, and the SQIsign2D variants. We first present the notation used throughout this paper.

Notation Let S denote a finite set. The notation $x \xleftarrow{\$} S$ means x is sampled uniformly from S. Denote by KDF a key-derivation function. Let p represent a prime of cryptographic size, i.e., $\log(p) \gtrsim 2\lambda$ bits, where λ is the security parameter. Denote by $\mathcal{O}_E$ the identity element of an elliptic curve E. Let $\mathbf{m}$, $\mathbf{s}$, and $\mathbf{i}$ denote the computational costs of a multiplication, a squaring, and an inversion in $\mathbb{F}_p$, respectively. Denote by $\mathbf{M}$, $\mathbf{S}$, and $\mathbf{I}$ denote the costs of a multiplication, a squaring, and an inversion in $\mathbb{F}_{p^2}$, respectively. Then $\mathbf{M} = 3\mathbf{m}$ and $\mathbf{S} = 2\mathbf{m}$.

In this paper, we consider only supersingular Montgomery curves $E_{A,B} : By^2 = x^3 + Ax^2 + x$ defined over the finite field $\mathbb{F}_{p^2}$, where $p \equiv 3 \bmod 4$. An isogeny $\varphi : E_1 \to E_2$ over $\mathbb{F}_{p^2}$ is a non-constant rational map defined over $\mathbb{F}_{p^2}$, which is also a homomorphism from $E_1(\mathbb{F}_{p^2})$ to $E_2(\mathbb{F}_{p^2})$ [45, III.4]. The degree of φ is its degree as a rational map. Define $\ker(\varphi)$ to be the kernel of φ. An isogeny φ is called *separable* if $\deg(\varphi) = \#\ker(\varphi)$. In this paper, only separable isogenies are considered. We denote an isogeny φ of degree n by an n-isogeny. For each isogeny φ between two elliptic curves E_1 and E_2, there exists another isogeny $\widehat{\varphi} : E_2 \to E_1$ such that $\widehat{\varphi} \circ \varphi = \varphi \circ \widehat{\varphi} = [\deg(\varphi)]$, called the *dual isogeny* of φ.

Let $\varphi : E_1 \to E_2$ and $\psi : E_1 \to E_3$ denote two isogenies of coprime degrees. We call an isogeny $\psi' : E_2 \to E_4$ the *pushforward* of ψ by φ if $\ker(\psi') = \varphi(\ker(\psi))$, and denote it by $\psi' = [\varphi]_*\psi$. On the other hand, the isogeny ψ is called the *pullback* of ψ' by φ, which is denoted by $\psi = [\varphi]^*\psi'$.

The n-torsion group of an elliptic curve E over $\mathbb{F}_{p^2}$ is a subgroup of $E(\overline{\mathbb{F}_{p^2}})$ defined as

$$E[n] = \{P \in E\left(\overline{\mathbb{F}_{p^2}}\right) \mid [n]P = \mathcal{O}_E\}.$$

It holds that $E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n$ for n such that $\gcd(p, n) = 1$.

2.1 Kani's Lemma and Two-Dimensional Representation

In this subsection, we introduce Kani's Lemma, which is widely used to construct high-dimensional cryptographic schemes by embedding the one-dimensional non-smooth isogenies between elliptic curves into high-dimensional smooth isogenies between principally polarized abelian varieties (PPAVs).

Theorem 1 (Kani's Lemma [28]). *Given the following isogeny diamond*

$$\begin{array}{ccc} E_0 & \xrightarrow{\varphi_1} & E_1 \\ {\scriptstyle\varphi_2}\downarrow & & \downarrow{\scriptstyle\varphi_2'} \\ E_2 & \xrightarrow[\varphi_1']{} & E_3 \end{array}$$

where $\deg(\varphi_1) = \deg(\varphi_1') = d_1, \deg(\varphi_2) = \deg(\varphi_2') = d_2$, *and* $\gcd(d_1, d_2) = 1$. *Then the following isogeny*

$$\Phi = \begin{pmatrix} \varphi_1 & -\widehat{\varphi_2'} \\ \varphi_2 & \widehat{\varphi_1'} \end{pmatrix} : E_0 \times E_3 \to E_1 \times E_2$$

is a two-dimensional $(d_1 + d_2, d_1 + d_2)$*-isogeny with kernel*

$$\ker(\Phi) = \{([-d_1]P, \varphi_2' \circ \varphi_1(P)) \mid P \in E_0[d_1 + d_2]\}.$$

In practice, we prefer to take $d_1 + d_2 = 2^n$, and evaluate a point $P \in E_0$ under φ_1 and φ_2 as follows:

$$\begin{pmatrix} \varphi_1(P) \\ \varphi_2(P) \end{pmatrix} = \Phi \begin{pmatrix} P \\ \mathcal{O}_{E_3} \end{pmatrix}.$$

By evaluating a smooth $(2^n, 2^n)$-isogeny, one can efficiently derive $\varphi_1(P)$ or $\varphi_2(P)$.

The isogeny $\varphi : E \to E'$ *admits a two-dimensional representation*

$$(q, E, E', (P_1, P_2), (\varphi(P_1), \varphi(P_2)))$$

if $\deg(\varphi) = q(2^a - q), q < 2^a, \gcd(q, 2) = 1$ and $\langle P_1, P_2 \rangle = E[2^a]$.

2.2 Theta Coordinates and (2,2)-Isogenies

In this subsection, we recall the definition and fundamental properties of theta coordinates, which serve as a powerful tool for $(2, 2)$-isogeny computations. We refer the reader to [20, Section 2] for a more comprehensive treatment.

Let k be a field with $\mathrm{char}(k) \neq 2$, and let A be a g-dimensional abelian variety defined over k. Let $\mathcal{L}$ be a symmetric line bundle on A, and denote by T_P the translation-by-P map on A. We define $K(\mathcal{L})$ as the subgroup of points $P \in A$ for which the pullback $T_P^*\mathcal{L}$ is isomorphic to $\mathcal{L}$

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

The set $\mathcal{G}(\mathcal{L})$, defined as

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

admits a natural group structure, known as the *Mumford theta group*. It fits into the following short exact sequence:

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

Furthermore, suppose there exist two maximal isotropic subgroups such that $K(\mathcal{L}) = K(\mathcal{L})_1 \oplus K(\mathcal{L})_2$. Consider a sequence of elementary divisors $\delta = (d_1, \ldots, d_g)$ such that

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

A *theta structure of type* δ is an isomorphism of groups $\Theta^{\mathcal{L}} : \mathcal{G}(\delta) \xrightarrow{\sim} \mathcal{G}(\mathcal{L})$, where $\mathcal{G}(\delta)$ is the Heisenberg group associated with δ. The space $V(\delta) = \{g : K(\delta) \to \bar{k}\}$ is the unique irreducible representation of $\mathcal{G}(\delta)$ on which $\bar{k}^*$ acts by multiplication. Similarly, the space of global sections $\Gamma(A, \mathcal{L})$ is the unique irreducible $\mathcal{G}(\mathcal{L})$-representation. Consequently, there exists an isomorphism β between $\Gamma(A, \mathcal{L})$ and $V(\delta)$. A basis $(\theta_i)_{i \in K(\mathcal{L})_1}$ of $\Gamma(A, \mathcal{L})$, derived via this isomorphism from the Kronecker delta basis $(\delta_i)_{i \in K(\delta)}$ of $V(\delta)$, provides the *theta coordinates*. If $d_1 = \cdots = d_g = n$, these are referred to as theta coordinates of *level* n.

For the purposes of this paper, we restrict our focus to level-2 theta coordinates and assume henceforth that A is a two-dimensional abelian surface. In this setting, the theta coordinates on A take the form

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

The projective point $(\theta_i^A(0))_{i \in K(2,2)}$ is called the *theta-null point*. Let $S_1, S_2 \in A[2]$ and $T_1, T_2 \in A[2]$ be the points derived from the canonical bases of $K(2,2)$ and $\mathrm{Hom}(K(2,2), \bar{k}^*)$, respectively, via the isomorphism $\overline{\Theta}^{\mathcal{L}} : H(2,2) \xrightarrow{\sim} K(\mathcal{L}) = A[2]$ induced by $\Theta^{\mathcal{L}}$. The theta structure Θ^A is then determined by a symplectic 4-torsion basis $S_1', S_2', T_1', T_2' \in A[4]$, where S_i' and T_i' lie above S_i and T_i, respectively [35, Remark 3, p. 319]. Given a point $P \in A$ and a 2-torsion point $T \in K(\mathcal{L})$ corresponding to $(s, \chi) \in H(2,2)$, the translation $P + T$ can be represented in theta coordinates as

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

Hadamard transform and duplication formula We now introduce the Hadamard transform and the duplication formula, which form the computational core of $(2,2)$-isogeny evaluations.

Let $\mathcal{H}$ denote the Hadamard transform acting on a theta coordinate tuple $(\theta_i)_i$. The resulting image coordinates are called the *dual theta coordinates*, denoted by $(\widetilde{\theta}_i)_i$. For an abelian surface, a point P in theta coordinates is represented as $P = (\theta_{00} : \theta_{10} : \theta_{01} : \theta_{11})$[4] $= (x : y : z : t)$. The action of $\mathcal{H}$ is defined as

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

Note that $\mathcal{H}$ is an involution up to a scalar factor, meaning $\mathcal{H}(\mathcal{H}((\theta_i(P))_i)) \sim (\theta_i(P))_i$ projectively for any P.

Recall from Section 2.2 that A is an abelian surface endowed with a type-$(2,2)$ theta structure induced by the symplectic 4-torsion basis $\langle S_1', S_2', T_1', T_2' \rangle$. Consider a $(2,2)$-isogeny $f : A \to B$ with $\ker(f) = K(\mathcal{L})_2 = \langle T_1, T_2 \rangle$. Let $\star$ denote the Hadamard multiplication operator, such that $(x_i)_i \star (y_i)_i = (x_i y_i)_i$. The *duplication formula* [35,41] then states

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

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

[4] The subscript indexing order follows the convention used in [20, Section 1.1].

where $\tilde{f}$ is the dual isogeny of f.

2.3 The POKÉ Protocol

We now introduce the POKÉ [7] protocol, which is based on an SIDH-like commutative diagram. Let $p = 2^a 3^b 5^c f - 1$ be a prime, where f is a small cofactor. Let $E_0/\mathbb{F}_{p^2} : y^2 = x^3 + x$ denote a supersingular curve with j-invariant 1728 and a known endomorphism ring $\mathrm{End}(E_0) \cong \mathbb{Z}\left\langle 1, i, \frac{i+j}{2}, \frac{1+ij}{2} \right\rangle$, where i corresponds to the automorphism $\iota : (x, y) \mapsto (-x, \sqrt{-1} \cdot y)$, and j corresponds to the Frobenius map $\pi : (x, y) \mapsto (x^p, y^p)$. Denote by $\{P_0, Q_0\}$, $\{R_0, S_0\}$ and $\{X_0, Y_0\}$ the bases of $E_0[2^a]$, $E_0[3^b]$ and $E_0[5^c]$, respectively.

Key generation The receiver uniformly samples an integer $q \in [0, 2^a)$ with $\gcd(q, 30) = 1$, and uses a rigorous or heuristic algorithm (see [7, Algorithms 2 and 3] for details) to compute an isogeny $\varphi : E_0 \to E_A$ of degree $q(2^a - q)$. The receiver then evaluates φ at the bases $\{P_0, Q_0\}$, $\{R_0, S_0\}$ and $\{X_0, Y_0\}$, and masks these image points using 2×2 scaling matrices to obtain $\{P_A, Q_A\}$, $\{R_A, S_A\}$ and $\{X_A, Y_A\}$ on E_A, respectively. The public key consists of $E_A, P_A, Q_A, R_A, S_A, X_A, Y_A$. Algorithm 1 summarizes the key generation of POKÉ.

Algorithm 1 Key Generation of POKÉ

Input: $E_0, P_0, Q_0 \in E_0[2^a], R_0, S_0 \in E_0[3^b], X_0, Y_0 \in E_0[5^c]$.
Output: A secret/public key pair.

1: Sample a random $q \in [0, 2^a)$ such that $\gcd(q(2^a - q), 30) = 1$.
2: Generate a $q(2^a - q)$-isogeny $\varphi = \varphi_2 \circ \varphi_1 : E_0 \to E_A$. ▷ Using [7, Algorithm 2] or [7, Algorithm 3]
3: $\alpha_2, \beta_2 \xleftarrow{\$} \mathbb{Z}_{2^a}^{\times}$, $\gamma_3 \xleftarrow{\$} \mathbb{Z}_{3^b}^{\times}$, $\delta_5 \xleftarrow{\$} \mathbb{Z}_{5^c}^{\times}$.
4: $P_A, Q_A \leftarrow [\alpha_2]\varphi(P_0), [\beta_2]\varphi(Q_0)$.
5: $R_A, S_A \leftarrow [\gamma_3]\varphi(R_0), [\gamma_3]\varphi(S_0)$.
6: $X_A, Y_A \leftarrow [\delta_5]\varphi(X_0), [\delta_5]\varphi(Y_0)$.
7: **return** $\mathsf{sk} = (q, \alpha_2, \beta_2, \delta_5)$, $\mathsf{pk} = (E_A, P_A, Q_A, R_A, S_A, X_A, Y_A)$.

Encryption The sender uniformly samples $r_3 \xleftarrow{\$} \mathbb{Z}_{3^b}$ and computes two parallel 3^b-isogenies $\psi : E_0 \to E_B$ and $\psi' : E_A \to E_{AB}$ with kernels $\ker(\psi) = \langle R_0 + [r_3]S_0 \rangle$ and $\ker(\psi') = \langle R_A + [r_3]S_A \rangle$, respectively. Subsequently, the sender evaluates ψ and ψ' at the respective bases $\{P_0, Q_0\}$ and $\{P_A, Q_A\}$, as well as at $\{X_0, Y_0\}$ and $\{X_A, Y_A\}$. The sender then scales these bases diagonally to derive the shared secret X_{AB}, Y_{AB}. Finally, the sender computes the ciphertext $\mathsf{ct} = m \oplus \mathrm{KDF}(X_{AB}, Y_{AB})$. The concrete encryption procedure is illustrated in Algorithm 2.

Decryption The receiver uses the scaling values α_2, β_2 to compute P'_{AB}, Q'_{AB}, and evaluates the two-dimensional isogeny Φ' to obtain the shared secret X_{AB}, Y_{AB}. The receiver then extracts the message from ct as $m = \mathsf{ct}' \oplus \mathrm{KDF}(X_{AB}, Y_{AB})$. The decryption phase of POKÉ is shown in Algorithm 3.

Alternative shared secret points As discussed in [7, Section 4.4], one can alternatively utilize a single point, rather than a full basis, as the shared secret. This approach allows for a smaller parameter p (allowing the shared secret point to lie on the quadratic twist) and consequently reduces both public key and ciphertext sizes. However, if this shared secret point has non-smooth order, the protocol requires the computation of two $(2, 2)$-isogeny chains, which dominates the computational cost. The method we propose in Section 3

Algorithm 2 Encryption of POKÉ

Input: A message m, a public key $E_A, P_A, Q_A, R_A, S_A, X_A, Y_A$.
Output: A ciphertext ct.

$r_3 \xleftarrow{\$} \mathbb{Z}_{3^b}$, $\omega_2 \xleftarrow{\$} \mathbb{Z}_{2^a}^{\times}$, $D_5 \xleftarrow{\$} \mathrm{SL}_2(\mathbb{Z}_{5^c})$.
Compute the isogeny $\psi : E_0 \to E_B$ with kernel $\langle R_0 + [r_3]S_0 \rangle$.
Compute the isogeny $\psi' : E_A \to E_{AB}$ with kernel $\langle R_A + [r_3]S_A \rangle$.
$P_B, Q_B \leftarrow [\omega_2]\psi(P_0), [1/\omega_2]\psi(Q_0)$.
$P_{AB}, Q_{AB} \leftarrow [\omega_2]\psi'(P_A), [1/\omega_2]\psi'(Q_A)$.
Compute $[X_B, Y_B]^T = D_5 \begin{bmatrix} \psi(X_0) \\ \psi(Y_0) \end{bmatrix}$.
Compute $[X_{AB}, Y_{AB}]^T = D_5 \begin{bmatrix} \psi'(X_A) \\ \psi'(Y_A) \end{bmatrix}$.
$\mathsf{ct}' \leftarrow \mathrm{KDF}(X_{AB}, Y_{AB}) \oplus m$.
return $\mathsf{ct} = ((E_B, P_B, Q_B, X_B, Y_B), (E_{AB}, P_{AB}, Q_{AB}), \mathsf{ct}')$.

Algorithm 3 Decryption of POKÉ

Input: $\mathsf{sk} = (q, \alpha_2, \beta_2, \delta_5)$, $\mathsf{ct} = ((E_B, P_B, Q_B, X_B, Y_B), (E_{AB}, P_{AB}, Q_{AB}), \mathrm{ct}')$.
Output: A message m'.

$P'_{AB}, Q'_{AB} \leftarrow [1/\alpha_2]P_{AB}, [1/\beta_2]Q_{AB}$.
Compute Φ' with kernel $\langle ([-q]P_B, P'_{AB}), ([-q]Q_B, Q'_{AB}) \rangle$.
$(X'_B, _) \leftarrow \Phi'(X_B, \mathcal{O}_{E_{AB}})$, $(Y'_B, _) \leftarrow \Phi'(Y_B, \mathcal{O}_{E_{AB}})$.
Generate a basis U, V of $E_{AB}[5^c]$.
$(U', _) \leftarrow \Phi'(\mathcal{O}_{E_B}, U)$, $(V', _) \leftarrow \Phi'(\mathcal{O}_{E_B}, V)$.
Find x, y such that $X'_B = [x]U' + [y]V'$.
Find w, z such that $Y'_B = [w]U' + [z]V'$.
$X_{AB} \leftarrow [\delta_5][2^a - q]([x]U + [y]V)$.
$Y_{AB} \leftarrow [\delta_5][2^a - q]([w]U + [z]V)$.
$m' \leftarrow \mathrm{KDF}(X_{AB}, Y_{AB}) \oplus \mathrm{ct}'$.
return m'.

successfully addresses this bottleneck, yielding substantial performance improvements for POKÉ variants that rely on a single-point shared secret (we refer to Section 4 for more details).

