# A Faster Software Implementation of SQIsign

Kaizhan Lin, Weize Wang, Zheng Xu, and Chang-An Zhao

***Abstract*— Isogeny-based cryptography is famous for its short key size. As one of the most compact digital signatures, SQIsign (Short Quaternion and Isogeny Signature) is attractive among post-quantum cryptography, but it is inefficient compared to other post-quantum competitors because of complicated procedures in the ideal-to-isogeny translation, which is the efficiency bottleneck of the signing phase. In this paper, we recall the current implementation of SQIsign and mainly focus on how to improve the execution of the ideal-to-isogeny translation in SQIsign. Specifically, we demonstrate how to utilize the reduced Tate pairing to save one of the two elliptic curve discrete logarithms. In addition, the efficient implementation of the remainder discrete logarithm computation is explored. We speed up other procedures in the ideal-to-isogeny translation with various techniques as well. It should be noted that our improvements also benefit the performance of key generation and verification in SQIsign. In the instantiation with $p_{1973}$, the improvements lead to a speedup of 5.47%, 8.80% and 25.34% for key generation, signature and verification, respectively.**

***Index Terms*— Isogeny-based cryptography, SQIsign, pairings, discrete logarithms.**

## I. Introduction

AMONG post-quantum cryptography, isogeny-based cryptography is famous for its short key size. In the last two decades, various isogeny-based key exchange schemes were proposed, such as SIDH [1], [2], CSIDH [3] and OSIDH [4], [5]. These protocols also motivate cryptographers to construct digital signatures. Currently, there are mainly three kinds of isogeny-based signatures: SIDH-based [6], [7], CSIDH-based [8], [9], [10], and quaternion-based [11], [12], [13], [14].

SQIsign (Short Quaternion and Isogeny Signature) was first introduced by De Feo, Kohel, Leroux, Petit and Wesolowski [12]. Compared with other isogeny-based signatures, the bitlength of the prime field characteristic used in SQIsign is relatively small. Besides, the public key of SQIsign does not reveal torsion point information (and thus it is not vulnerable to the Castryck-Decru-Maino-Martindale-Robert attacks [15], [16], [17]). Furthermore, the signer needs to respond for each challenge bit respectively in most isogeny-based signatures, but there is no need for such a procedure in SQIsign. This makes SQIsign, as one of the most compact signatures, highly competitive in the field of post-quantum cryptography.

SQIsign is obtained by applying the Fiat-Shamir transform [18] to an identification protocol. The signing phase of SQIsign mainly involves two procedures: ideal generation with the SigningKLPT algorithm and the ideal-to-isogeny translation.

The SigningKLPT algorithm, which builds upon the KLPT algorithm initially introduced by Kohel et al. [19], takes a left ideal $I$ as input and outputs another left ideal $J$ of a smooth power reduced norm which is equivalent to $I$. After obtaining $J$, the signer needs to translate it into the corresponding isogeny $\varphi_J$ and compress it to be a part of the signature. The translation from the ideal $J$ to the isogeny $\varphi_J$ is the efficiency bottleneck of SQIsign, since it involves expensive procedures such as large degree isogeny computations. Recently, De Feo et al. [13] proposed a novel approach to speed up the ideal-to-isogeny translation. Besides isogeny computations, the state-of-the-art contains torsion point generation, discrete logarithm computations, etc.

In this paper, we explore the current SQIsign implementation and further accelerate it by utilizing various techniques, especially the performance of the ideal-to-isogeny translation, as we summarize in the following:

1. In [13], each step of the new algorithm for the ideal-to-isogeny translation requires computing two elliptic curve discrete logarithms, i.e.,

$$\theta(P) = [x_1]P + [x_2]Q,$$
$$\theta(Q) = [x_3]P + [x_4]Q,$$

where $P, Q \in E[2^a]$, and the endomorphism $\theta$ has reduced norm coprime to 2. For efficiency, the previous work used an $x$-only arithmetic to obtain the absolute values of $x_1$, $x_2$, $x_3$ and $x_4$, then employed the reduced trace of the endomorphism to determine their signs.