To instantiate POKÉ with a single point as the shared secret, the parameter selection must be carefully adjusted to ensure λ-bit security. Specifically, to recover the shared secret point X_{AB}, an adversary can proceed as follows:

- Compute a basis $U_{AB}, V_{AB} \in E_{AB}[D]$, since the curve E_{AB} is public knowledge;
- Perform an exhaustive search over the coefficients $x_{AB}, y_{AB} \in \mathbb{Z}_D^{\times}$ such that $[x_{AB}]U_{AB} + [y_{AB}]V_{AB} = X_{AB}$.

Consequently, the adversary must determine the two coefficients $x_{AB}, y_{AB} \in \mathbb{Z}_D^{\times}$. To preclude this brute-force attack and achieve the target security level, one must enforce the condition $D \gtrsim 2^{\lambda/2}$.

2.4 Overview of the SQIsign2D Variants

In this subsection, we briefly review several variants of SQIsign that leverage two-dimensional isogenies, specifically SQIsign-v3.0 [1] and SQIsign2DPush [38].

SQIsign2D-West [5] and SQIsign2D-East [39] are the optimized variants of the high-dimensional signature scheme SQIsignHD [19]. By employing two-dimensional isogenies for verification instead of the four- or eight-dimensional isogenies required in SQIsignHD, these schemes achieve a substantial verification speedup, albeit at the expense of a less efficient

signing phase due to the computation of auxiliary isogenies. Notably, SQIsign2D-West has been incorporated into Round 3 of the NIST Post-Quantum Cryptography standardization process under the name SQIsign-v3.0. Furthermore, SQIsign2DPush is an improved iteration of SQIsign2D-East. For simplicity, we concentrate on SQIsign2DPush instead of SQIsign2D-East in this paper. All SQIsign2D variants share a similar algorithmic framework, as illustrated by the commutative diagram in Figure 1.

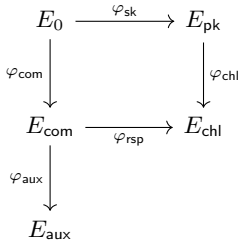


Figure 1: The commutative diagram for SQIsign2D variants.

The execution flow of the underlying Σ-protocol for the SQIsign2D variants is summarized as follows. Let φ_{sk} denote the prover's secret isogeny. First, the prover generates a random commitment isogeny $\varphi_{\mathsf{com}} : E_0 \to E_{\mathsf{com}}$ and transmits it to the verifier. Upon receipt, the verifier samples a random challenge isogeny $\varphi_{\mathsf{chl}} : E_{\mathsf{pk}} \to E_{\mathsf{chl}}$ and sends it to the prover. The prover then computes the response isogeny $\varphi_{\mathsf{rsp}} : E_{\mathsf{com}} \to E_{\mathsf{chl}}$ alongside an auxiliary isogeny φ_{aux}, sending both to the verifier. Finally, the verifier checks whether φ_{rsp} is a valid isogeny from E_{com} to E_{chl} with the assistance of φ_{aux}. For some variants, such as SQIsign2D^2 [47], the roles of the secret isogeny and the commitment isogeny in the protocol diagram are interchanged.

As detailed in Table 1, the primary distinction among these SQIsign2D variants lies in the specific algorithms employed to generate φ_{sk}, φ_{com}, and φ_{aux}. The procedures FastDoublePath and FastCommit are structurally similar; however, FastDoublePath computes both 2- and 3-isogeny chains, whereas FastCommit strictly constructs a 3-isogeny chain. Notably, the algorithmic building blocks IdealToIsogeny and PushRandIsog rely heavily on the computation and evaluation of $(2, 2)$-isogenies, making them computationally dominant within the protocol.

Table 1: The fundamental building blocks of each variant of SQIsign2D to generate φ_{sk}, φ_{com} and φ_{aux}.

Scheme	φ_{sk}	φ_{com}	φ_{aux}
SQIsign-v3.0	IdealToIsogeny [5, Algorithm 2]	IdealToIsogeny	IdealToIsogeny
SQIsign2DPush	FastDoublePath [19, Algorithm 1]	FastCommit [19, Algorithm 1]	PushRandIsog [38, Algorithm 2]

PushRandIsog The PushRandIsog algorithm provides a streamlined mechanism for computing an auxiliary isogeny of a fixed, generally non-smooth degree d from a target curve E. Given a large-degree isogeny $\psi : E_0 \to E$ (e.g., a 3^{2e_2}-isogeny), the algorithm initiates by

generating an initial d-isogeny φ_0 originating from the starting curve E_0. By methodically decomposing ψ into two sequential 3^{e_2}-isogenies, ψ_0 and ψ_m, PushRandIsog computes the push-forward of φ_0 along the path of ψ. Using the KaniEval subroutine based on Kani's Lemma, the algorithm systematically propagates the isogeny evaluations across the commutative diagram. This technique efficiently yields the domain, codomain, and torsion evaluations of the target d-isogeny derived from E, minimizing the reliance on full ideal translations during the response phase.

FastCommit In the setting of SQIsign2DPush, with $p = c \cdot 2^{e_1}3^{e_2} - 1$ for a cofactor c, the algorithm outputs a 3^{2e_2}-isogeny $\psi : E_0 \to E$, together with its corresponding left $\mathcal{O}_0$-ideal J.

3 Fast Evaluation of Two-Dimensional Representations

In this section, we propose a new algorithm for the efficient evaluation of two-dimensional representations. Specifically, our method accelerates the evaluation of points under isogenies of degree $q(2^a - q)$, where $\gcd(q, 2) = 1$. Most notably, this algorithmic optimization yields significant performance improvements, substantially accelerating both the key generation and decryption phases of the POKÉ scheme. It also benefits the signing procedure of the SQIsign2D variants.

3.1 A New Algorithm for Evaluating Two-Dimensional Representations

The evaluation of two-dimensional representations is a fundamental building block of many cryptographic schemes, such as SQIsign and POKÉ. Both schemes rely on the same commutative diagram shown in Figure 2, but the two-dimensional representations to be evaluated are different. In SQIsign-v3.0 [1], the curves E_1 and E_2 are known and the target is to evaluate $\varphi_1 \circ \widehat{\varphi_2} : E_2' \to E_1'$. In POKÉ [7], the curves E_1 and E_2 are also given and the goal is $\varphi_2' \circ \varphi_1 : E_1 \to E_2$. In this section, we focus on the POKÉ case; the SQIsign case can be handled analogously.

$$\begin{array}{ccc} E_1 & \xrightarrow{\varphi_1} & E_1' \\ {\scriptstyle\varphi_2}\downarrow & \circlearrowright & \downarrow{\scriptstyle\varphi_2'} \\ E_2' & \xrightarrow[\varphi_1']{} & E_2 \end{array}$$

Figure 2: The commutative diagram for two-dimensional representations.

For POKÉ, the previous method [29, Algorithm 1] and its variants first compute a $(2^a, 2^a)$-isogeny $\Phi : E_1 \times E_2 \to E_1' \times E_2'$ with kernel $\ker(\Phi) = \{([-q]P, \varphi_2' \circ \varphi_1(P)) \mid P \in E_1[2^a]\}$. By Theorem 1, it follows that one can evaluate Φ at $(Q, \mathcal{O}_{E_2}) \in E_1 \times E_2$ to derive $\varphi_1(Q)$ as

$$\begin{pmatrix} \varphi_1(Q) \\ * \end{pmatrix} = \begin{pmatrix} \varphi_1 & -\widehat{\varphi_2'} \\ \varphi_2 & \widehat{\varphi_1'} \end{pmatrix} \begin{pmatrix} Q \\ \mathcal{O}_{E_2} \end{pmatrix}.$$

If the order $\mathrm{ord}(Q) = D$ is smooth, then we can generate a basis $\langle U, V \rangle = E_2[D]$ and evaluate its points under Φ to obtain points on E_1' as follows:

$$\begin{pmatrix}\widehat{\varphi_2'}(U)\\ *\end{pmatrix} = \begin{pmatrix}\varphi_1 & -\widehat{\varphi_2'}\\ \varphi_2 & \widehat{\varphi_1'}\end{pmatrix}\begin{pmatrix}\mathcal{O}_{E_1}\\ -U\end{pmatrix}, \quad \begin{pmatrix}\widehat{\varphi_2'}(V)\\ *\end{pmatrix} = \begin{pmatrix}\varphi_1 & -\widehat{\varphi_2'}\\ \varphi_2 & \widehat{\varphi_1'}\end{pmatrix}\begin{pmatrix}\mathcal{O}_{E_1}\\ -V\end{pmatrix}.$$

In practice, to leverage the x-only arithmetic, one should also evaluate Φ at $(\mathcal{O}_{E_1}, V-U)$. By solving the two-dimensional discrete logarithm, one can derive the coordinates u, v of $\varphi_1(Q)$ with respect to $\widehat{\varphi_2'}(U)$ and $\widehat{\varphi_2'}(V)$, and the resulting point $\varphi_2' \circ \varphi_1(Q)$ can be computed as $\varphi_2'\left([u]\widehat{\varphi_2'}(U) + [v]\widehat{\varphi_2'}(V)\right) = [2^a - q]([u]U + [v]V)$. The two-dimensional discrete logarithm computation and basis generation entail a certain computational overhead. Moreover, the discrete logarithm cannot be efficiently computed if D is non-smooth. Under these circumstances, the existing countermeasure [7, Remark 6] suggests computing and evaluating two high-dimensional isogenies to extract $\varphi_2' \circ \varphi_1(Q)$, which dominates the computational cost.

To address the aforementioned efficiency bottlenecks, our novel approach leverages the property of dual isogenies and enables the evaluation of two-dimensional representations using essentially only a single two-dimensional isogeny computation. Furthermore, this method does not depend on whether $\mathrm{ord}(Q) = D$ is smooth, providing more flexibility in the choice of parameters for POKÉ. The same strategy can be adapted to the SQIsign case by considering the dual isogeny $\widehat{\varphi_1}$ and swapping the roles of the curves accordingly. Algorithm 4 illustrates our new method for the POKÉ case.

Algorithm 4 $\mathsf{ForwardDual}(Q, q, P, \varphi_2' \circ \varphi_1(P))$

Input: A point $Q \in E_1(\mathbb{F}_{p^2})$ of order D with $\gcd(D, q(2^a - q)) = 1$. The 2^a-torsion points $P \in E_1[2^a]$ and their images $\varphi_2' \circ \varphi_1(P) \in E_2[2^a]$.
Output: The image point $\varphi_2' \circ \varphi_1(Q)$.
1: Compute Φ with kernel $\ker(\Phi) = \{([-q]P, \varphi_2' \circ \varphi_1(P)) \mid P \in E_1[2^a]\}$.
2: Evaluate $(\varphi_1(Q), _) = \Phi(Q, \mathcal{O}_{E_2})$.
3: Evaluate $(_, \varphi_2' \circ \varphi_1(Q)) = \widehat{\Phi}(-\varphi_1(Q), \mathcal{O}_{E_2'})$ with kernel $\ker(\widehat{\Phi}) = \{(\varphi_1(P), \varphi_2(P)) \mid P \in E_1[2^a]\}$. ▷ See Algorithm 8 for further details
4: **return** $\varphi_2' \circ \varphi_1(Q)$.

We establish the correctness of the algorithm in Proposition 1.

Proposition 1. *Algorithm 4 is correct.*

Proof. Consider the dual of Φ:

$$\widehat{\Phi} = \begin{pmatrix}\widehat{\varphi_1} & \widehat{\varphi_2}\\ -\varphi_2' & \varphi_1'\end{pmatrix} : E_1' \times E_2' \to E_1 \times E_2.$$

A simple calculation shows that

$$\ker(\widehat{\Phi}) = \{([q - 2^a]P_1, \varphi_2 \circ \widehat{\varphi_1}(P_1)) \mid P_1 \in E_1'[2^a]\}.$$

Taking $P_1 = \varphi_1(P)$, from $\gcd(q, 2^a) = 1$ we deduce

$$\varphi_1(E_1[2^a]) = E_1'[2^a],$$

and thus

$$\begin{aligned}\ker(\widehat{\Phi}) &= \{([q - 2^a]\varphi_1(P), \varphi_2 \circ \widehat{\varphi_1} \circ \varphi_1(P)) \mid P \in E_1[2^a]\}\\ &= \{([q]\varphi_1(P), [q]\varphi_2(P)) \mid P \in E_1[2^a]\}\\ &= \{(\varphi_1(P), \varphi_2(P)) \mid P \in E_1[2^a]\}.\end{aligned}$$

Since the matrix of $\widehat{\Phi}$ contains the information needed to evaluate φ_2', we can extract $\varphi_2' \circ \varphi_1(Q)$ as follows:

$$\begin{pmatrix} * \\ \varphi_2' \circ \varphi_1(Q) \end{pmatrix} = \begin{pmatrix} \widehat{\varphi_1} & \widehat{\varphi_2} \\ -\varphi_2' & \varphi_1' \end{pmatrix} \begin{pmatrix} -\varphi_1(Q) \\ \mathcal{O}_{E_2'} \end{pmatrix}.$$

The above operations correspond to Line 3 of Algorithm 4, which completes the proof. □

Analogously, we present Algorithm 5 for the response phase of SQIsign-v3.0; see Section 4.2 for details.

Algorithm 5 DualForward$(Q, q, P, \varphi_2' \circ \varphi_1(P))$

Input: A point $Q \in E_2'(\mathbb{F}_{p^2})$ of order D with $\gcd(D, q(2^a - q)) = 1$. The 2^a-torsion points $P \in E_1[2^a]$ and their images $\varphi_2' \circ \varphi_1(P) \in E_2[2^a]$.
Output: The image point $\varphi_1 \circ \widehat{\varphi_2}(Q)$.
1: Compute Φ with kernel $\ker(\Phi) = \{([-q]P, \varphi_2' \circ \varphi_1(P)) \mid P \in E_1[2^a]\}$.
2: Evaluate $(\widehat{\varphi_2}(Q),\ _) = \widehat{\Phi}(\mathcal{O}_{E_1'},\ Q)$, with kernel $\ker(\widehat{\Phi}) = \{(\varphi_1(P), \varphi_2(P)) \mid P \in E_1[2^a]\}$.
▷ See Algorithm 8 for further details
3: Evaluate $(\varphi_1 \circ \widehat{\varphi_2}(Q),\ _) = \Phi(\widehat{\varphi_2}(Q),\ \mathcal{O}_{E_2})$.
4: **return** $\varphi_1 \circ \widehat{\varphi_2}(Q)$.

3.2 Algorithmic Details of Our Method

From Algorithms 4 and 5 it seems that we need to compute two high-dimensional isogenies $\Phi : E_1 \times E_2 \to E_1' \times E_2'$ and $\widehat{\Phi} : E_1' \times E_2' \to E_1 \times E_2$. In fact, we can reuse the information within the computation of Φ while evaluating the dual $\widehat{\Phi}$ at $(-\varphi_1(Q), \mathcal{O}_{E_2'})$, resulting in marginal additional overhead (see Table 2 for more details). Moreover, our method does not even require knowledge of $\ker(\widehat{\Phi})$, thereby obviating the evaluation of Φ at the point $(P, \mathcal{O}_{E_2})$.

The $(2^a, 2^a)$-isogenies Φ and $\widehat{\Phi}$ can be decomposed into a $(2,2)$-isogenies as follows:

$$\Phi : E_1 \times E_2 \xrightarrow{f_g} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} \cdots \xrightarrow{f_{a-2}} A_{a-1} \xrightarrow{f_s} E_1' \times E_2',$$
$$\widehat{\Phi} : E_1' \times E_2' \xrightarrow{\widehat{f_s}} A_{a-1} \xrightarrow{\widehat{f_{a-2}}} A_{a-2} \xrightarrow{\widehat{f_{a-3}}} \cdots \xrightarrow{\widehat{f_1}} A_1 \xrightarrow{\widehat{f_g}} E_1 \times E_2,$$

where the A_i are two-dimensional abelian surfaces, and f_i, f_g, and f_s denote the generic, gluing, and splitting isogenies, respectively. The notation $\widehat{f_i}$ represents the dual of the $(2,2)$-isogeny f_i. We note that a more general higher-dimensional method for computing dual isogenies is given in [18, Section 4.4]. It requires the theta-null point to have no zero component, and hence applies to the duals of generic and splitting isogenies but not to gluing. In this subsection, we specialize this framework to our setting and present the algorithmic details.

Assume that Φ has been computed. We now investigate how to evaluate a point on $E_1' \times E_2'$ via the isogeny chain of $\widehat{\Phi}$, considering three distinct cases: generic, gluing, and splitting isogenies. For efficiency, we use the theta model [20] (which is introduced in Section 2.2) for the computation of $(2,2)$-isogenies.

Gluing isogenies of $\widehat{\Phi}$ The gluing isogeny $\widehat{f_s} : E_1' \times E_2' \to A_{a-1}$ of $\widehat{\Phi}$ naturally decomposes into two components

$$\widehat{f_{s1}} : E_1' \times E_2' \to A_{E_1' \times E_2'}, \quad \widehat{f_{s2}} : A_{E_1' \times E_2'} \to A_{a-1},$$

where $\widehat{f_{s1}}$ and $\widehat{f_{s2}}$ are the duals of the splitting isomorphism $f_{s1} : A_{E_1' \times E_2'} \to E_1' \times E_2'$ and the isogeny $f_{s2} : A_{a-1} \to A_{E_1' \times E_2'}$, respectively. Specifically, f_{s2} constitutes the final $(2,2)$-isogeny of Φ, whereas f_{s1} acts as an isomorphism mapping the abelian surface $A_{E_1' \times E_2'}$ onto the elliptic product $E_1' \times E_2'$, thereby retrieving the explicit curve equations for E_1' and E_2'.

Evaluation of $\widehat{f_{s1}}$ The relation

$$f_{s2} \circ f_{s1} \circ \widehat{f_{s1}} \circ \widehat{f_{s2}} = [2]$$

indicates that $\widehat{f_{s1}}$ serves as the inverse isomorphism of f_{s1}. Let $U_{j,k}$, $j,k \in (\mathbb{Z}/2\mathbb{Z})^2$ denote the level-$(2,2)$ theta coordinates, which are derived from the given level-2 theta coordinates $\left(\theta_j^{A_i}(Q_i)\right)_j$ via

$$U_{j,k}(Q_i) = \sqrt{\sum_{t \in (\mathbb{Z}/2\mathbb{Z})^2} (-1)^{j \cdot t^T} \theta_t^{A_i}(Q_i) \theta_{t+k}^{A_i}(Q_i)} \,.$$

As outlined in [20, Section 4.1], the splitting map f_{s1} is decomposed into two successive steps:

(1) Compute a symplectic automorphism ψ, associated with a 4×4 unitary matrix N_ψ, that maps the index (j,k) to $(11,11)$ such that $U_{11,11}\left(0_{A_{E_1' \times E_2'}}\right) = 0$. Consequently, the transformation

$$N_\psi \cdot \left(\theta_j^{A_{E_1' \times E_2'}}(R)\right)_j^T = N_\psi \cdot \left(\theta_{00}^{A_{E_1' \times E_2'}}(R), \theta_{10}^{A_{E_1' \times E_2'}}(R), \theta_{01}^{A_{E_1' \times E_2'}}(R), \theta_{11}^{A_{E_1' \times E_2'}}(R)\right)^T$$

yields the product theta coordinates [20, Proposition 3].

(2) Suppose $N_\psi \cdot \left(\theta_j^{A_{E_1' \times E_2'}}(R)\right)_j^T = (x,y,z,w)^T$. The pairs $(x : y)$ and $(y : w)$ yield the theta coordinates of $R_1' \in E_1'$ and $R_2' \in E_2'$, respectively, where $f_{s_1}(R) = R' = (R_1', R_2')$. Finally, convert these theta coordinates into the Montgomery model via

$$R_1' = \begin{pmatrix} x_1 \\ z_1 \end{pmatrix} = N_1 \begin{pmatrix} x \\ y \end{pmatrix}, \quad R_2' = \begin{pmatrix} x_2 \\ z_2 \end{pmatrix} = N_2 \begin{pmatrix} y \\ w \end{pmatrix},$$

where $N_1 = \begin{pmatrix} b_1 & a_1 \\ -b_1 & a_1 \end{pmatrix}$ and $N_2 = \begin{pmatrix} b_2 & a_2 \\ -b_2 & a_2 \end{pmatrix}$ represent the conversion matrices for the Montgomery curves E_1' and E_2', with $(a_1 : b_1)$ and $(a_2 : b_2)$ being their theta-null points, respectively [43, Appendix A.1].

Therefore, to recover the evaluation of a given point $R' = (R_1', R_2') = \big((x_1, z_1), (x_2, z_2)\big) \in E_1' \times E_2'$ under $\widehat{f_{s1}}$, we perform the following operations:

(1) Utilize the theta-null points of E_1' and E_2' derived from f_{s1}, construct the (projective) inverse conversion matrices $\widetilde{N_1} = \begin{pmatrix} a_1 & -a_1 \\ b_1 & b_1 \end{pmatrix}$ and $\widetilde{N_2} = \begin{pmatrix} a_2 & -a_2 \\ b_2 & b_2 \end{pmatrix}$. The theta coordinates of R_1' and R_2' are then computed as

$$\begin{pmatrix} \theta_0^{E_1'}(R_1') \\ \theta_1^{E_1'}(R_1') \end{pmatrix} = \widetilde{N_1} \begin{pmatrix} x_1 \\ z_1 \end{pmatrix}, \quad \begin{pmatrix} \theta_0^{E_2'}(R_2') \\ \theta_1^{E_2'}(R_2') \end{pmatrix} = \widetilde{N_2} \begin{pmatrix} x_2 \\ z_2 \end{pmatrix}.$$

(2) Compute the conjugate transpose $\widetilde{N_\psi}$ of N_ψ (which acts as its exact inverse due to unitarity), and extract the theta coordinates of $\widehat{f_{s1}}(R') \in A_{E_1' \times E_2'}$ via

$$\left(\theta_j^{A_{E_1' \times E_2'}}(\widehat{f_{s1}}(R'))\right)_j^T = \widetilde{N_\psi} \cdot \left(\left(\theta_0^{E_1'}(R_1'), \theta_1^{E_1'}(R_1')\right) \otimes \left(\theta_0^{E_2'}(R_2'), \theta_1^{E_2'}(R_2')\right)\right)^T, \quad (3)$$

where $\otimes$ denotes the Kronecker (tensor) product.

Evaluation of $\widehat{f_{s_2}}$ The evaluation via $\widehat{f_{s2}}$ proceeds analogously to that of generic isogenies. Once the inverse theta-null point $\left(\theta_j^{A_{a-1}}(0_{A_{a-1}})\right)_j^{-1}$ is derived, the evaluation of a point $R := \widehat{f_{s1}}(R') \in A_{E_1' \times E_2'}$ through $\widehat{f_{s2}}$ using Eq. (10) is given as follows:

$$\left(\theta_j^{A_{a-1}}(\widehat{f_{s2}}(R))\right)_j = \left(\mathcal{H} \circ \mathcal{S}\left(\left(\widetilde{\theta}_j^{A_{E_1' \times E_2'}}(R)\right)_j\right)\right) \star \left(\theta_j^{A_{a-1}}(0_{A_{a-1}})\right)_j^{-1}. \quad (4)$$

Nevertheless, it follows from [20, Section 3] that one of the coordinates of the theta-null point

$$\left(\theta_j^{A_{a-1}}(0)\right)_j = \left(\theta_{00}^{A_{a-1}}(0) : \theta_{10}^{A_{a-1}}(0) : \theta_{01}^{A_{a-1}}(0) : \theta_{11}^{A_{a-1}}(0)\right)$$

equals zero since $A_{E_1' \times E_2'}$ corresponds to a product of two elliptic curves. If we directly evaluate $\widehat{f_{s2}}$ by Eq. (4), then one of the coordinate components cannot be recovered. Fortunately, the following proposition demonstrates that if the evaluated point $R' \in E_1' \times E_2'$ assumes a specific form, the corresponding coordinate component of the image point $\widehat{f_{s2}}(R) \in A_{a-1}$ strictly vanishes.

Proposition 2. *Suppose that* $\left(\theta_j^{A_{a-1}}(0)\right)_j = \left(\theta_{00}^{A_{a-1}}(0) : \theta_{10}^{A_{a-1}}(0) : \theta_{01}^{A_{a-1}}(0) : 0\right)$. *If* $R' = (R_1', \mathcal{O}_{E_2'})$ *or* $R' = (\mathcal{O}_{E_1'}, R_2')$, *then the last coordinate component of the image* $\widehat{f}_s(R') = \widehat{f_{s2}} \circ \widehat{f_{s1}}(R')$ *must be zero.*

Proof. Let $X = E_1' \times E_2'$, $A = A_{a-1}$, and $g = \widehat{f}_s = \widehat{f}_{s2} \circ \widehat{f}_{s1} : X \to A$. From the above analysis, there exists a 4×4 matrix $M = H \cdot \widetilde{N_\psi} \cdot (\widetilde{N_1} \otimes \widetilde{N_2})$ such that

$$\widetilde{\theta}_j^{A_{E_1' \times E_2'}}(\widehat{f_{s1}}(T)) = M \cdot (T_1 \otimes T_2), \quad T_1 \otimes T_2 = \begin{pmatrix} x_1 x_2 \\ x_1 z_2 \\ z_1 x_2 \\ z_1 z_2 \end{pmatrix}, \quad (5)$$

where $(x_i : z_i)$, $i = 1, 2$ are the Montgomery coordinates of the i-th component of $T = (T_1, T_2) \in X$, and H denotes the Hadamard matrix.

With these compatible theta structures, the dual duplication formula (Eqs. (1) and (2)) gives, up to a nonzero common scalar,

$$\mathcal{H}\left(\widetilde{\theta}_j^{A_{E_1' \times E_2'}}(\widehat{f_{s1}}(P+Q)) \star \widetilde{\theta}_j^{A_{E_1' \times E_2'}}(\widehat{f_{s1}}(P-Q))\right) = \theta_j^{A_{a-1}}(g(P)) \star \theta_j^{A_{a-1}}(g(Q)), \quad P, Q \in X. \quad (6)$$

Since $\theta_{11}^{A_{a-1}}(0_{A_{a-1}}) = 0$, setting $Q = (\mathcal{O}_{E_1'}, \mathcal{O}_{E_2'})$ and $P = T$ in Eq. (6), and then extending the resulting identity to all T, yields

$$\left[\mathcal{H} \circ \mathcal{S}(\widetilde{\theta}_j^{A_{E_1' \times E_2'}}(\widehat{f_{s1}}(T)))\right]_{11} = 0, \quad (7)$$

where $[\cdot]_{11}$ denotes the 11-component of the (dual) theta coordinates. Let $D = \operatorname{diag}(1,-1,-1,1)$ denote the diagonal matrix determined by the signs in the 11-row of the Hadamard matrix in the coordinate order $(00,10,01,11)$, and put $B = M^{\mathbf{T}} D M$. Then for any theta coordinates Θ_1, Θ_2 regarded as a vector, it holds that

$$[\mathcal{H}(\Theta_1 \star \Theta_2)]_{11} = \Theta_1^{\mathbf{T}} D \Theta_2.$$

Eqs. (5) and (7) imply

$$\begin{aligned}\left[\mathcal{H} \circ \mathcal{S}\big(\widetilde{\theta}_j^{\mathcal{A}_{E_1' \times E_2'}}(\widehat{f_{s1}}(T))\big)\right]_{11} &= \left(\widetilde{\theta}_j^{\mathcal{A}_{E_1' \times E_2'}}(\widehat{f_{s1}}(T))\right)^{\mathbf{T}} D \left(\widetilde{\theta}_j^{\mathcal{A}_{E_1' \times E_2'}}(\widehat{f_{s1}}(T))\right) \\ &= (T_1 \otimes T_2)^{\mathbf{T}} M^{\mathbf{T}} D M (T_1 \otimes T_2) \\ &= (T_1 \otimes T_2)^{\mathbf{T}} B (T_1 \otimes T_2) \\ &= 0. \end{aligned} \tag{8}$$

The two Kummer projections $E_i' \to \mathbb{P}^1$ are surjective over the algebraic closure. Thus Eq. (8) holds for every vector $\mathbf{z} = T_1 \otimes T_2 = (x_1x_2, x_1z_2, x_2z_1, z_1z_2)^{\mathbf{T}}$, with arbitrary x_1, x_2, z_1, z_2. Since B is symmetric and the characteristic is different from 2, comparison of the coefficients of

$$(T_1 \otimes T_2)^{\mathbf{T}} B (T_1 \otimes T_2)$$

gives

$$B_{00} = B_{01} = B_{02} = B_{11} = B_{13} = B_{22} = B_{23} = B_{33} = 0, \qquad B_{03} = -B_{12}.$$

Consequently, for some $\kappa \neq 0$, we obtain

$$M^{\mathbf{T}} D M = \kappa \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{9}$$

The scalar is nonzero because M and D are invertible.

Now fix $R' = (R_1', R_2') \in X$ as in the statement. Choose $S' = (S_1', S_2') \in X$ such that $\theta_{11}^{A_{a-1}}(g(S')) \neq 0$ and the barycentric coordinates below are defined. With the notation of Superglue [23], write the barycentric coordinates of $R_i' \pm S_i'$ as $(u_i : v_i : w_i)$, $i = 1, 2$, thus

$$(x(R_i' + S_i') : z(R_i' + S_i')) = (u_i - v_i : w_i), \quad (x(R_i' - S_i') : z(R_i' - S_i')) = (u_i + v_i : w_i).$$

Define

$$\mathbf{z}_+ = \begin{pmatrix} u_1 - v_1 \\ w_1 \end{pmatrix} \otimes \begin{pmatrix} u_2 - v_2 \\ w_2 \end{pmatrix}, \qquad \mathbf{z}_- = \begin{pmatrix} u_1 + v_1 \\ w_1 \end{pmatrix} \otimes \begin{pmatrix} u_2 + v_2 \\ w_2 \end{pmatrix}.$$

Using Eqs. (5) and (9), the 11-component on the left of Eq. (6) is proportional to

$$\begin{aligned}\mathcal{H}\left(\widetilde{\theta}_j^{\mathcal{A}_{E_1' \times E_2'}}(\widehat{f_{s1}}(R' + S')) \star \widetilde{\theta}_j^{\mathcal{A}_{E_1' \times E_2'}}(\widehat{f_{s1}}(R' - S'))\right) &= \mathbf{z}_+^{\mathbf{T}} M^{\mathbf{T}} D M \mathbf{z}_- \\ &= \kappa w_1 w_2 \big((u_1 - v_1)(u_2 - v_2) + (u_1 + v_1)(u_2 + v_2) \\ &\quad - (u_1 - v_1)(u_2 + v_2) - (u_1 + v_1)(u_2 - v_2)\big) \\ &= 4\kappa\, v_1 v_2 w_1 w_2.\end{aligned}$$

If $R_2' = O_{E_2'}$, then $R_2' + S_2' = S_2'$ and $R_2' - S_2' = -S_2'$ have the same Kummer coordinate, hence $v_2 = 0$. Likewise, $R_1' = O_{E_1'}$ implies $v_1 = 0$. Therefore, the displayed expression is strictly zero in either case. Taking the 11-component of Eq. (6) now gives

$$\theta_{11}^{A_{a-1}}(g(R'))\, \theta_{11}^{A_{a-1}}(g(S')) = 0.$$

Since $\theta_{11}^{A_{a-1}}(g(S')) \neq 0$, we obtain

$$\theta_{11}^{A_{a-1}}\left(\widehat{f}_s(R')\right) = 0,$$

which ends the proof of this proposition. □

Since the evaluation points in our setting are all of the form in Proposition 2, we can select the parameters in advance to ensure that $\theta_{11}^{A_{a-1}}(0) = 0$. On this basis, evaluating $\widehat{f_{s2}}$ at $R \in A_{E_1' \times E_2'}$ via Eq. (4) fully preserves the information across all coordinate components of $\theta_j^{A_{a-1}}(\widehat{f_{s2}}(R))$. Algorithm 6 outlines the procedure for evaluating the gluing isogeny $\widehat{f}_s$ at $R' = \left(R_1', \mathcal{O}_{E_2'}\right)$ or $\left(\mathcal{O}_{E_1'}, R_2'\right)$.

Algorithm 6 EvalDualGluing$\left(R', \left(\theta_j^{A_{E_1' \times E_2'}}\left(0_{A_{E_1' \times E_2'}}\right)\right)_j, \left(\theta_j^{A_{a-1}}\left(0_{A_{a-1}}\right)\right)_j^{-1}, N_\psi\right)$

Input: The point $R' = (R_1', R_2') \in E_1' \times E_2'$ such that $R_1' = \mathcal{O}_{E_1'}$ or $R_2' = \mathcal{O}_{E_2'}$, the theta-null point $\left(\theta_j^{A_{E_1' \times E_2'}}\left(0_{A_{E_1' \times E_2'}}\right)\right)_j = \left(\alpha_{A_{E_1' \times E_2'}} : \beta_{A_{E_1' \times E_2'}} : \gamma_{A_{E_1' \times E_2'}} : \delta_{A_{E_1' \times E_2'}}\right)$ on $A_{E_1' \times E_2'}$, the inverse of the theta-null point $\left(\theta_j^{A_{a-1}}\left(0_{A_{a-1}}\right)\right)_j^{-1} = \left(\alpha_{A_{a-1}}^{-1} : \beta_{A_{a-1}}^{-1} : \gamma_{A_{a-1}}^{-1} : 0\right)$ on A_{a-1}, and the matrix N_ψ with respect to the symplectic isomorphism.

Output: Theta coordinates of $\widehat{f}_s(R') = \widehat{f_{s2}} \circ \widehat{f_{s1}}(R')$, where $\widehat{f}_s : E_1' \times E_2' \to A_{a-1}$.

1: Derive the theta coordinates:

$$(a_1,\ b_1) \leftarrow \left(\alpha_{A_{E_1' \times E_2'}},\ \beta_{A_{E_1' \times E_2'}}\right),\ (a_2,\ b_2) \leftarrow \left(\beta_{A_{E_1' \times E_2'}},\ \delta_{A_{E_1' \times E_2'}}\right).$$

2: Compute the theta coordinates on the product surface: ▷ 2**M**

$$\begin{pmatrix} \theta_0^{E_1'}(R_1') \\ \theta_1^{E_1'}(R_1') \end{pmatrix} = \begin{pmatrix} a_1 & -a_1 \\ b_1 & b_1 \end{pmatrix} \begin{pmatrix} x_{R_1'} \\ z_{R_1'} \end{pmatrix},\ \begin{pmatrix} \theta_0^{E_2'}(R_2') \\ \theta_1^{E_2'}(R_2') \end{pmatrix} = \begin{pmatrix} a_2 & -a_2 \\ b_2 & b_2 \end{pmatrix} \begin{pmatrix} x_{R_2'} \\ z_{R_2'} \end{pmatrix}.$$

3: Compute: ▷ 20**M**

$$\left(\theta_j^{A_{E_1' \times E_2'}}(R)\right)_j = \widetilde{N_\psi} \cdot \left(\left(\theta_0^{E_1'}(R_1'),\ \theta_1^{E_1'}(R_1')\right) \otimes \left(\theta_0^{E_2'}(R_2'),\ \theta_1^{E_2'}(R_2')\right)\right)^T,$$

where $\widetilde{N_\psi}$ is the conjugate transpose of N_ψ, and $R = \widehat{f_{s1}}(R')$.