Manuscript received 6 September 2023; revised 2 March 2024; accepted 1 July 2024. Date of publication 4 July 2024; date of current version 20 August 2024. This work was supported in part by Guangdong Major Project of Basic and Applied Basic Research under Grant 2019B030302008, in part by the National Natural Science Foundation of China under Grant 61972428 and Grant 12371013, in part by the Innovation Program for Quantum Science and Technology under Grant 2021ZD0302902, and in part by Anhui Initiative in Quantum Information Technologies under Grant AHY150200. *(Corresponding author: Zheng Xu.)*

Kaizhan Lin is with the Department of Mathematics, School of Mathematics, Sun Yat-sen University, Guangzhou, Guangdong 510275, China (e-mail: linkzh5@mail2.sysu.edu.cn).

Weize Wang is with the School of Computer Science, Fudan University, Shanghai 200433, China (e-mail: wzwang23@m.fudan.edu.cn).

Zheng Xu is with Hefei National Laboratory, University of Science and Technology of China, Hefei, Anhui 230088, China (e-mail: xuzheng1@mail.ustc.edu.cn).

Chang-An Zhao is with the School of Mathematics, Sun Yat-sen University, Guangzhou, Guangdong 510275, China, also with the State Key Laboratory of Information Security (Institute of Information Engineering), Beijing 100190, China, and also with Guangdong Key Laboratory of Information Security, Guangzhou, Guangdong 510006, China (e-mail: zhaochan3@mail.sysu.edu.cn).

Communicated by T. Johansson, Associate Editor for Sequences and Cryptography.

Digital Object Identifier 10.1109/TIT.2024.3423675

We claim that one can eliminate the second elliptic curve discrete logarithm computation by fully exploiting the properties of $\theta$. Additionally, we provide a much more efficient approach to solve the other elliptic curve discrete logarithm by utilizing pairing computations and discrete logarithm computations over the finite field $\mathbb{F}_{p^2}$. The experimental results show that our method offers a $3.6\times$ speedup. It should be noted that the improvement benefits not only the signing phase but also key generation.

2. We introduce new techniques to optimize other procedures in the ideal-to-isogeny translation. In particular, our algorithm offers a speedup of approximately $2\times$ for torsion point generation. Besides, we improve the performance of isogeny computations in SQIsign, leading to considerable improvements. We also demonstrate that one can accelerate the first execution of the ideal-to-isogeny translation in the signing phase via precomputation in the key generation phase. It may enlarge the cost of key generation, but reduces the signing cost. This would be preferred when the signer intends to sign a number of messages with the same secret key.
3. Based on [20], we complied and benchmarked our code. The experimental results show that our techniques yield a significant acceleration of all the above procedures. Besides, we not only enhance the signing phase but also improve key generation and verification of SQIsign. The experimental results show that the instantiation of key generation, signature and verification with our techniques are 5.47%, 8.80% and 25.34% faster than those of the state-of-the-art, respectively. In particular, when adapting the precomputation technique in the key generation phase, the performance of the signing phase can be up to 18.02% faster than that of the previous work.

Recently, Dartois et al. introduced SQIsignHD [14], a novel isogeny-based signature inspired by SQIsign. While the signing phase of SQIsignHD is simpler and more efficient compared to SQIsign, its verification phase is much more expensive. Our work bridges the gap between the signing performance of SQIsign and SQIsignHD, and achieves a faster verification of SQIsign.

The remainder of this paper is organized as follows. In Section II we explain some mathematical concepts and review SQIsign, especially the ideal-to-isogeny translation. Section III presents an efficient approach to compute discrete logarithms in the ideal-to-isogeny translation. In Section IV we provide other improvements to speed up the performance. Finally, we report experimental results and give a performance comparison between ours and the previous work in Section V and conclude in Section VI.

## II. Notations and Preliminaries

In this section, we recap the required background that will be used throughout the paper. We also provide a brief review of SQIsign and the implementation of the ideal-to-isogeny translation.

### A. Mathematical Background

In this subsection we recall supersingular elliptic curves, isogenies and ideals in quaternion algebras, for more in-deep details see [21] and [22].

*Elliptic Curves and Isogenies:* Elliptic curves are nonsingular projective curves with genus 1. We denote the infinity point of an elliptic curve $E$ as $\infty_E$. An isogeny $\varphi : E_1 \to E_2$ is a non-trivial morphism that sends $\infty_{E_1}$ to $\infty_{E_2}$. If the degree of isogeny $\varphi$ equals the size of $\ker(\varphi)$, we call $\varphi$ a separable isogeny. We abbreviate a separable isogeny of degree $\ell$ as $\ell$-isogeny. For any subgroup $G$ of an elliptic curve $E$, we can compute an isogeny with kernel $G$ by Vélu's formula [23], [24]. For any isogeny $\varphi$ from $E_1$ to $E_2$, there exists a unique isogeny $\hat{\varphi}$ from $E_2$ to $E_1$ such that $\hat{\varphi} \circ \varphi = \varphi \circ \hat{\varphi} = [\deg(\varphi)]$. We call $\hat{\varphi}$ the dual isogeny of $\varphi$. Let $\varphi_1$ and $\varphi_2$ be two isogenies from $E_1$ to $E_2$ with degree $\ell$. We say that $\varphi_1$ and $\varphi_2$ are equivalent if $\ker(\varphi_1) = \ker(\varphi_2)$.

An endomorphism of an elliptic curve $E$ is either the constant zero morphism or an isogeny from $E$ to itself. The set of endomorphisms of $E$ defined over $\overline{\mathbb{F}_p}$ forms a ring under addition and composition. This ring, denoted by $\mathrm{End}_{\overline{\mathbb{F}_p}}(E)$ or $\mathrm{End}(E)$ for brevity, is called the endomorphism ring. Since the scalar multiplication $[n]$ is an isogeny, we have $\mathbb{Z} \subseteq \mathrm{End}(E)$. Moreover, if $\mathrm{End}(E) \neq \mathbb{Z}$, we say that $E$ has complex multiplication.

Every elliptic curve over a finite field has complex multiplication, and they can be divided into two types by endomorphism rings. An elliptic curve $E$ is said to be ordinary if $\mathrm{End}_{\overline{\mathbb{F}_p}}(E)$ is isomorphic to an order in a quadratic imaginary field. Conversely, if $\mathrm{End}_{\overline{\mathbb{F}_p}}(E)$ is isomorphic to a maximal order in a quaternion algebra, then the elliptic curve $E$ is said to be supersingular.

*Orders and Ideals in Quaternion Algebra:* A quaternion algebra over $\mathbb{Q}$ ramified only at $p$ and $\infty$ is of the form $B_{p,\infty} = \mathbb{Q} + \mathbb{Q}i + \mathbb{Q}j + \mathbb{Q}k$, where $i^2 = -q$, $j^2 = -p$ and $k = ij = -ji$ for some integer $q$. For any $\alpha = a_1 + a_2 i + a_3 j + a_4 k \in B_{p,\infty}$, the canonical involution is the map sending $\alpha$ to $\bar{\alpha} = a_1 - a_2 i - a_3 j - a_4 k$. The reduced trace and the reduced norm of $\alpha$ are respectively defined by

$$\mathrm{Trd}(\alpha) = \alpha + \bar{\alpha} = 2a_1,$$
$$\mathrm{Nrd}(\alpha) = \alpha\bar{\alpha} = {a_1}^2 + {a_2}^2 + p{a_3}^2 + p{a_4}^2.$$

An order in $B_{p,\infty}$ is a full-rank lattice and it is also a subring. We call an order maximal when it is not contained in any other larger order. The endomorphism rings of supersingular elliptic curves over $\overline{\mathbb{F}_p}$ are isomorphic to maximal orders in $B_{p,\infty}$.