4: Compute: ▷ 4**S**

$$(X_R,\ Y_R,\ Z_R,\ 0) \leftarrow \mathcal{H} \circ \mathcal{S}\left(\theta_{00}^{A_{E_1' \times E_2'}}(R),\ \theta_{10}^{A_{E_1' \times E_2'}}(R),\ \theta_{01}^{A_{E_1' \times E_2'}}(R),\ \theta_{11}^{A_{E_1' \times E_2'}}(R)\right).$$

5: $\left(X_{\widehat{f_{s2}}(R)},\ Y_{\widehat{f_{s2}}(R)},\ Z_{\widehat{f_{s2}}(R)}\right) \leftarrow X_R \cdot \alpha_{A_{a-1}}^{-1},\ Y_R \cdot \beta_{A_{a-1}}^{-1},\ Z_R \cdot \gamma_{A_{a-1}}^{-1}$. ▷ 3**M**

6: $\left(\theta_j^{A_{a-1}}(\widehat{f}_s(R'))\right)_j \leftarrow \mathcal{H}\left(X_{\widehat{f_{s2}}(R)},\ Y_{\widehat{f_{s2}}(R)},\ Z_{\widehat{f_{s2}}(R)},\ 0\right)$.

7: **return** $\left(\theta_j^{A_{a-1}}(\widehat{f}_s(R'))\right)_j$. ▷ Total: 25**M** + 4**S** = 83**m**

Remark 1. Note that $\left(\theta_j^{A_{a-1}}\left(0_{A_{a-1}}\right)\right)_j^{-1} = \left(\alpha_{A_{a-1}}^{-1} : \beta_{A_{a-1}}^{-1} : \gamma_{A_{a-1}}^{-1} : 0\right)$ can be derived using the batch inversion as $\left(\beta_{A_{a-1}}\gamma_{A_{a-1}} : \alpha_{A_{a-1}}\gamma_{A_{a-1}} : \alpha_{A_{a-1}}\beta_{A_{a-1}} : 0\right)$ within the com-

putation of Φ, at a cost of $3\mathbf{M}$. In Line 2 of Algorithm 6, we have $(x_{R'_1}, z_{R'_1}) = (1, 0)$ or $(x_{R'_2}, z_{R'_2}) = (1, 0)$, thus the computational overhead can be reduced to $2\mathbf{M}$.

Generic isogenies of $\widehat{\Phi}$ Define Q_i to be a point on A_i, and let $\left(\theta_j^{A_i}(Q_i)\right)_j = (x_i : y_i : z_i : w_i)$, $j \in \mathbb{Z}_2 \times \mathbb{Z}_2$ be the level-2 theta coordinates of Q_i. Suppose that all abelian surfaces A_i have been endowed with target theta structures. For each generic isogeny $f_i : A_i \to A_{i+1}$ and its dual $\widehat{f}_i : A_{i+1} \to A_i$, by Eqs. (1) and (2) we obtain

$$\left(\theta_j^{A_i}(P_i + Q_i)\right)_j \star \left(\theta_j^{A_i}(P_i - Q_i)\right)_j = \mathcal{H}\left(\left(\widetilde{\theta}_j^{A_{i+1}}(f_i(P_i))\right)_j \star \left(\widetilde{\theta}_j^{A_{i+1}}(f_i(Q_i))\right)_j\right),$$

$$\mathcal{H}\left(\left(\theta_j^{A_i}(\widehat{f}_i(R_{i+1}))\right)_j \star \left(\theta_j^{A_i}(\widehat{f}_i(S_{i+1}))\right)_j\right) = \left(\widetilde{\theta}_j^{A_{i+1}}(R_{i+1} + S_{i+1})\right)_j \star \left(\widetilde{\theta}_j^{A_{i+1}}(R_{i+1} - S_{i+1})\right)_j,$$

where $P_i, Q_i \in A_i$, and $R_{i+1}, S_{i+1} \in A_{i+1}$. Substituting $S_{i+1} = 0_{A_{i+1}}$ into the second equality above yields

$$\left(\theta_j^{A_i}(\widehat{f}_i(R_{i+1}))\right)_j \star \left(\theta_j^{A_i}(0_{A_i})\right)_j = \mathcal{H} \circ \mathcal{S}\left(\left(\widetilde{\theta}_j^{A_{i+1}}(R_{i+1})\right)_j\right), \tag{10}$$

where $\mathcal{S}$ represents the Hadamard squaring such that $\mathcal{S}\left((x_j)_j\right) = \left(x_j^2\right)_j$.

From Eq. (10) it follows that evaluating R_{i+1} by $\widehat{f}_i$ requires the (Hadamard) inverse of the theta coordinates of 0_{A_i}. Note that all theta-null points 0_{A_i}, $i = 1, \cdots, a-1$ have been derived during the computation of Φ. Additionally, we should determine whether the inverses $\left(\theta_j^{A_i}(0_{A_i})\right)_j^{-1}$, $i = 1, \cdots, a-1$ are also computed, as summarized in Remark 2.

Remark 2. Following [20, Section 5.2] and [30, Section 3.1], the inverse theta-null points $\left(\theta_j^{A_i}(0_{A_i})\right)_j^{-1}$ and the inverse squared dual theta-null points $\left(\widetilde{\theta}_j^{A_i}(0_{A_i})\right)_j^{-2}$ for $i = 1, \ldots, a-1$ are typically precomputed to accelerate point doublings, provided that a particular branch of the optimal strategy tree necessitates a massive number of these operations. In certain scenarios, however, such precomputations can be omitted if a branch relies exclusively on point evaluations rather than doublings. Nevertheless, since evaluating points via $\widehat{\Phi}$ inherently requires all the inverse theta-null points $\left(\theta_j^{A_i}(0_{A_i})\right)_j^{-1}$ for $i = 1, \ldots, a-1$, the (mixed) optimal strategy for computing Φ originally proposed in [30, Section 3.3] must be explicitly adapted to guarantee their availability. Fortunately, such adjustment incurs only a marginal performance penalty as the precomputation overhead is negligible relative to the overall cost of the $(2, 2)$-isogeny chain Φ.

Splitting isogenies of $\widehat{\Phi}$ We now focus on the splitting isogeny $\widehat{f}_g : A_1 \to E_1 \times E_2$ of $\widehat{\Phi}$. As with the gluing isogeny, $\widehat{f}_g$ decomposes into the following two maps

$$\widehat{f_{g1}} : A_1 \to A_{E_1 \times E_2}, \quad \widehat{f_{g2}} : A_{E_1 \times E_2} \to E_1 \times E_2,$$

where $\widehat{f_{g1}}$ and $\widehat{f_{g2}}$ are the duals of the gluing isogenies $f_{g1} : A_{E_1 \times E_2} \to A_1$ and $f_{g2} : E_1 \times E_2 \to A_{E_1 \times E_2}$ of Φ, respectively. The isogeny f_{g2} corresponds to a 4×4 change-of-basis matrix M, which associates the theta structure of $A_{E_1 \times E_2}$ with the correct symplectic 4-torsion basis [20, Section 2.3]. Under this change of basis, the corresponding theta coordinates are given by

$$\left(\theta_j^{A_{E_1 \times E_2}}(\widetilde{R})\right)_j = M \cdot \left((\widetilde{x}_1, \widetilde{z}_1) \otimes (\widetilde{x}_2, \widetilde{z}_2)\right)^T,$$

where $\widetilde{R} \in A_{E_1 \times E_2}$, and $(\widetilde{x}_1 : \widetilde{z}_1)$ and $(\widetilde{x}_2 : \widetilde{z}_2)$ are the Montgomery coordinates on E_1 and E_2, respectively.

From Eq. (10), it follows that the evaluation of a point $R \in A_1$ via $\widehat{f_{g1}}$ is accomplished using the inverse theta-null point $\left(\theta_j^{A_{E_1 \times E_2}}\left(0_{A_{E_1 \times E_2}}\right)\right)_j^{-1}$

$$\left(\theta_j^{A_{E_1 \times E_2}}(\widehat{f_{g1}}(R))\right)_j = \left(\mathcal{H} \circ \mathcal{S}\left(\left(\widetilde{\theta}_j^{A_1}(R)\right)_j\right)\right) \star \left(\theta_j^{A_{E_1 \times E_2}}(0_{A_{E_1 \times E_2}})\right)_j^{-1}.$$

Regarding $\widehat{f_{g2}}$, the relation $f_{g1} \circ f_{g2} \circ \widehat{f_{g2}} \circ \widehat{f_{g1}} = [2]$ implies that $\widehat{f_{g2}}$ serves as the inverse of f_{g2}. Consequently, the evaluation of a point $\widetilde{R} = \widehat{f_{g1}}(R)$ via $\widehat{f_{g2}}$ is executed as follows:

(1) Compute the (projective) inverse M^{-1} of M to derive the tensor product

$$((\widetilde{x}_1, \widetilde{z}_1) \otimes (\widetilde{x}_2, \widetilde{z}_2))^T = (\widetilde{x}_1\widetilde{x}_2, \widetilde{z}_1\widetilde{x}_2, \widetilde{x}_1\widetilde{z}_2, \widetilde{z}_1\widetilde{z}_2)^T = M^{-1} \cdot \left(\theta_j^{A_{E_1 \times E_2}}(\widetilde{R})\right)_j.$$

(2) Recover the Montgomery coordinates on E_1 and E_2 by exploiting the projective relations $(\widetilde{x}_1 : \widetilde{z}_1) = (\widetilde{x}_1\widetilde{x}_2 : \widetilde{z}_1\widetilde{x}_2)$ and $(\widetilde{x}_2 : \widetilde{z}_2) = (\widetilde{x}_1\widetilde{x}_2 : \widetilde{x}_1\widetilde{z}_2)$.

The procedure for evaluating the splitting isogeny $\widehat{f_g}$ at $R \in A_1$ is presented in Algorithm 7. Specifically, the projective inverse M^{-1} can be obtained by computing the adjugate matrix of M (Line 2 of Algorithm 7), resulting in a cost of 72**M** (see Algorithm 15 for more details).

Algorithm 7 EvalDualSplitting$\left(R, \left(\theta_j^{A_{E_1 \times E_2}}\left(0_{A_{E_1 \times E_2}}\right)\right)_j^{-1}, M\right)$

Input: The point $R \in A_1$, the inverse of the theta-null point $\left(\theta_j^{A_{E_1 \times E_2}}\left(0_{A_{E_1 \times E_2}}\right)\right)_j^{-1} = \left(\alpha^{-1}_{A_{E_1 \times E_2}} : \beta^{-1}_{A_{E_1 \times E_2}} : \gamma^{-1}_{A_{E_1 \times E_2}} : \delta^{-1}_{A_{E_1 \times E_2}}\right)$ on $A_{E_1 \times E_2}$, and the change-of-basis matrix M.
Output: Montgomery coordinates of $\widehat{f_g}(R) = (\widetilde{R_1}, \widetilde{R_2})$, where $\widehat{f_g} : A_1 \to E_1 \times E_2$.

1: $\left(\theta_j^{A_{E_1 \times E_2}}(\widehat{f_{g1}}(R))\right)_j \leftarrow \left(\mathcal{H} \circ \mathcal{S}\left(\left(\widetilde{\theta}_j^{A_1}(R)\right)_j\right)\right) \star \left(\theta_j^{A_{E_1 \times E_2}}(0_{A_{E_1 \times E_2}})\right)_j^{-1}$. ▷ 4**M** + 4**S**
2: $M^{-1} \leftarrow \mathsf{MatrixInverseProj}(M)$. ▷ Algorithm 15, 72**M**
3: $(\widetilde{x}_1\widetilde{x}_2, \widetilde{z}_1\widetilde{x}_2, \widetilde{x}_1\widetilde{z}_2, \widetilde{z}_1\widetilde{z}_2)^T \leftarrow M^{-1} \cdot \left(\theta_j^{A_{E_1 \times E_2}}(\widetilde{R})\right)_j$. ▷ 16**M**
4: $(\widetilde{x}_1 : \widetilde{z}_1) \leftarrow (\widetilde{x}_1\widetilde{x}_2 : \widetilde{z}_1\widetilde{x}_2)$, $(\widetilde{x}_2 : \widetilde{z}_2) \leftarrow (\widetilde{x}_1\widetilde{x}_2 : \widetilde{x}_1\widetilde{z}_2)$.
5: **return** $\left((\widetilde{x}_1, \widetilde{z}_1), (\widetilde{x}_2, \widetilde{z}_2)\right)$. ▷ Total: 92**M** + 4**S** = 284**m**

Remark 3. In certain scenarios, the Superglue technique [23] is employed to compute $f_g = f_{g1} \circ f_{g2}$, thereby circumventing expensive arithmetic operations over the extension field $\mathbb{F}_{p^4}$. Unlike traditional gluing isogenies, superglue leverages barycentric coordinates to directly obtain the theta coordinates on the codomain A_1. However, in this approach, the entries of the change-of-basis matrix M are not entirely determined. Since recovering $\widehat{f_{g2}}$ necessitates complete knowledge of M, additional computations are required during the evaluation of f_{g2}. For the sake of clarity, we omit the theoretical exposition of the Superglue integration here, though it is fully supported in our accompanying implementation.

Algorithm 8 EvalDualTwoDim(Q', Θ, M, N_ψ): A detailed and optimized version of evaluating a two-dimensional representation via dual isogenies

Input: A point $Q' = (Q'_1,\ Q'_2) \in E'_1 \times E'_2$, where $Q'_i = \mathcal{O}_{E'_i}$ for some $i \in \{1, 2\}$ and the other point is of order D with $\gcd(D, q(2^a - q)) = 1$.

The tuple $\Theta = \left(\left(\theta_j^{A_{E_1 \times E_2}}(0_{A_{E_1 \times E_2}}) \right)_j,\ \left\{ \left(\theta_j^{A_i}(0_{A_i}) \right)_j \right\}_{i=1}^{a-1},\ \left(\theta_j^{A_{E'_1 \times E'_2}}(0_{A_{E'_1 \times E'_2}}) \right)_j \right)$ along with the inverses of the theta coordinates in Θ, the change-of-basis matrix M, and the symplectic automorphism N_ψ obtained from the computation of $\Phi : E_1 \times E_2 \to E'_1 \times E'_2$.

Output: The image point $\widehat{\varphi_2}(Q'_2)$ if $Q'_1 = \mathcal{O}_{E'_1}$ or $\varphi'_2(Q'_1)$ if $Q'_2 = \mathcal{O}_{E'_2}$.

1: $Q' \leftarrow (-Q'_1, Q'_2)$.

2: $\left(\theta_j^{A_{a-1}}(Q_{a-1}) \right)_j \leftarrow$ EvalDualGluing$\left(Q', \left(\theta_j^{A_{E'_1 \times E'_2}}\left(0_{A_{E'_1 \times E'_2}}\right) \right)_j, \left(\theta_j^{A_{a-1}}\left(0_{A_{a-1}}\right) \right)_j^{-1}, N_\psi \right)$.
▷ Algorithm 6, $25\mathbf{M} + 4\mathbf{S}$

3: **for** $i = a - 1, \ldots, 2$ **do**

4: $\left(\theta_j^{A_{i-1}}(Q_{i-1}) \right)_j \leftarrow \left(\mathcal{H} \circ \mathcal{S}\left((\widetilde{\theta}_j^{A_i}(Q_i))_j \right) \right) \star \left(\theta_j^{A_{i-1}}(0_{A_{i-1}}) \right)_j^{-1}$. ▷ $4\mathbf{M} + 4\mathbf{S}$ per bit

5: $\left(\widetilde{\theta}_j^{A_{i-1}}(Q_{i-1}) \right)_j \leftarrow \mathcal{H}\left(\left(\theta_j^{A_{i-1}}(Q_{i-1}) \right)_j \right)$.

6: **end for**

7: $(P_1, P_2) \leftarrow$ EvalDualSplitting$\left(Q_1, \left(\theta_j^{A_{E_1 \times E_2}}\left(0_{A_{E_1 \times E_2}}\right) \right)_j^{-1}, M \right)$. ▷ Algorithm 7, $92\mathbf{M} + 4\mathbf{S}$

8: **return** P_1 if $Q'_1 = \mathcal{O}_{E'_1}$ or P_2 if $Q'_2 = \mathcal{O}_{E'_2}$.
▷ Total: $(4a + 109)\mathbf{M} + 4a\mathbf{S} = (20a + 327)\mathbf{m}$

Full evaluation of $\widehat{\Phi}$ Algorithm 8 details the procedure for evaluating two-dimensional representations via dual isogenies, serving as an explicit version of Line 3 in Algorithm 4, and Line 2 of Algorithm 5.

Notably, Lines 2–7 of Algorithm 8 correspond to the evaluation at $\left(-\varphi_1(Q), 0_{E'_2}\right)$ via $\widehat{\Phi} = \widehat{f_{g2}} \circ \widehat{f_{g1}} \circ \widehat{f_1} \circ \cdots \circ \widehat{f_{a-2}} \circ \widehat{f_{s2}} \circ \widehat{f_{s1}}$. In particular, Lines 2 and 7 evaluate the isogeny $\widehat{f_s} = \widehat{f_{s2}} \circ \widehat{f_{s1}}$ and $\widehat{f_g} = \widehat{f_{g2}} \circ \widehat{f_{g1}}$, respectively. Furthermore, Algorithm 8 demonstrates that explicitly computing $\widehat{\Phi}$ or using $\ker(\widehat{\Phi})$ is unnecessary because information obtained while computing the isogeny chain Φ can be reused.