Let $\mathcal{O}$ be a maximal order. A full-rank lattice $I \subseteq \mathcal{O}$ is a left $\mathcal{O}$-ideal if $\mathcal{O}I \subseteq I$, and it is a right $\mathcal{O}$-ideal if $I\mathcal{O} \subseteq I$. For any left ideal $I$ of a maximal order $\mathcal{O}$ in $B_{p,\infty}$, define the left order and right order of $I$ as

$$\mathcal{O}_L(I) = \{x \in B_{p,\infty} \mid xI \subseteq I\},$$
$$\mathcal{O}_R(I) = \{x \in B_{p,\infty} \mid Ix \subseteq I\}.$$

Note that $\mathcal{O}_L(I)$ and $\mathcal{O}_R(I)$ are also maximal orders. We say that $I$ connects $\mathcal{O}_L(I)$ and $\mathcal{O}_R(I)$, and the corresponding Eichler order of $I$ is defined as $\mathfrak{O} = \mathcal{O}_L(I) \cap \mathcal{O}_R(I)$. The reduced norm of $I$ is given by $\mathrm{Nrd}(I) = \gcd(\{\mathrm{Nrd}(\alpha) \mid \alpha \in I\})$. The conjugate of $I$, denoted by $\bar{I}$, is the set of conjugates of elements of $I$ satisfying $I\bar{I} = \mathrm{Nrd}(I)\mathcal{O}_L(I)$ and $\bar{I}I = \mathrm{Nrd}(I)\mathcal{O}_R(I)$. Two left ideals $I$ and $J$ in $\mathcal{O}$ are equivalent if there exists $\alpha \in B_{p,\infty}^{\times}$ such that $J = I\alpha$, and we denote the set of such classes by $\mathrm{cl}(\mathcal{O})$.

*Isogeny Graphs:* The $\ell$-isogeny graph $\mathcal{G}_\ell(\overline{\mathbb{F}_p})$ is a graph whose vertices represent $\overline{\mathbb{F}_p}$-isomorphism classes $[E]$ of supersingular elliptic curves defined over $\overline{\mathbb{F}_p}$. All the elliptic curves in the $\overline{\mathbb{F}_p}$-isomorphism class have the same $j$-invariant. An edge in this graph represents an equivalent class of $\ell$-isogenies. From [25], the $\ell$-isogeny graph $\mathcal{G}_\ell(\overline{\mathbb{F}_p})$ is a Ramanujan graph.

*Deuring Correspondence:* Suppose that $E$ is a supersingular elliptic curve over $\mathbb{F}_{p^2}$, and its endomorphism ring $\mathrm{End}(E)$ is isomorphic to a maximal order $\mathcal{O}$ of $B_{p,\infty}$.

For a left integral ideal $I$ of $\mathcal{O}$ with $\gcd(p, \mathrm{Nrd}(I)) = 1$, let $E[I]=\{P \in E \mid \alpha(P) = \infty_E$ for any $\alpha \in I\}$, then the isogeny

$$\varphi_I : E \to E_I = E/E[I]$$

has $\ker(\varphi_I) = E[I]$ and $\deg(\varphi_I) = \mathrm{Nrd}(I)$. Conversely, if $\varphi : E \to E'$ is a separable isogeny of degree $n$, then the cardinality of $\ker(\varphi)$ is $n$ and $I_\varphi = \{\alpha \in \mathcal{O} \mid \alpha(P) = \infty_E$ for any $P \in \ker(\varphi)\}$ is a left $\mathcal{O}$-ideal of reduced norm $n$.

The Deuring Correspondence Theorem gives the connection between isogenies and ideals:

There is a one-to-one correspondence between left $\mathcal{O}$-ideals $I$ of reduced norm $n$ and isogenies $\varphi : E \to E'$ of degree $n$, given by $I \mapsto \varphi_I$ and $\varphi \mapsto I_\varphi$. If $\varphi : E \to E'$ and $I$ correspond to each other, then $\mathrm{End}(E')$ is isomorphic to the right order of $I$ in $B_{p,\infty}$. Particularly, $\varphi \in \mathrm{End}(E)$ if and only if $I = \mathcal{O}\varphi$ is a principal ideal. Furthermore, suppose that $\varphi_1 : E \to E_1$ and $\varphi_2 : E \to E_2$ are two isogenies corresponding to the left ideals $I_1, I_2 \subseteq \mathcal{O}$, respectively. Then $E_1$ and $E_2$ are in the same isomorphism class if and only if $I_1$ and $I_2$ are equivalent.

Here we illustrate the endomorphism ring of $E_0 : y^3 = x^3 + x$, which is the starting curve of the SQIsign implementation.

*Example of Endomorphism Ring:* Let $p \equiv 3 \pmod 4$ and $E_0 : y^2 = x^3 + x$ be a supersingular elliptic curve with $j$-invariant 1728. The endomorphism ring of $E_0$ is isomorphic to the maximal order $\mathcal{O}_0 = \mathbb{Z} + \mathbb{Z}i + \mathbb{Z}\frac{i+j}{2} + \mathbb{Z}\frac{1+k}{2}$ where $i^2 = -1$, $j^2 = -p$ and $ij = -ji = k$. Indeed, the Frobenius map $\pi : (x, y) \to (x^p, y^p)$ corresponds to $j$, while the distortion map $\omega : (x, y) \to (-x, iy)$ corresponds to $i$.

## B. SQIsign

SQIsign (Short Quaternion and Isogeny Signature) was first introduced by De Feo et al. [12] in 2020 and it is known as a compact post-quantum signature. This signature is based on an identification protocol with Fiat-Shamir transform [18]. The main procedures of the identification protocol are as follows:

- **Setup**: Generate a prime $p \equiv 3 \pmod 4$ of $2\lambda$ bits, where $\lambda$ is the security parameter. Define a supersingular elliptic curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_p$ with $j(E) = 1728$, and $\mathrm{End}(E_0) \cong \mathcal{O}_0$. Choose an odd smooth number $D_c$ of $\lambda$ bits such that $D_c \mid p^2 - 1$. Besides, let $D = 2^e$ where $e$ is larger than the diameter of $\mathcal{G}_2(\overline{\mathbb{F}_p})$.[1]
- **Key Generation**: Choose a prime $N_\tau \sim p^{\frac{1}{4}}$ and randomly select a $N_\tau$-isogeny $\tau : E_0 \to E_A$. The secret key is the isogeny $\tau$ (note that the degree of $\tau$ is also private), and the public key is the image curve $E_A$.
- **Commitment**: The prover generates a random isogeny $\psi_1 : E_0 \to E_1$, and sends $E_1$ to the verifier.
- **Challenge**: The verifier sends a cyclic isogeny $\psi_2 : E_1 \to E_2$ of degree $D_c$ to the prover.
- **Response**: From the knowledge of the isogeny $\psi_2 \circ \psi_1 \circ \hat{\tau} : E_A \to E_2$, the prover uses the SigningKLPT algorithm [12, Algorithm 5] to construct an isogeny $\sigma : E_A \to E_2$ of degree $D$ such that $\hat{\psi}_2 \circ \sigma$ is cyclic. The prover then transmits $\sigma$ to the verifier.
- **Verification**: The verifier accepts if the isogeny $\sigma : E_A \to E_2$ has degree $D$ and $\hat{\psi}_2 \circ \sigma$ is cyclic. It rejects otherwise.

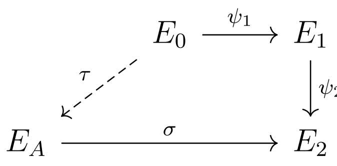
$$\begin{tikzcd} & E_0 \arrow[r, "\psi_1"] \arrow[dl, dashed, "\tau"'] & E_1 \arrow[d, "\psi_2"] \\ E_A \arrow[rr, "\sigma"] & & E_2 \end{tikzcd}$$

Fig. 1. Sketch of the identification protocol.