In summary, our proposed method consists of one two-dimensional isogeny computation (Line 1 of Algorithm 4) and two isogeny evaluations. The evaluation in Line 2 of Algorithm 4 and the evaluation in Lines 2-7 of Algorithm 8 together have a cost that can be approximated as that of a single two-dimensional isogeny evaluation. Compared to the previous method [29, Algorithm 1], our approach successfully eliminates two high-dimensional isogeny evaluations, as well as the costly basis generation and two-dimensional discrete logarithm computation.

Table 2 shows the computational costs of each building block of the previous [29, Algorithm 1] method and our method, according to our cost analysis for Algorithms 6-8 (Rows 6-8 of Table 2) and the implementation results of POKÉ (Rows 1-5 of Table 2). Note that the previous method requires evaluating four distinct points under Φ, whereas our optimized approach halves this cost by evaluating only two.

Remark 4. By employing the Superglue technique, we achieve an efficiency gain in the computation of Φ, despite the execution of extra inversions of theta-null points (see Remark 2 for further details) compared with the existing algorithm.

From Table 2, it follows that our proposed approach reduces the computational overhead by about 67.2% (a squaring over $\mathbb{F}_p$ is estimated as $\mathbf{s} = 0.8\mathbf{m}$), yielding a significant performance improvement compared with existing methods. Algorithm 8 can be utilized

Table 2: Comparison of computational costs of the previous [29, Algorithm 1] method and our new method for evaluating two-dimensional representations (the Superglue method has been integrated into our implementation). The overhead is measured as multiplications/squarings over $\mathbb{F}_p$. The length of the $(2,2)$-isogeny chain is $a = 124$. The order of the evaluated point is 3^b, $b = 158$. The cost of evaluating generic isogenies of $\widehat{\Phi}$ can be estimated as $(a-2)(4\mathbf{M}+4\mathbf{S}) = 488\mathbf{M}+488\mathbf{S} = 2440\mathbf{m}$ according to Line 4 of Algorithm 8.

	[29, Algorithm 1]	Our Method
Computation of Φ	73653**m** + 7295**s**	73024**m** + 6523**s**
Evaluation(s) of Φ	10376**m**	2644**m**
Basis generation	20908**m** + 3766**s**	–
Two-dimensional discrete logarithm	116354**m** + 16186**s**	–
Two-dimensional scalar multiplication	11574**m** + 386**s**	–
EvalDualGluing (Algorithm 6)	–	83**m**
Evaluate generic isogenies of $\widehat{\Phi}$	–	2440**m**
EvalDualSplitting (Algorithm 7)	–	284**m**
Total	232865**m** + 27633**s**	78475**m** + 6523**s**

to accelerate the key generation and decryption phases of POKÉ and its variants, together with the signing phase of SQIsign (see Section 4 for more details). Furthermore, our approach relaxes the constraints on parameter selection (the order of the evaluated point is no longer required to be smooth), thereby providing greater flexibility in the selection of parameters and the construction of variants for POKÉ.

4 Applications to Isogeny-Based Cryptography

In this section, we apply the new evaluation algorithm for two-dimensional representations (introduced in Section 3) to optimize isogeny-based cryptographic schemes, including the POKÉ variants, SQIsign-v3.0, and SQIsign2DPush. To facilitate reference, we denote the speed-optimized variants of POKÉ and PIKE as POKÉD and PIKED, respectively, following the convention established in Section 1.1.

4.1 Fast Variants of POKÉ and PIKE

In this subsection, we demonstrate how our proposed method accelerates the POKÉ protocol instantiated with a single-point shared secret. Subsequently, we provide the framework of POKÉD and PIKED, two efficient variants integrating our optimization techniques. Crucially, our approach removes the constraint that D must be a smooth integer. Furthermore, by employing the *Superglue* [23] technique, we entirely avoid costly arithmetic operations over the extension field $\mathbb{F}_{p^4}$. This combined flexibility naturally facilitates the deployment of the twist trick, thereby allowing part of the torsion points to lie on the quadratic twist.

POKÉD In the following, we present the framework of POKÉD. We first state the parameters. Let $p = 2^a f - 1$ be a prime and $3^b D \mid p-1$, where f is a cofactor. Ensuring that both 3^b and D divide $p-1$ is designed to yield a more compact public key. With the same notation as in Section 2.3, define $E_0 : y^2 = x^3 + x$ to be a supersingular curve with endomorphism ring $\mathrm{End}(E_0) = \mathbb{Z}\left\langle 1, i, \frac{i+j}{2}, \frac{1+ij}{2} \right\rangle$. Denote $\{P_0, Q_0\}$ and $\{R_0, S_0\}$ as bases of $E_0[2^a]$ and $E_0[3^b]$, respectively, and let $X_0 \in E_0$ be the point of order D.

The key generation phase of POKÉD entails evaluating an isogeny $\varphi : E_0 \to E_A$ of degree $q(2^a - q)$ with $\gcd(q(2^a - q), 6D) = 1$, admitting a two-dimensional representation. By utilizing our approach, the evaluations of R_0, S_0, and X_0 can be executed through EvalDualTwoDim (Algorithm 8). The key-generation procedure is presented in Algorithm 9.

Algorithm 9 Key Generation of POKÉD

Input: $E_0, P_0, Q_0 \in E_0[2^a], R_0, S_0 \in E_0[3^b], X_0 \in E_0[D]$.
Output: A secret/public key pair $(\mathsf{sk}, \mathsf{pk})$.

1: Sample a random $q \in (0, 2^a)$ such that $\gcd(q(2^a - q), 6D) = 1$.
2: Generate an endomorphism θ such that $\deg(\theta) = q(2^a - q)3^b$.
3: Compute the isogeny $\tau : E_0 \to E_A$ with kernel $\ker(\tau) = \ker(\widehat{\theta}) \cap E_0[3^b]$.
4: $\varphi(P_0), \varphi(Q_0) \leftarrow [(3^b)^{-1} \bmod 2^a]\tau \circ \theta(P_0), [(3^b)^{-1} \bmod 2^a]\tau \circ \theta(Q_0)$.
5: Compute Φ with kernel $\ker(\Phi) = \langle([-q]P_0, \varphi(P_0)), ([-q]Q_0, \varphi(Q_0))\rangle$, obtaining

$$\Theta = \left(\left(\theta_j^{A_{E_0 \times E_A}}(0_{A_{E_0 \times E_A}})\right)_j, \left\{ \left(\theta_j^{A_i}(0_{A_i})\right)_j \right\}_{i=1}^{a-1}, \left(\theta_j^{A_{E'_0 \times E'_A}}(0_{A_{E'_0 \times E'_A}})\right)_j \right).$$

along with the inverses of the theta coordinates in Θ, the change-of-basis matrix M, and the symplectic automorphism N_ψ. ▷ $\Phi : E_0 \times E_A \to E'_0 \times E'_A$.

6: $(\varphi_1(R_0), _), (\varphi_1(S_0), _) \leftarrow \Phi(R_0, \mathcal{O}_{E_A}), \Phi(S_0, \mathcal{O}_{E_A})$.
7: $\varphi(R_0) \leftarrow \mathsf{EvalDualTwoDim}\left((\varphi_1(R_0), \mathcal{O}_{E'_A}), \Theta, M, N_\psi\right)$. ▷ Algorithm 8
8: $\varphi(S_0) \leftarrow \mathsf{EvalDualTwoDim}\left((\varphi_1(S_0), \mathcal{O}_{E'_A}), \Theta, M, N_\psi\right)$. ▷ Algorithm 8
9: $(\varphi_1(X_0), _) \leftarrow \Phi(X_0, \mathcal{O}_{E_A})$.
10: $\varphi(X_0) \leftarrow \mathsf{EvalDualTwoDim}\left((\varphi_1(X_0), \mathcal{O}_{E'_A}), \Theta, M, N_\psi\right)$. ▷ Algorithm 8
11: Sample $\alpha_2, \beta_2 \xleftarrow{\$} \mathbb{Z}_{2^a}^\times$, $\gamma_3 \xleftarrow{\$} \mathbb{Z}_{3^b}^\times$, $\delta_D \xleftarrow{\$} \mathbb{Z}_D^\times$.
12: $P_A, Q_A \leftarrow [\alpha_2]\varphi(P_0), [\beta_2]\varphi(Q_0)$.
13: $R_A, S_A \leftarrow [\gamma_3]\varphi(R_0), [\gamma_3]\varphi(S_0)$.
14: $X_A \leftarrow [\delta_D]\varphi(X_0)$.
15: **return** $\mathsf{sk} = (q, \alpha_2, \beta_2, \delta_D)$, $\mathsf{pk} = (E_A, P_A, Q_A, R_A, S_A, X_A)$.

Remark 5. An alternative method for evaluating X_0 is to compute $\varphi(X_0) = [(3^b)^{-1} \bmod D]\tau \circ \theta(X_0)$. However, our analysis and experimental results demonstrate that the computational overhead of this strategy actually exceeds that of evaluating the two-dimensional representation by our proposed techniques (Algorithm 8). Consequently, we prefer to evaluate points whose order is distinct from 2^a via two-dimensional isogenies.

The encryption phase closely follows that of POKÉ. Because we optimize the single-point shared-secret variant of POKÉ, we solely map the individual points X_0 and X_A via the parallel 3^b-isogenies $\psi : E_0 \to E_B$ and $\psi' : E_A \to E_{AB}$, respectively. Algorithm 10 summarizes this procedure.

For decryption, the receiver can evaluate X_B via $\varphi' = [\psi]_*\varphi$ to derive the shared information X_{AB} using Algorithm 8, as presented in Algorithm 11.

To achieve λ-bit security, one must set $a \gtrsim \lambda$ to resist weak-key attacks [33], $3^b \gtrsim 2^{2\lambda}$ to prevent meet-in-the-middle (MITM) attacks, and $D \gtrsim 2^{\lambda/2}$ to resist brute-force attacks (discussed in Section 2.3), resulting in a prime p of about 3λ bits. Compared with POKÉ, this variant possesses the following advantages:

(1) Both the key generation and decryption phases exhibit substantial efficiency gains due to the employment of our new method (the time-consuming two-dimensional discrete logarithm no longer needs to be performed);

Algorithm 10 Encryption of POKÉD

Input: A message m, a public key $E_A, P_A, Q_A, R_A, S_A, X_A$.
Output: A ciphertext ct.

1: Sample $r_3 \xleftarrow{\$} \mathbb{Z}_{3^b}$, $\omega_2 \xleftarrow{\$} \mathbb{Z}_{2^a}^{\times}$, $\eta_D \xleftarrow{\$} \mathbb{Z}_D^{\times}$.
2: Compute the isogeny $\psi : E_0 \to E_B$ with kernel $\langle R_0 + [r_3]S_0 \rangle$.
3: Compute the isogeny $\psi' : E_A \to E_{AB}$ with kernel $\langle R_A + [r_3]S_A \rangle$.
4: $P_B, Q_B \leftarrow [\omega_2]\psi(P_0), [1/\omega_2]\psi(Q_0)$.
5: $P_{AB}, Q_{AB} \leftarrow [\omega_2]\psi'(P_A), [1/\omega_2]\psi'(Q_A)$.
6: $X_B, X_{AB} \leftarrow [\eta_D]\psi(X_0), [\eta_D]\psi'(X_A)$.
7: $\mathsf{ct}' \leftarrow \mathrm{KDF}(X_{AB}) \oplus m$.
8: **return** $\mathsf{ct} = ((E_B, P_B, Q_B, X_B), (E_{AB}, P_{AB}, Q_{AB}), \mathsf{ct}')$.

Algorithm 11 Decryption of POKÉD

Input: $\mathsf{sk} = (q, \alpha_2, \beta_2, \delta_D)$, $\mathsf{ct} = ((E_B, P_B, Q_B, X_B), (E_{AB}, P_{AB}, Q_{AB}), \mathsf{ct}')$.
Output: A message m'.

1: $P'_{AB}, Q'_{AB} \leftarrow [1/\alpha_2]P_{AB}, [1/\beta_2]Q_{AB}$.
2: Compute Φ' with kernel $\langle ([-q]P_B, P'_{AB}), ([-q]Q_B, Q'_{AB}) \rangle$, obtaining

$$\Theta' = \left(\left(\theta_j^{A_{E_B \times E_{AB}}}(0_{A_{E_B \times E_{AB}}})\right)_j, \left\{ \left(\theta_j^{A'_i}(0_{A'_i})\right)_j \right\}_{i=1}^{a-1}, \left(\theta_j^{A_{E'_B \times E'_{AB}}}(0_{A_{E'_B \times E'_{AB}}})\right)_j \right).$$

along with the inverses of the theta coordinates in Θ', the change-of-basis matrix M', and the symplectic automorphism N'_ψ. ▷ $\Phi' : E_B \times E_{AB} \to E'_B \times E'_{AB}$.

3: $(\overline{\varphi}_1(X_B), _) \leftarrow \Phi'(X_B, \mathcal{O}_{E_{AB}})$.
4: $\varphi'(X_B) \leftarrow \mathsf{EvalDualTwoDim}\left((\overline{\varphi}_1(X_B),\ \mathcal{O}_{E'_{AB}}), \Theta', M', N'_\psi\right)$. ▷ Algorithm 8
5: $X_{AB} \leftarrow [\delta_D]\varphi'(X_B)$.
6: $m' \leftarrow \mathrm{KDF}(X_{AB}) \oplus \mathsf{ct}'$.
7: **return** m'.

(2) The size of the parameter p is reduced from $\frac{10}{3}\lambda$ to 3λ, leading to smaller public key/ciphertext sizes and an overall efficiency enhancement.

PIKED For PIKED, we also choose p of the form $p = 2^a f - 1,\ 3^b D \mid p - 1$ as in POKÉD. Since PIKE uses the pairing values as the shared secret, our proposed method in Section 3 is only applicable to accelerating its key generation phase. The explicit procedure for the key generation phase of PIKED is the same as that of POKÉD (Algorithm 9). For brevity, we omit the details of the encryption and decryption phases of PIKED. One can refer to [11, Algorithms 5 and 6] for more details.

According to the security analysis in [11, Section 4], we should take $a \gtrsim \lambda$, $3^b \gtrsim 2^{2\lambda}$, and $D \gtrsim 2^\lambda$ to ensure λ-bit security for PIKED, leading to a 3λ-bit prime p. In comparison with PIKE, this variant exhibits the following advantages:

(1) The performance of the key generation phase is significantly enhanced;
(2) The encryption phase exclusively involves 3-isogeny evaluations (whereas PIKE necessitates both 3- and 5-isogenies), thereby yielding a moderate speedup.

4.2 Applications to SQIsign-v3.0

We next investigate the application to SQIsign-v3.0, whose response diagram is illustrated in Figure 3.

In the Σ protocol, the prover is required to construct an efficient representation of the response isogeny φ_{rsp} by evaluating $\varphi_{\mathsf{rsp}} \circ \widehat{\varphi_{\mathsf{aux}}}$ on a deterministic 2^e-torsion basis

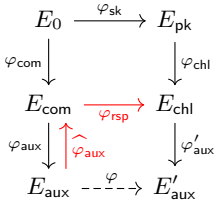


Figure 3: The diagram for SQIsign-v3.0.

$\langle P_{\mathsf{aux}}, Q_{\mathsf{aux}} \rangle = E_{\mathsf{aux}}[2^e]$ of the auxiliary curve E_{aux} (as marked in red in Figure 3). The state-of-the-art approach accomplishes this by solving a two-dimensional discrete logarithm problem to recover the change-of-basis matrix, which corresponds to [1, Algorithms 3.9 and 4.133] in SQIsign-v3.0.

We now propose an optimization that eliminates the need to solve the two-dimensional discrete logarithm problem by integrating the dual isogeny evaluation technique. Recall the two-dimensional isogeny Φ defined in the response phase

$$\Phi = \begin{pmatrix} \varphi_{\mathsf{rsp}} & -\widehat{\varphi'_{\mathsf{aux}}} \\ \varphi_{\mathsf{aux}} & \widehat{\varphi} \end{pmatrix} : E_{\mathsf{com}} \times E'_{\mathsf{aux}} \to E_{\mathsf{chl}} \times E_{\mathsf{aux}}.$$

Its dual isogeny is given by

$$\widehat{\Phi} = \begin{pmatrix} \widehat{\varphi_{\mathsf{rsp}}} & \widehat{\varphi_{\mathsf{aux}}} \\ -\varphi'_{\mathsf{aux}} & \varphi \end{pmatrix} : E_{\mathsf{chl}} \times E_{\mathsf{aux}} \to E_{\mathsf{com}} \times E'_{\mathsf{aux}}.$$

Instead of computing the preimages of P_{aux} and Q_{aux} via a discrete logarithm, we directly evaluate the target basis using the two-step "dual-then-forward" approach (Algorithm 5) to obtain $\varphi_{\mathsf{rsp}} \circ \widehat{\varphi_{\mathsf{aux}}}(P_{\mathsf{aux}})$ and $\varphi_{\mathsf{rsp}} \circ \widehat{\varphi_{\mathsf{aux}}}(Q_{\mathsf{aux}})$.