Since the reduced norm of $I_\tau$ is a large prime, it is expensive to compute the corresponding isogeny $\tau$ directly by Vélu's formula. To compute the coefficient of $E_A$ efficiently, one can utilize the KLPT algorithm to translate $I_\tau$ to another equivalent ideal $I_2$ of reduced norm $2^{e_\tau}$, which corresponds to an isogeny from $E_0$ to $E_A$ of degree $2^{e_\tau}$. An alternative approach is to generate $I_\tau$ and $I_2$ simultaneously by finding $\gamma' \in \mathbb{Z} + \mathbb{Z}i + \mathbb{Z}j + \mathbb{Z}k$ with reduced norm $N_\tau 2^{e_\tau}$, then set $I_\tau = \langle \gamma', N_\tau \rangle$ and $I_2 = \langle \overline{\gamma'}, 2^{e_\tau} \rangle$. Compared to the former one, the latter method is more efficient, but it makes the distribution of secret keys unclear [12, Appendix D].

The response phase is the most complicated procedure. To avoid revealing the secret, one should first construct a new ideal $I_\sigma$ using the SigningKLPT algorithm [12] from the knowledge of $\psi_2 \circ \psi_1 \circ \hat{\tau}$, and then translate $I_\sigma$ to the corresponding isogeny $\sigma$ of degree $D$. Compared with the generation of $I_\sigma$, the translation from $I_\sigma$ to the corresponding isogeny is much more expensive. In the following, we review the ideal-to-isogeny translation in the current SQIsign implementation.

## C. Ideal-to-Isogeny Translation

The efficiency bottleneck of SQIsign is the translation from the ideal $I_\sigma$ to the corresponding isogeny $\sigma$. In the current implementation of SQIsign, the signer needs to decompose the isogeny $\sigma$ of degree $2^e$ into a sequence of isogenies $\varphi_i$, $i = 1, 2, \cdots, n$ of degree $2^a$ such that

$$\sigma = \varphi_n \circ \cdots \circ \varphi_2 \circ \varphi_1,$$

[1]In practice, set $e \approx 3.75 \log p$ as the SigningKLPT algorithm [12, Algorithm 5] outputs an ideal of reduced norm $\approx p^{3.75}$.

where $a$ is the integer such that $2^a \| p+1$.

Let $J$ be an $(\mathcal{O}_0, \mathcal{O})$-ideal of reduced norm $2^\bullet$. The core of the ideal-to-isogeny translation is, given an ideal $K = \overline{J} + 2^a\mathcal{O}$ and the corresponding isogeny $\varphi_K$ of degree $2^a$ with kernel $\langle P\rangle$, one can find the corresponding isogeny of $I = \langle \alpha, 2^a\rangle$ by computing the kernel $\langle [C]P + [D]\theta(P)\rangle$, where $\theta \in \mathcal{O}\backslash(\mathbb{Z} + K + 2\mathcal{O})$ has smooth reduced norm and satisfies that $\alpha(C + D\theta) \in K$ [13, Lemma 8]. To achieve this, the following two algorithms are required:

**SpecialEichlerNorm**$_T(\mathcal{O}, K)$: Given a maximal order $\mathcal{O}$ and a left $\mathcal{O}$-ideal $K$ of reduced norm $\ell$, outputs $\beta \in \mathcal{O}\backslash(\mathbb{Z} + K)$ of reduced norm dividing $T^2$, where $T$ is a parameter such that $\gcd(T, \ell) = 1$ and $T | p^2 - 1$.

**IdealToIsogeny**$(I)$: Given an ideal $I \subseteq \mathcal{O}_0$ of reduced norm dividing $T$, outputs the corresponding isogeny $\varphi_I$.