We first evaluate the dual isogeny $\widehat{\Phi}$ at $(\mathcal{O}_{E_{\mathsf{chl}}}, P_{\mathsf{aux}})$ and $(\mathcal{O}_{E_{\mathsf{chl}}}, Q_{\mathsf{aux}})$

$$\widehat{\Phi}\begin{pmatrix} \mathcal{O}_{E_{\mathsf{chl}}} \\ P_{\mathsf{aux}} \end{pmatrix} = \begin{pmatrix} \widehat{\varphi_{\mathsf{aux}}}(P_{\mathsf{aux}}) \\ * \end{pmatrix}, \ \widehat{\Phi}\begin{pmatrix} \mathcal{O}_{E_{\mathsf{chl}}} \\ Q_{\mathsf{aux}} \end{pmatrix} = \begin{pmatrix} \widehat{\varphi_{\mathsf{aux}}}(Q_{\mathsf{aux}}) \\ * \end{pmatrix}$$

using Algorithm 8. Let $R_P = \widehat{\varphi_{\mathsf{aux}}}(P_{\mathsf{aux}})$, $R_Q = \widehat{\varphi_{\mathsf{aux}}}(Q_{\mathsf{aux}}) \in E_{\mathsf{com}}$ be the extracted first components. Subsequently, we evaluate the forward isogeny Φ at $(R_P, \mathcal{O}_{E'_{\mathsf{aux}}})$ and $(R_Q, \mathcal{O}_{E'_{\mathsf{aux}}})$

$$\Phi\begin{pmatrix} R_P \\ \mathcal{O}_{E'_{\mathsf{aux}}} \end{pmatrix} = \begin{pmatrix} \varphi_{\mathsf{rsp}} \circ \widehat{\varphi_{\mathsf{aux}}}(P_{\mathsf{aux}}) \\ * \end{pmatrix}, \ \Phi\begin{pmatrix} R_Q \\ \mathcal{O}_{E'_{\mathsf{aux}}} \end{pmatrix} = \begin{pmatrix} \varphi_{\mathsf{rsp}} \circ \widehat{\varphi_{\mathsf{aux}}}(Q_{\mathsf{aux}}) \\ * \end{pmatrix}.$$

Let

$$\widetilde{P_{\mathsf{chl}}} = \varphi_{\mathsf{rsp}} \circ \widehat{\varphi_{\mathsf{aux}}}(P_{\mathsf{aux}}), \ \widetilde{Q_{\mathsf{chl}}} = \varphi_{\mathsf{rsp}} \circ \widehat{\varphi_{\mathsf{aux}}}(Q_{\mathsf{aux}}).$$

The required target basis $P_{\mathsf{chl}}, Q_{\mathsf{chl}}$ would normally be obtained by performing a scalar multiplication by $\deg(\varphi_{\mathsf{rsp}})^{-1} \pmod{2^{e_{\mathsf{rsp}}}}$ on $\widetilde{P_{\mathsf{chl}}}, \widetilde{Q_{\mathsf{chl}}}$. In fact, this scalar multiplication can be avoided by postponing it to the subsequent matrix computation. To make this explicit, we denote by $\langle U_{\mathsf{chl}}, V_{\mathsf{chl}} \rangle$ a deterministic basis of $E_{\mathsf{chl}}[2^e]$ obtained from [1, Algorithm 4.94]. The original signing algorithm computes the matrix M_{chl} satisfying

$$\begin{pmatrix} P_{\mathsf{chl}} \\ Q_{\mathsf{chl}} \end{pmatrix} = M_{\mathsf{chl}} \begin{pmatrix} U_{\mathsf{chl}} \\ V_{\mathsf{chl}} \end{pmatrix}.$$

Instead, we compute the matrix $\widetilde{M_{\mathsf{chl}}}$ satisfying

$$\begin{pmatrix} \widetilde{P_{\mathsf{chl}}} \\ \widetilde{Q_{\mathsf{chl}}} \end{pmatrix} = \widetilde{M_{\mathsf{chl}}} \begin{pmatrix} U_{\mathsf{chl}} \\ V_{\mathsf{chl}} \end{pmatrix} \tag{11}$$

and recover M_{chl} by $M_{\mathsf{chl}} = \deg(\varphi_{\mathsf{rsp}})^{-1} \cdot \widetilde{M_{\mathsf{chl}}} \pmod{2^{e_{\mathsf{rsp}}}}$.

Algorithm 12 formalizes this improved response based on the response phase of [1, Algorithm 3.23]. For brevity, we omit the steps that are identical to those in [1, Algorithm 3.23, lines 1–15] and [1, Algorithm 3.21, lines 1–11].

Algorithm 12 Optimized response phase of SQIsign-v3.0

Input: The two-dimensional representation $(q, E_{\mathsf{com}}, E'_{\mathsf{aux}}, (P_{\mathsf{com}}, Q_{\mathsf{com}}), (P'_{\mathsf{aux}}, Q'_{\mathsf{aux}}))$ of the forward isogeny Φ.

Output: $(E_{\mathsf{aux}}, M_{\mathsf{chl}}, \mathrm{hint}_{\mathsf{aux}}, \mathrm{hint}_{\mathsf{chl}})$.

1: Compute Φ with kernel $\langle([-q]P_{\mathsf{com}}, P'_{\mathsf{aux}}), ([-q]Q_{\mathsf{com}}, Q'_{\mathsf{aux}})\rangle$, obtaining

$$\Theta = \left(\left(\theta_j^{A_{E_{\mathsf{com}} \times E'_{\mathsf{aux}}}}(0_{A_{E_{\mathsf{com}} \times E'_{\mathsf{aux}}}})\right)_j, \left\{ \left(\theta_j^{A'_i}(0_{A'_i})\right)_j \right\}_{i=1}^{a-1}, \left(\theta_j^{A_{E_{\mathsf{chl}} \times E_{\mathsf{aux}}}}(0_{A_{E_{\mathsf{chl}} \times E_{\mathsf{aux}}}})\right)_j \right).$$

along with the inverses of the theta coordinates in Θ, the change-of-basis matrix M, and the symplectic automorphism N_ψ. ▷ $\Phi : E_{\mathsf{com}} \times E'_{\mathsf{aux}} \to E_{\mathsf{chl}} \times E_{\mathsf{aux}}$.

2: $(P_{\mathsf{aux}}, Q_{\mathsf{aux}})$, $\mathrm{hint}_{\mathsf{aux}} \leftarrow \mathsf{TorsionBasisToHint}(E_{\mathsf{aux}}, f, e_{\mathsf{rsp}} + 2)$. ▷ [1, Algorithm 4.94]

3: $\widehat{\varphi_{\mathsf{aux}}}(P_{\mathsf{aux}}) \leftarrow \mathsf{EvalDualTwoDim}\left((\mathcal{O}_{E_{\mathsf{chl}}}, P_{\mathsf{aux}}), \Theta, M, N_\psi\right)$. ▷ Algorithm 8

4: $\widehat{\varphi_{\mathsf{aux}}}(Q_{\mathsf{aux}}) \leftarrow \mathsf{EvalDualTwoDim}\left((\mathcal{O}_{E_{\mathsf{chl}}}, Q_{\mathsf{aux}}), \Theta, M, N_\psi\right)$. ▷ Algorithm 8

5: $\left(\widetilde{P_{\mathsf{chl}}}, _\right), \left(\widetilde{Q_{\mathsf{chl}}}, _\right) \leftarrow \Phi\left(\widehat{\varphi_{\mathsf{aux}}}(P_{\mathsf{aux}}), \mathcal{O}_{E'_{\mathsf{aux}}}\right), \Phi\left(\widehat{\varphi_{\mathsf{aux}}}(Q_{\mathsf{aux}}), \mathcal{O}_{E'_{\mathsf{aux}}}\right)$. ▷ $\widetilde{P_{\mathsf{chl}}} = \varphi_{\mathsf{rsp}} \circ \widehat{\varphi_{\mathsf{aux}}}(P_{\mathsf{aux}})$, $\widetilde{Q_{\mathsf{chl}}} = \varphi_{\mathsf{rsp}} \circ \widehat{\varphi_{\mathsf{aux}}}(Q_{\mathsf{aux}})$.

6: $(U_{\mathsf{chl}}, V_{\mathsf{chl}})$, $\mathrm{hint}_{\mathsf{chl}} \leftarrow \mathsf{TorsionBasisToHint}(E_{\mathsf{chl}}, f, e_{\mathsf{rsp}} + 2)$. ▷ [1, Algorithm 4.94]

7: Compute $\widetilde{M_{\mathsf{chl}}}$ according to Eq. (11).

8: $M_{\mathsf{chl}} \leftarrow q^{-1} \cdot \widetilde{M_{\mathsf{chl}}} \pmod{2^{e_{\mathsf{rsp}}}}$.

9: $M_{\mathsf{chl}} \leftarrow \mathsf{NormalizeMatrix}(M_{\mathsf{chl}}, e_{\mathsf{rsp}} + 2)$. ▷ [1, Algorithm 3.10]

10: **return** $(E_{\mathsf{aux}}, M_{\mathsf{chl}}, \mathrm{hint}_{\mathsf{aux}}, \mathrm{hint}_{\mathsf{chl}})$.

4.3 Applications to SQIsign2DPush

We finally apply the technique in Section 3 to SQIsign2DPush. Figure 4 depicts the commutative diagram arising in its response phase. Our optimizations focus on two components, PushRandIsog and FastCommit. The red part of the diagram corresponds to our optimization of PushRandIsog, while the blue part corresponds to our optimization of FastCommit.

Optimized PushRandIsog The original PushRandIsog algorithm [38, Algorithms 1 and 2] requires a two-dimensional discrete logarithm computation over a 3^{e_2}-torsion module in order to recover the kernel of the 3^{e_2}-isogeny ψ''_0.

We retain the notation of [38]. Let

$$\langle P_0, Q_0 \rangle = E_0[2^{e_2}3^{e_3}],$$

and define

$$P_0^1 = [3^{e_3}]P_0,\ Q_0^1 = [3^{e_3}]Q_0,\ \langle P_0^1, Q_0^1 \rangle = E_0[2^{e_2}].$$

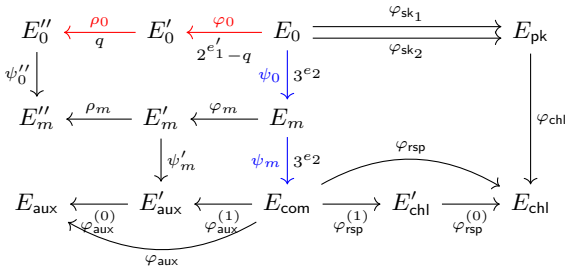


Figure 4: The diagram of SQIsign2DPush.

Let $\psi = \psi_m \circ \psi_0 \colon E_0 \to E_{\mathsf{com}}$, where $\deg(\psi_0) = \deg(\psi_m) = 3^{e_2}$, and let K_0 be a generator of $\ker(\psi_0)$.

For an input degree $q \in (0, 2^{e_1})$, $\gcd(q, 6) = 1$, let $e_1' \le e_1$ be the smallest integer satisfying $2^{e_1'} - q > 0$ and $\gcd(2^{e_1'} - q, 3) = 1$, and set $\delta = e_1 - e_1'$. The upper half of the original PushRandIsog algorithm constructs and evaluates a two-dimensional isogeny $\Phi : E_0 \times E_0'' \to E_0' \times F_0$ whose kernel is

$$\ker(\Phi) = \left\langle [-q2^{\delta}]P_0^1,\ \rho_0 \circ \varphi_0([2^{\delta}]P_0^1) \right\rangle \oplus \left\langle [-q2^{\delta}]Q_0^1,\ \rho_0 \circ \varphi_0([2^{\delta}]Q_0^1) \right\rangle .$$

Evaluating Φ at the points

$$(K_0, \mathcal{O}_{E_0''}), \quad (\mathcal{O}_{E_0}, P_0''), \quad (\mathcal{O}_{E_0}, Q_0'')$$

yields the images $\varphi_0(K_0)$ together with $\big(\widehat{\rho}_0(P_0''),\ \widehat{\rho}_0(Q_0'')\big)$, where $\langle P_0'', Q_0'' \rangle = E_0''[3^{e_2}]$. The original algorithm then expresses $\varphi_0(K_0)$ with respect to the basis $\big(\widehat{\rho}_0(P_0''), \widehat{\rho}_0(Q_0'')\big)$ by solving a two-dimensional discrete logarithm problem and subsequently rescaling the recovered coordinates to obtain $\rho_0 \circ \varphi_0(K_0)$. These computations are exactly the same as those in the POKÉ case. Therefore, the same optimization method can be applied. The optimized algorithm is given in Algorithm 13, which replaces the call to [38, Algorithm 1] on line 4 of [38, Algorithm 2].

Optimized FastCommit In the previous sections, two-dimensional duals were central to optimizing the evaluation of two-dimensional representations. We now turn to one-dimensional duals to optimize FastCommit [19, Algorithm 1], which is needed in the commit phase of SQIsign2DPush. The diagram of FastCommit is illustrated in Figure 5.

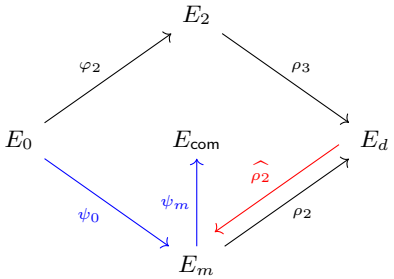


Figure 5: Diagram of FastCommit.

The target isogenies of FastCommit are highlighted in blue in Figure 4 and Figure 5. The algorithm obtains a point $K_d \in E_d[3^{e_2}]$ such that $\langle K_d \rangle = \ker(\widehat{\rho_3})$. It must then recover

Algorithm 13 Optimized subroutine of PushRandIsog

Input: An integer $d \in (0, 2^{e_1})_6$, a generator K_0 of ψ_0, the integer $e_1' \leqslant e_1$ such that $2^{e_1'} - d \in (0, 2^{e_1})_3$, and $\delta := e_1 - e_1'$.

Output: E_m'' and $(\rho_m \circ \varphi_m([2^\delta]P_m^1),\ \rho_m \circ \varphi_m([2^\delta]Q_m^1))$ for a $d(2^{e_1'} - d)$-isogeny $\rho_m \circ \varphi_m : E_m \to E_m''$.

1: Let $\alpha \leftarrow \mathsf{FullRepresentInteger}_{\mathcal{O}_0}\left(d(2^{e_1'} - d)2^\delta 3^{e_2}\right)$. ▷ [22, Algorithm 1]

▷ Write $\alpha = \widehat{\tau_0} \circ \rho_0 \circ \varphi_0$, where $\deg(\varphi_0) = d,\ \deg(\rho_0) = 2^{e_1'} - d,\ \deg(\tau_0) = 2^\delta 3^{e_2}$.

2: Let $\tau_0 : E_0 \to E_0''$ be a $2^\delta 3^{e_2}$-isogeny with kernel $E_0[2^\delta 3^{e_2}] \cap \ker(\widehat{\alpha})$.

3: $R_0'',\ S_0'' \leftarrow \tau_0 \circ \alpha(P_0),\ \tau_0 \circ \alpha(Q_0)$. ▷ $R_0'' = \rho_0 \circ \varphi_0([2^\delta]P_0^1),\ S_0'' = \rho_0 \circ \varphi_0([2^\delta]Q_0^1)$.

4: Compute Φ with kernel $\ker(\Phi) = \langle([-d2^\delta]P_0^1, R_0''), ([-d2^\delta]Q_0^1, S_0'')\rangle$, obtaining

$$\Theta = \left(\left(\theta_j^{A_{E_0 \times E_0''}}(0_{A_{E_0 \times E_0''}})\right)_j, \left\{ \left(\theta_j^{A_i}(0_{A_i})\right)_j \right\}_{i=1}^{a-1}, \left(\theta_j^{A_{E_0' \times F_0'}}(0_{A_{E_0' \times F_0'}})\right)_j \right)$$

along with the inverses of the theta coordinates in Θ, the change-of-basis matrix M, and the symplectic automorphism N_ψ. ▷ $\Phi : E_0 \times E_0'' \to E_0' \times F_0'$.

5: $(K_0',\ _) \leftarrow \Phi(K_0,\ \mathcal{O}_{E_0''})$. ▷ $K_0' = \varphi_0(K_0)$.

6: $K_0'' \leftarrow \mathsf{EvalDualTwoDim}\left((K_0',\ \mathcal{O}_{F_0'}),\ \Theta,\ M,\ N_\psi\right)$. ▷ Algorithm 8, $K_0'' = \rho_0 \circ \varphi_0(K_0)$.

7: Compute a 3^{e_2}-isogeny $\psi_0'' : E_0'' \to E_m''$ with kernel $\langle K_0''\rangle$.

8: $R_m'',\ S_m'' \leftarrow \psi_0''(R_0''),\ \psi_0''(S_0'')$.

9: **return** $E_m'',\ (R_m'',\ S_m'')$.

the kernel of the second 3^{e_2}-isogeny $\psi_m : E_m \to E_{\mathsf{com}}$, which is the pullback isogeny $[\rho_2]^*\widehat{\rho_3}$ or the pushforward isogeny $[\widehat{\rho_2}]_*\widehat{\rho_3}$. Therefore, the kernel of ψ_m is $\ker(\psi_m) = \langle\widehat{\rho_2}(K_d)\rangle$.

The original algorithm generates a basis $\langle P_m, Q_m\rangle = E_m[3^{e_2}]$ and evaluates the basis points under ρ_2, giving $P_d = \rho_2(P_m)$ and $Q_d = \rho_2(Q_m)$. A single-point two-dimensional discrete logarithm problem is then solved to obtain

$$K_d = [u]P_d + [v]Q_d\ ,$$

after which $K_m = [u]P_m + [v]Q_m$ is used as a generator of $\ker(\psi_m)$. This is costly because it requires the generation of a full basis of $E_m[3^{e_2}]$ and a two-dimensional discrete logarithm.

Instead, we compute the dual isogeny $\widehat{\rho_2} : E_d \to E_m$ directly and evaluate it at K_d, giving $K_m = \widehat{\rho_2}(K_d)$. The object we need to compute is therefore precisely the red arrow $\widehat{\rho_2}$ in Figure 5. Instead of the traditional 2-isogeny chain computation, we exploit the intermediate data of the 2-isogeny or 4-isogeny to obtain its dual.

Consider ρ_2 as a composition of e_1 2-isogenies

$$\rho_2 = \eta_{e_1} \circ \cdots \circ \eta_2 \circ \eta_1 : E_d \to E_m$$

where η_i is a 2-isogeny for $i = 1, 2, \cdots, e_1$. If e_1 is even, set $\sigma_r = \eta_{2r} \circ \eta_{2r-1}$, otherwise set $\sigma_r = \eta_{2r+1} \circ \eta_{2r}$ and $\eta = \eta_1$, for $r = 1, 2, \cdots, \lfloor e_1/2 \rfloor$.

We write $K_0^{(2)}$ for the order-2 kernel generator of η, when it exists, and $K_r^{(4)}$ for the order-4 kernel generator of σ_r. Depending on whether $x\left([2]K_r^{(4)}\right)$ or $x\left(K_0^{(2)}\right)$ equals zero, both 4-isogenies and 2-isogenies have two cases. The details are given below.

Generic 4-isogenies We first consider the case where $x\left([2]K_r^{(4)}\right) \neq 0$. Let $(X_r : Z_r)$ denote the x-coordinate of $K_r^{(4)}$ and $(A_{24,r} : C_{24,r}) = (A_r + 2C_r : 4C_r)$, where A_r/C_r is the parameter of the Montgomery curve E_r: the codomain of σ_r.

With the above notation, [36, Proposition 6] states the dual isogeny formula as

$$\widehat{\sigma_r}(X,Z) = \frac{(X_r^2 - Z_r^2)\cdot P_r(X,Z) + (X_r^2 + Z_r^2)\cdot Q_r(X,Z)}{2X_rZ_r \cdot Q_r(X,Z)}, \tag{12}$$

where

$$\begin{aligned} P_r(X,Z) &= (X+Z)^2\Big(C_{24,r}\cdot(X+Z)^2 - (C_{24,r} - A_{24,r})\cdot 4XZ\Big),\\ Q_r(X,Z) &= (C_{24,r} - A_{24,r})\cdot 4XZ\cdot(X-Z)^2. \end{aligned}$$

Therefore, it suffices to compute the generic dual 4-isogeny with the constants

$$\mathcal{C}_r^{(4)} = \big(A_{24,r}\ ,\ C_{24,r}\ ,\ X_r^2 - Z_r^2\ ,\ X_r^2 + Z_r^2\ ,\ 2X_rZ_r\big).$$

Singular 4*-isogenies* We next consider the singular case where $x\left([2]K_r^{(4)}\right) = 0$, which means $x\left(K_r^{(4)}\right) = \pm 1$. When $x\left(K_r^{(4)}\right) = 1$, the 4-isogeny formula with kernel $\left\langle K_r^{(4)}\right\rangle$ is given in [25, Eqs. (4.4)-(4.5)] as

$$\begin{aligned} \sigma_r(X,Z) &= \frac{(X+Z)^2\left((X-Z)^2 + (A_{r-1}+2)XZ\right)}{(2-A_{r-1})XZ(X-Z)^2}\\ &= \frac{(X+Z)^2\left((X+Z)^2 - (2-A_{r-1})XZ\right)}{(2-A_{r-1})XZ(X-Z)^2}\\ &= \frac{P_{r-1}(X,Z)}{Q_{r-1}(X,Z)} \end{aligned} \tag{13}$$

where A_{r-1} is the parameter of the domain of σ_r. Observe that the kernel generator P of $\widehat{\sigma_r}$ also satisfies $x(P) = 1$. Hence, the above formula also applies to $\widehat{\sigma_r}$, upon replacing A_{r-1} by A_r, the parameter of the codomain of σ_r.

Turning to the case $x\left(K_r^{(4)}\right) = -1$, this can be reduced to $x\left(K_r^{(4)}\right) = 1$ by means of

$$\begin{aligned} \sigma_r' &= \iota^{-1}\circ\sigma_r\circ\iota,\\ \widehat{\sigma_r'} &= \iota^{-1}\circ\widehat{\sigma_r}\circ\iota, \end{aligned}$$

where $\iota : E_A \to E_{-A},\ (x,y)\mapsto(-x,iy)$ is an isomorphism with $i^2 = -1$.

Based on the preceding discussion, for the singular 4-isogeny case, all the constants for computing the dual are

$$\mathcal{C}_r^{(4)} = \big(A_{24,r}, C_{24,r}\big).$$

Generic 2*-isogenies* For the generic case where $x\left(K_0^{(2)}\right) \neq 0$, let $(X_0 : Z_0)$ denote the x-coordinate of $K_0^{(2)}$. [36, Proposition 7] provides the dual isogeny formula as

$$\widehat{\eta}(X,Z) = \frac{Z_0(X+Z)^2}{4X_0XZ} = \frac{Z_0(X+Z)^2}{X_0\big((X+Z)^2 - (X-Z)^2\big)}. \tag{14}$$

Hence, in the generic case, computing the dual 2-isogeny only requires the constants

$$\mathcal{C}_0^{(2)} = \big(X_0, Z_0\big).$$

Singular 2-isogenies For the singular case where $x\left(K_0^{(2)}\right) = 0$, set $T_1 = K_0^{(2)} = (0:1)$ and $T_1' = (1:1)$. Let $T_1'' = (r:s)$ be an 8-torsion point above T_1', and $(\gamma:\delta) = \left(4rs : (r-s)^2\right)$. With the above notation, [44, Proposition 3] showed that the dual isogeny formula is given by

$$\widehat{\eta}(X,Z) = \frac{u(X,Z) + 2\delta \cdot v(X,Z)}{u(X,Z) - 2\delta \cdot v(X,Z)} = \frac{\delta X^2 + \gamma XZ}{\delta Z^2 + \gamma XZ} \tag{15}$$

where

$$\begin{aligned} u(X,Z) &= (\gamma+\delta)(X+Z)^2 - (\gamma-\delta)(X-Z)^2, \\ v(X,Z) &= (X+Z)(X-Z). \end{aligned}$$

It follows that in the singular case, we need only the constants

$$\mathcal{C}_0^{(2)} = (\gamma, \delta)$$

to compute the dual 2-isogeny.

Full evaluation of optimized FastCommit Algorithm 14 gives the full optimized evaluation of FastCommit. It combines the dual computation above with the remaining steps of the original procedure.

Algorithm 14 Optimized FastCommit

Input: A basis of $\mathcal{O}_0$ and an isomorphism $\varepsilon : \mathcal{O}_0 \xrightarrow{\sim} \mathrm{End}(E_0)$.

Output: $\psi_0, \psi_m, E_m, E_{\mathsf{com}}, K_0, K_m$, where $\psi_0 : E_0 \to E_m$ and $\psi_m : E_m \to E_{\mathsf{com}}$ are both of degree 3^{e_2} with $\ker(\psi_0) = \langle K_0 \rangle$ and $\ker(\psi_m) = \langle K_m \rangle$.

1: $\gamma \leftarrow \mathsf{FullRepresentInteger}_{\mathcal{O}_0}(2^{e_1}3^{e_2})$. ▷ [22, Algorithm 1], $\varepsilon(\gamma) = \widehat{\psi_0} \circ \widehat{\rho_2} \circ \rho_3 \circ \varphi_2 \in \mathrm{End}(E_0)$.
2: Compute $\varphi_2 : E_0 \to E_2$ with $\ker(\varphi_2) = \ker(\varepsilon(\gamma)) \cap E_0[2^{e_1}]$.
3: Compute R_3, the generator of $\ker(\varepsilon(\gamma)) \cap E_0[3^{e_2}]$, which corresponds to a 3^{e_2}-isogeny $\theta_3 : E_0 \to F_3$.
4: Evaluate $\varphi_2(R_3)$ and find a point $S_3 \in E_2[3^{e_2}]$ such that $E_2[3^{e_2}] = \langle \varphi_2(R_3), S_3 \rangle$.
5: Compute $\rho_3 = [\varphi_2]_* \theta_3$ with $\ker(\rho_3) = \langle \varphi_2(R_3) \rangle$.
6: Evaluate $K_d = \rho_3(S_3)$.
7: Compute $\psi_0 : E_0 \to E_m$ with $\ker(\psi_0) = \langle K_0 \rangle = \ker(\varepsilon(\widehat{\gamma})) \cap E_0[3^{e_2}]$.
8: Compute R_2, the generator of $\ker(\varepsilon(\widehat{\gamma})) \cap E_0[2^{e_1}]$, which corresponds to a 2^{e_1}-isogeny $\theta_2 : E_0 \to F_2$.
9: Compute $\rho_2 = [\psi_0]_* \theta_2$ with $\ker(\rho_2) = \langle \psi_0(R_2) \rangle$, obtaining the set of constants

$$\mathcal{C} = \left(\mathcal{C}_0^{(2)}, \left\{\mathcal{C}_r^{(4)}\right\}_{r=1}^{\lfloor \frac{e_1}{2} \rfloor}\right)$$

for dual isogenies.
10: Evaluate $K_m = \widehat{\rho_2}(K_d)$ with the constants $\mathcal{C}$ using Eqs. (12)–(15).
11: Compute $\psi_m : E_m \to E_{\mathsf{com}}$ with $\ker(\psi_m) = \langle K_m \rangle$.
12: **return** $\psi_0, \psi_m, E_m, E_{\mathsf{com}}, K_0, K_m$.

5 Experimental Results

This section details the parameter sets and the implementation results for the optimized protocols discussed in Section 4. Furthermore, we provide a comprehensive performance comparison between the original POKÉ, PIKE, SQIsign-v3.0, and SQIsign2DPush schemes and their improved versions incorporating our new algorithm for evaluating two-dimensional representations.

5.1 Parameters of the Improved Variants of POKÉ and PIKE

In this subsection, we present the parameters for the optimized variants of POKÉ and PIKE at the NIST-I security level.

Parameters Following the framework established in Section 4, the corresponding parameters of POKÉD and PIKED at the NIST-I level ($\lambda = 128$) are presented below:

POKÉD NIST-I

$$\begin{aligned} p &= 2^{126} \cdot 251 \cdot 337 \cdot 349837 \cdot 158415637294761427561 \cdot \\ &\quad 43176887214720337703096707961986794523309266337 - 1 \\ &= 2 \cdot 3^{158} \cdot 11 \cdot 41953 \cdot 322633 \cdot 2210318753 \cdot D + 1, \\ \log p &= 383, \\ D &= 5 \cdot 151273 \cdot 14248171259089. \end{aligned}$$

PIKED NIST-I

$$\begin{aligned} p &= 2^{126} \cdot 251 \cdot 337 \cdot 349837 \cdot 158415637294761427561 \cdot \\ &\quad 43176887214720337703096707961986794523309266337 - 1 \\ &= 2 \cdot 3^{158} \cdot D + 1, \\ \log p &= 383, \\ D &= 5 \cdot 11 \cdot 41953 \cdot 151273 \cdot 322633 \cdot 2210318753 \cdot 14248171259089. \end{aligned}$$

We use the prime-finding methods in [11, Appendix A] to construct the parameters of POKÉD and PIKED, respectively.

5.2 Implementation Results

This subsection presents the computational efficiency of the improved POKÉ and PIKE variants (POKÉD and PIKED), as well as the SQIsign2D variants (SQIsign-v3.0 and SQIsign2DPush) using our new method, and compares them with the original protocols.

The benchmarks were conducted on an Intel Xeon w3-2423 processor with a base frequency of 2.1 GHz, running Ubuntu 24.04 LTS. To eliminate frequency scaling noise and OS scheduling interference, features such as Turbo Boost and Hyper-Threading were strictly disabled in the BIOS. All code was compiled using CMake in Release mode with the `-O3` and `-march=native` flags.

Performance Comparison for POKÉ and PIKE Variants To ensure a fair and highly accurate comparison, all implementations, including POKÉ and PIKE, were integrated into a unified framework. This framework builds upon the existing open-source C implementations of POKÉ[5] and PIKE[6]. The finite field arithmetic for all protocols was generated using `fiat-crypto`[7]. Furthermore, our implementations are integrated with optimized differential addition chains [32,14,2,9] for scalar multiplications and optimal strategies [15] for one-dimensional isogeny computations using Vélu's formulas [46,17] and Vélu [10].

Figure 6 visually illustrates the CPU cycle count distribution across the three phases for each variant of POKÉ. Table 3 provides the exact timings and the corresponding speedup ratios relative to the baseline POKÉ protocol at the NIST-I security level. The reported cycle counts are averages over 100 executions, each of which averages 100 internal iterations.

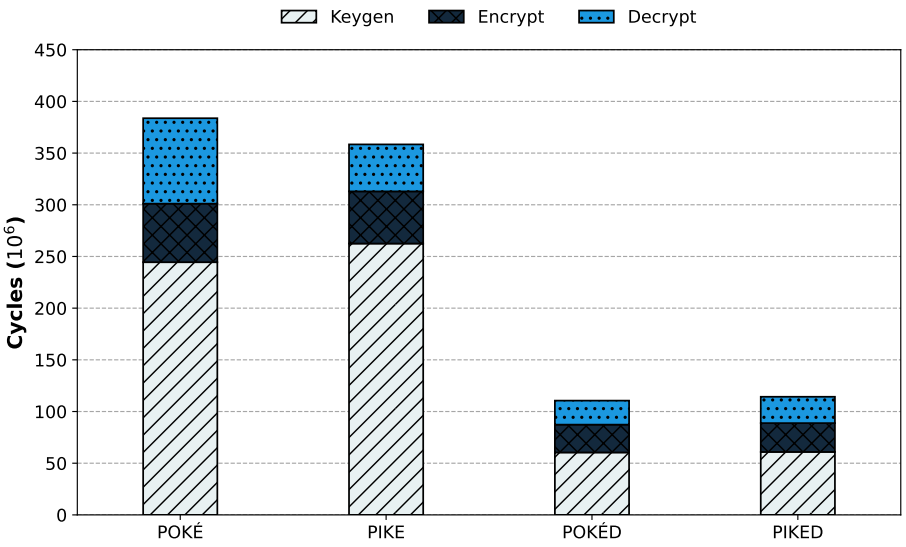


Figure 6: Performance breakdown of the optimized variants integrated with our proposed techniques versus the original POKÉ and PIKE.

Table 3: Performance comparison of the improved variants of POKÉ and PIKE incorporating our new techniques. Timings are presented in millions of clock cycles ($\times 10^6$). The speedup ratio represents the performance gain over the baseline POKÉ protocol, calculated by dividing the baseline cycle count by that of the evaluated variant.

Variants	Keygen		Encrypt		Decrypt		Total	
	Cycles	Ratio	Cycles	Ratio	Cycles	Ratio	Cycles	Ratio
POKÉ [7]	244.45	1.00×	56.63	1.00×	82.64	1.00×	383.73	1.00×
PIKE [11]	262.50	0.93×	50.46	1.12×	45.44	1.82×	358.40	1.07×
POKÉD	60.41	4.05×	26.91	2.10×	23.23	3.56×	110.55	3.47×
PIKED	60.91	4.01×	27.87	2.03×	25.49	3.24×	114.28	3.36×

As shown in Figure 6 and Table 3, POKÉD and PIKED consistently outperform their respective original schemes across the key generation, encryption, and decryption phases. In particular, POKÉD reduces the total cost of POKÉ from 383.73 to 110.55 million cycles, achieving a 3.47× overall speedup. The key-generation, encryption, and decryption costs correspond to speedups of 4.05×, 2.10×, and 3.56×, respectively.

Likewise, PIKED reduces the total cost of PIKE from 358.40 to 114.28 million cycles, yielding a 3.14× overall speedup. Its key-generation, encryption, and decryption costs decrease from 262.50, 50.46, and 45.44 million cycles to 60.91, 27.87, and 25.49 million cycles, respectively, giving phase-wise speedups of 4.31×, 1.81×, and 1.78×. Compared with POKÉ, PIKED attains the 3.36× overall speedup reported in Table 3.

As illustrated in Figure 6, key generation remains the most expensive phase but also exhibits the largest absolute reduction in cycles. Moreover, POKÉD achieves a slightly lower overall cost than PIKED, with 110.55 versus 114.28 million cycles. In conclusion, these

[5] https://github.com/gusgkr0117/poke
[6] https://github.com/Kaizhan-Lin/PIKE-C-Implementation
[7] https://github.com/mit-plv/fiat-crypto

results confirm the effectiveness of our proposed two-dimensional dual isogeny evaluation algorithm in accelerating both the key-generation and decryption operations.

Performance Comparison for SQIsign2D Variants The benchmarks were obtained by modifying the open-source C implementations of SQIsign-v3.0[8] and SQIsign2DPush[9]. Table 4 reports the signing performance of the original and optimized variants of SQIsign-v3.0 and SQIsign2DPush across the three NIST security levels. Our optimizations yield consistent speedups in all configurations. For SQIsign-v3.0, the speedup is modest, ranging from 1.04× to 1.05×. In contrast, SQIsign2DPush benefits more substantially, with speedups between 1.11× and 1.13×. The reported cycle counts are averaged over 10,000 iterations.

Table 4: Signing performance comparison of SQIsign2D variants incorporating our new method. Timings are presented in millions of clock cycles ($\times 10^6$). The speedup ratio represents the performance gain over the corresponding baseline, calculated by dividing the baseline cycle count by that of the evaluated variant.

Schemes	Levels	Original	Ours	Ratio
SQIsign-v3.0 [1]	NIST-I	126.85	120.48	1.05×
	NIST-III	463.34	443.95	1.04×
	NIST-V	815.03	778.87	1.05×
SQIsign2DPush [38]	NIST-I	93.92	83.36	1.13×
	NIST-III	244.73	219.19	1.12×
	NIST-V	527.75	474.39	1.11×

In addition to the overall signing timings, we further analyze the elliptic-curve operations in the signing procedure of SQIsign2DPush, i.e., the elliptic-curve computations other than isogeny operations, including basis generation, scalar multiplication, and two-dimensional discrete logarithm problems. Table 5 compares the cost of these operations between the original and optimized algorithms. The reported CPU times are also averaged over 10,000 iterations.

Table 5: Performance comparison of elliptic-curve operations in the signing procedure of SQIsign2DPush. Timings are presented in CPU time (ms). The speedup ratio is calculated by dividing the baseline CPU time by that of the evaluated variant.

Levels	Original	Ours	Ratio
NIST-I	14.26	9.77	1.46×
NIST-III	35.53	24.36	1.46×
NIST-V	72.09	49.64	1.45×

The elliptic-curve operations achieve a consistent speedup of 1.45× to 1.46× across all three NIST security levels. This speedup is mainly attributed to the optimized PushRandIsog and the optimized FastCommit. To make the resulting algorithmic changes explicit, we compare the operations performed in the original and optimized versions below.

[8] https://github.com/SQIsign/the-sqisign/tree/nist-v3
[9] https://github.com/isec-tut/SQIsign2DPush

For PushRandIsog, the original and optimized algorithms differ in the computation of ρ_0 in Figure 4. To push a point $K_0 \in E_0[3^{e_2}]$ from E_0 to E_0'', the original algorithm proceeds as follows:

1. Compute a two-dimensional isogeny $\Phi : E_0 \times E_0'' \to E_0' \times F_0'$;
2. Generate a basis of $E_0''[3^{e_2}]$, writing $\langle P_0'', Q_0'' \rangle$;
3. Evaluate $K_0' = \varphi_0(K_0), P_0' = \widehat{\rho_0}(P_0''), Q_0' = \widehat{\rho_0}(Q_0'')$ with

$$\begin{aligned}(K_0', *) &= \Phi(K_0, \mathcal{O}_{E_0''}),\\ (P_0', *) &= \Phi(\mathcal{O}_{E_0}, P_0''),\\ (Q_0', *) &= \Phi(\mathcal{O}_{E_0}, Q_0'');\end{aligned}$$

4. Solve a two-dimensional discrete logarithm problem to obtain (u, v) such that

$$K_0' = [u]P_0' + [v]Q_0';$$

5. Reconstruct K_0'' by $[u]P_0'' + [v]Q_0''$.

In our optimized algorithm, the intermediate data of Φ are stored to enable evaluation via $\widehat{\Phi}$. Consequently, the time-consuming operations, namely basis generation, discrete logarithm computation, and point reconstruction, are replaced by directly evaluating the dual two-dimensional isogeny $\widehat{\Phi}$ at K_0' with only a small computational overhead.

For FastCommit, the main difference lies in the computation of $\widehat{\rho_2}$ in Figure 5. In the original version, $\widehat{\rho_2}(K_d)$ is obtained indirectly:

1. Generate a basis of $E_m[3^{e_2}]$, writing $\langle U_m, V_m \rangle$;
2. Evaluate $U_d = \rho_2(U_m)$ and $V_d = \rho_2(V_m)$;
3. Solve a two-dimensional discrete logarithm problem to obtain (u, v) such that

$$K_d = [u]U_d + [v]V_d;$$

4. Reconstruct K_m by $[u]U_m + [v]V_m$.

Our optimized version instead exploits the intermediate data of the forward 2-isogeny to compute $\widehat{\rho_2}$ directly and evaluate it at K_d. As with PushRandIsog, this eliminates costly computations at low additional cost.

6 Conclusion and Future Work

In this work, we introduced a novel algorithm for the efficient evaluation of two-dimensional representations. By leveraging the properties of dual isogenies, our approach avoids the requirement for computationally expensive two-dimensional discrete logarithms and basis generation, thereby achieving significant efficiency improvements. When coupled with the Superglue technique, this methodology not only accelerates computations but also provides greater flexibility in parameter selection for POKÉ variants. Furthermore, our proposed algorithm has been used to optimize the signature schemes SQIsign2DPush [38] and SQIsign-v3.0 [1], and has the potential to accelerate other protocols that rely on two-dimensional representation evaluation as a fundamental building block. Ultimately, this work provides a new perspective on evaluating two-dimensional representations, establishing our approach as a basic tool for broader protocol design. Future work will focus on leveraging the techniques proposed in this paper to construct a wider array of efficient isogeny-based cryptographic schemes.

Algorithm 15 MatrixInverseProj(M)

Input: A 4×4 matrix M with elements $m_{i,j}$ for $0 \le i, j \le 3$.
Output: The adjugate matrix R, projectively equivalent to M^{-1}.

$s_0 \leftarrow m_{0,0}m_{1,1} - m_{0,1}m_{1,0}$, $s_1 \leftarrow m_{0,0}m_{1,2} - m_{0,2}m_{1,0}$ ▷ 6 minors from top two rows
$s_2 \leftarrow m_{0,0}m_{1,3} - m_{0,3}m_{1,0}$, $s_3 \leftarrow m_{0,1}m_{1,2} - m_{0,2}m_{1,1}$
$s_4 \leftarrow m_{0,1}m_{1,3} - m_{0,3}m_{1,1}$, $s_5 \leftarrow m_{0,2}m_{1,3} - m_{0,3}m_{1,2}$
$c_0 \leftarrow m_{2,0}m_{3,1} - m_{2,1}m_{3,0}$, $c_1 \leftarrow m_{2,0}m_{3,2} - m_{2,2}m_{3,0}$ ▷ 6 minors from bottom two rows
$c_2 \leftarrow m_{2,0}m_{3,3} - m_{2,3}m_{3,0}$, $c_3 \leftarrow m_{2,1}m_{3,2} - m_{2,2}m_{3,1}$
$c_4 \leftarrow m_{2,1}m_{3,3} - m_{2,3}m_{3,1}$, $c_5 \leftarrow m_{2,2}m_{3,3} - m_{2,3}m_{3,2}$
$r_{0,0} \leftarrow m_{1,1}c_5 - m_{1,2}c_4 + m_{1,3}c_3$, $r_{1,0} \leftarrow -m_{1,0}c_5 + m_{1,2}c_2 - m_{1,3}c_1$ ▷ Column 0 of R
$r_{2,0} \leftarrow m_{1,0}c_4 - m_{1,1}c_2 + m_{1,3}c_0$, $r_{3,0} \leftarrow -m_{1,0}c_3 + m_{1,1}c_1 - m_{1,2}c_0$
$r_{0,1} \leftarrow -m_{0,1}c_5 + m_{0,2}c_4 - m_{0,3}c_3$, $r_{1,1} \leftarrow m_{0,0}c_5 - m_{0,2}c_2 + m_{0,3}c_1$ ▷ Column 1 of R
$r_{2,1} \leftarrow -m_{0,0}c_4 + m_{0,1}c_2 - m_{0,3}c_0$, $r_{3,1} \leftarrow m_{0,0}c_3 - m_{0,1}c_1 + m_{0,2}c_0$
$r_{0,2} \leftarrow m_{3,1}s_5 - m_{3,2}s_4 + m_{3,3}s_3$, $r_{1,2} \leftarrow -m_{3,0}s_5 + m_{3,2}s_2 - m_{3,3}s_1$ ▷ Column 2 of R
$r_{2,2} \leftarrow m_{3,0}s_4 - m_{3,1}s_2 + m_{3,3}s_0$, $r_{3,2} \leftarrow -m_{3,0}s_3 + m_{3,1}s_1 - m_{3,2}s_0$
$r_{0,3} \leftarrow -m_{2,1}s_5 + m_{2,2}s_4 - m_{2,3}s_3$, $r_{1,3} \leftarrow m_{2,0}s_5 - m_{2,2}s_2 + m_{2,3}s_1$ ▷ Column 3 of R
$r_{2,3} \leftarrow -m_{2,0}s_4 + m_{2,1}s_2 - m_{2,3}s_0$, $r_{3,3} \leftarrow m_{2,0}s_3 - m_{2,1}s_1 + m_{2,2}s_0$
return R ▷ Cost: 72**M**

A Projective Inverse of a Four-by-Four Matrix

This appendix presents the algorithm for the computation of the projective inverse of a 4×4 matrix, which is summarized in Algorithm 15.

References

1. Aardal, M.A., Adj, G., Aranha, D.F., Basso, A., Borin, G., Canales Martínez, I.A., Chávez-Saab, J., Corte-Real Santos, M., Dartois, P., De Feo, L., Duparc, M., Espitau, T., Eriksen, J.K., Fouotsa, T.B., Gazzoni Filho, D.L., Hess, B., Invernizzi, R., Kohel, D., Leroux, A., Longa, P., Maino, L., Meyer, M., Mula, M., Nakagawa, K., Onuki, H., Panny, L., Patranabis, S., Petit, C., Pope, G., Reijnders, K., Robert, D., Rodríguez-Henríquez, F., Schaeffler, S., Vercauteren, F., Wallet, A., Wesolowski, B., van Woerden, W.: SQIsign. Tech. rep., National Institute of Standards and Technology (2026), https://sqisign.org
2. Banegas, G., Bernstein, D.J., Campos, F., Chou, T., Lange, T., Meyer, M., Smith, B., Sotáková, J.: CTIDH: Faster Constant-Time CSIDH. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(4), 351–387 (Aug 2021)
3. Basso, A., Borin, G., Castryck, W., Santos, M.C., Invernizzi, R., Leroux, A., Maino, L., Vercauteren, F., Wesolowski, B.: PRISM: simple and compact identification and signatures from large prime degree isogenies. In: Jager, T., Pan, J. (eds.) Public-Key Cryptography - PKC 2025 - 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Røros, Norway, May 12-15, 2025, Proceedings, Part III. Lecture Notes in Computer Science, vol. 15676, pp. 300–332. Springer (2025)
4. Basso, A., Borin, G., Castryck, W., Santos, M.C., Invernizzi, R., Leroux, A., Maino, L., Vercauteren, F., Wesolowski, B.: PRISM with a pinch of salt: Simple, efficient, and strongly unforgeable signatures from isogenies. J. Cryptol. **39**(4), 30 (2026)
5. Basso, A., Dartois, P., Feo, L.D., Leroux, A., Maino, L., Pope, G., Robert, D., Wesolowski, B.: SQIsign2D-West - The Fast, the Small, and the Safer. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part III. pp. 339–370. Lecture Notes in Computer Science, Springer (2024)

6. Basso, A., Fouotsa, T.B.: New SIDH Countermeasures for a More Efficient Key Exchange. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VIII. pp. 208–233. Lecture Notes in Computer Science, Springer (2023)
7. Basso, A., Maino, L.: POKÉ: A Compact and Efficient PKE from Higher-Dimensional Isogenies. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part II. pp. 94–123. Lecture Notes in Computer Science, Springer (2025)
8. Basso, A., Maino, L., Pope, G.: FESTA: Fast Encryption from Supersingular Torsion Attacks. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VII. pp. 98–126. Lecture Notes in Computer Science, Springer (2023)
9. Bernstein, D.J., Cottaar, J., Lange, T.: Searching for Differential Addition Chains. Research in Number Theory **11**(2), 45 (2025)
10. Bernstein, D.J., De Feo, L., Leroux, A., Smith, B.: Faster Computation of Isogenies of Large Prime Degree. ANTS XIV Proceedings of the Fourteenth Algorithmic Number Theory Symposium, Open Book Series **4**(1), 39–55 (2020)
11. Cai, S., Chen, M., Lai, Y., Lin, K.: PIKE: Faster Isogeny-Based Public Key Encryption with Pairing-Assisted Decryption. In: Bai, S., Persichetti, E. (eds.) Public-Key Cryptography - PKC 2026 - 29th IACR International Conference on Practice and Theory of Public-Key Cryptography, West Palm Beach, FL, USA, May 25-28, 2026, Proceedings, Part III. Lecture Notes in Computer Science, vol. 16553, pp. 58–91. Springer (2026)
12. Campagna, M., Costello, C., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Renes, J., Urbanik, D., et al.: Supersingular Isogeny Key Encapsulation (2019)
13. Castryck, W., Decru, T.: An Efficient Key Recovery Attack on SIDH. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V. pp. 423–447. Lecture Notes in Computer Science, Springer (2023)
14. Cervantes-Vázquez, D., Chenu, M., Chi-Domínguez, J., Feo, L.D., Rodríguez-Henríquez, F., Smith, B.: Stronger and Faster Side-Channel Protections for CSIDH. In: Schwabe, P., Thériault, N. (eds.) Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11774, pp. 173–193. Springer (2019)
15. Chi-Domínguez, J., Rodríguez-Henríquez, F.: Optimal Strategies for CSIDH. Adv. Math. Commun. **16**(2), 383–411 (2022)
16. Costello, C.: B-SIDH: Supersingular Isogeny Diffie-Hellman Using Twisted Torsion. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II. pp. 440–463. Lecture Notes in Computer Science, Springer (2020)
17. Costello, C., Hisil, H.: A Simple and Compact Algorithm for SIDH with Arbitrary Degree Isogenies. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10625, pp. 303–329. Springer (2017)
18. Dartois, P.: Fast Computation of 2-Isogenies in Dimension 4 and Cryptographic Applications. Journal of Algebra **683**, 449–514 (2025)
19. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: SQIsignHD: New Dimensions in Cryptography. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 3–32. Springer Nature Switzerland, Cham (2024)
20. 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: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the

Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part III. Lecture Notes in Computer Science, vol. 15486, pp. 304–338. Springer (2024)
21. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. pp. 64–93. Springer International Publishing, Cham (2020)
22. De Feo, L., Leroux, A., Longa, P., Wesolowski, B.: New Algorithms for the Deuring Correspondence. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 659–690. Springer Nature Switzerland, Cham (2023)
23. Duparc, M.: Superglue: Fast Formulae for (2,2)-Gluing Isogenies. In: Hanaoka, G., Yang, B. (eds.) Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 16248, pp. 372–400. Springer (2025)
24. Duparc, M., Fouotsa, T.B.: SQIPrime: A Dimension 2 Variant of SQISignHD with Non-smooth Challenge Isogenies. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part III. pp. 396–429. Lecture Notes in Computer Science, Springer (2024)
25. Feo, L., Jao, D., Plût, J.: Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. Journal of Mathematical Cryptology **8**, 209–247 (2014)
26. Fouotsa, T.B., Moriya, T., Petit, C.: M-SIDH and MD-SIDH: Countering SIDH Attacks by Masking Information. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V. pp. 282–309. Lecture Notes in Computer Science, Springer (2023)
27. Jao, D., Feo, L.D.: Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In: Yang, B. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings. pp. 19–34. Lecture Notes in Computer Science, Springer (2011)
28. Kani, E.: The Number of Curves of Genus Two with Elliptic Differentials. Journal für die reine und angewandte Mathematik **1997**(485), 93–122 (1997)
29. Kim, H., Hong, S., Kim, S.: INKE: Fast Isogeny-Based PKE using Intermediate Curves. IACR Cryptol. ePrint Arch. **2025**, 1458 (2025)
30. Lin, J., Wang, S., Zhao, C.: A Note on (2,2)-Isogenies via Theta Coordinates. Finite Fields Their Appl. **99**, 102496 (2024)
31. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A Direct Key Recovery Attack on SIDH. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V. pp. 448–471. Lecture Notes in Computer Science, Springer (2023)
32. Montgomery, P.L.: Evaluating recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains (1992), available at https://cr.yp.to/bib/1992/montgomery-lucas.pdf
33. Moriya, T.: On Weak Keys of POKÉ, https://eprint.iacr.org/2026/1002, accepted for publication at Asiacrypt 2026.
34. Moriya, T., Stopar, M.: LIT-SiGamal: An Efficient Isogeny-Based PKE Based on a LIT Diagram. In: Hanaoka, G., Yang, B. (eds.) Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 16248, pp. 275–306. Springer (2025)
35. Mumford, D.: On the Equations Defining Abelian Varieties. I. Invent. Math. **1**, 287–354 (1966)
36. Naehrig, M., Renes, J.: Dual Isogenies and Their Application to Public-Key Compression for Isogeny-Based Cryptography. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 243–272. Springer International Publishing, Cham (2019)
37. Nakagawa, K., Onuki, H.: QFESTA: Efficient Algorithms and Parameters for FESTA Using Quaternion Algebras. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August

18-22, 2024, Proceedings, Part V. pp. 75–106. Lecture Notes in Computer Science, Springer (2024)
38. Nakagawa, K., Onuki, H.: SQIsign2DPush: Faster Signature Scheme Using 2-Dimensional Isogenies. In: Daemen, J., Thomé, E. (eds.) Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 16544, pp. 511–540. Springer (2026)
39. Nakagawa, K., Onuki, H., Castryck, W., Chen, M., Invernizzi, R., Lorenzon, G., Vercauteren, F.: SQIsign2D-East: A New Signature Scheme Using 2-Dimensional Isogenies. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part III. pp. 272–303. Lecture Notes in Computer Science, Springer (2024)
40. Ray, D.: ECLIPSE: Strongly Unforgeable Isogeny Signatures from the Prime-Degree Variant of PRISM. Cryptology ePrint Archive, Paper 2026/2312 (2026), https://eprint.iacr.org/2026/2312
41. Robert, D.: Fonctions thêta et applications à la cryptographie. Ph.D. thesis, Université Henri Poincaré-Nancy I (2010)
42. Robert, D.: Breaking SIDH in Polynomial Time. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V. pp. 472–503. Lecture Notes in Computer Science, Springer (2023)
43. Robert, D.: Some Notes on Algorithms for Abelian Varieties. Cryptology ePrint Archive, Paper 2024/406 (2024), https://eprint.iacr.org/2024/406
44. Robert, D., Sarkis, N.: Computing 2-Isogenies Between Kummer Lines. IACR Communications in Cryptology **1**(1) (2024). https://doi.org/10.62056/abvua69p1
45. Silverman, J.H.: The Arithmetic of Elliptic Curves, vol. 106. Springer (2009)
46. Vélu, J.: Isogénies entre courbes elliptiques. Comptes Rendus de l'Académie des Sciences de Paris **273**, 238–241 (1971)
47. Xu, Z., Lin, K., Zhao, C., Ouyang, Y.: SQIsign2D^2: New SQIsign2D Variant by Leveraging Power Smooth Isogenies in Dimension One. In: Hanaoka, G., Yang, B. (eds.) Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part IV. pp. 341–371. Lecture Notes in Computer Science, Springer (2025)