**Algorithm 1** IdealToIsogenyEichler$_{2^a}(\mathcal{O}, I, J, \varphi_J, P)$ [13, Algorithm 4]

```
Require: A left $\mathcal{O}$-ideal $I$ of reduced norm $2^a$, an $(\mathcal{O}_0, \mathcal{O})$-ideal $J$ of reduced norm $2^\bullet$ and $\varphi_J\colon E_0 \to E$ the corresponding isogeny, a generator $P$ of $E[2^a] \cap \ker(\hat{\varphi}_J)$.
Ensure: $\varphi_I$ of degree $2^a$.
1: $K \leftarrow \overline{J} + 2^a\mathcal{O}$;
2: $\theta \leftarrow$ SpecialEichlerNorm$_T(\mathcal{O}, K + 2\mathcal{O})$;
3: Select $\alpha \in I$ such that $I = \mathcal{O}\langle \alpha, 2^a\rangle$;
4: Compute $C, D$ such that $\alpha(C + D\theta) \in K$ and $\gcd(C, D, 2) = 1$;
5: Take any $n_1 \mid T$ and $n_2 \mid T$ such that $n_1 n_2 = \mathrm{Nrd}(\theta)$. Compute $H_1 = \mathcal{O}\langle \theta, n_1\rangle$ and $H_2 = \mathcal{O}\langle \overline{\theta}, n_2\rangle$;
6: $L_i \leftarrow [J]^* H_i$, $\phi_i \leftarrow [\varphi_J]_*$IdealToIsogeny$(L_i)$ for $i \in \{1, 2\}$;
7: Compute $Q \leftarrow \hat{\phi}_2 \circ \phi_1(P)$;
8: Compute $\varphi_I$ of kernel $\langle [C]P + [D]Q\rangle$;
9: return $\varphi_I$.
```

Algorithm 1 describes how to translate each $\varphi_i$. In the first execution to compute $\varphi_1$, the signer takes $\mathcal{O} = \mathcal{O}_A \cong \mathrm{End}(E_A)$, $I = I_\sigma + 2^a\mathcal{O}_A$, $J = I_2$, $\varphi_J = \varphi_{I_2}$ (as defined in Section II-B, $I_2$ is an $(\mathcal{O}_0, \mathcal{O}_A)$-ideal of reduced norm $2^{e_\tau}$) and the generator $P$ of $E_A[2^a] \cap \ker(\hat{\varphi}_J)$ as the input. We present Figure 2 to illustrate the procedure of the ideal-to-isogeny translation.

The most expensive step of Algorithm 1 is to compute $Q = \theta(P) = \hat{\phi}_2 \circ \phi_1(P)$. To reduce the computational cost, one can utilize Algorithm 2 to obtain the $x$-coordinate of $[C]P + [D]Q$ from the knowledge of $\mathrm{Trd}(\theta)$. Compared to directly computing $Q = \theta(P)$, one isogeny construction can be saved.

### *D. Reduced Tate Pairing*

Let $E$ be an elliptic curve over $\overline{\mathbb{F}_p}$, the reduced Tate pairing is a map :

$$e_n : E(\mathbb{F}_q)[n] \times E(\mathbb{F}_q)/nE(\mathbb{F}_q) \to \mu_n,$$

where $q$ is the power of $p$ and $\mu_n$ is the $n$-roots of unity in $\overline{\mathbb{F}_p}$. There are some properties of the reduced Tate pairing [26, Theorems IX.7, IX.9]:

**Algorithm 2** EndomorphismEvaluation$(\phi_1, \phi_2, C, D, t, P)$ [13, Algorithm 6]

```
Require: Two isogenies $\phi_1$, $\phi_2$ from $E$ to $E'$, scalars $C$ and $D$, the reduced trace $\mathrm{Trd}(\theta)$ where $\theta = \hat{\phi}_2 \circ \phi_1$, and a point $P \in E[2^a]$.
Ensure: The $x$-coordinate of $[C]P + [D]\theta(P)$.
1: Compute $Q$ such that $\langle P, Q\rangle = E[2^a]$ and compute $P+Q$;
2: Compute $x_{\phi_1(P)}$, $x_{\phi_1(Q)}$, $x_{\phi_2(P)}$, $x_{\phi_2(Q)}$, $x_{\phi_2(P+Q)}$;
3: Compute $s_1, s_2$ such that $x_{\phi_1(P)}$ is equal to the $x$-coordinate of $[s_1]\phi_2(P) + [s_2]\phi_2(Q)$;
4: Compute $s_3, s_4$ such that $x_{\phi_1(Q)}$ is equal to the $x$-coordinate of $[s_3]\phi_2(P) + [s_4]\phi_2(Q)$;
5: Change the signs of $(s_1, s_2)$, $(s_3, s_4)$ until $(s_1 + s_4)\deg(\phi_2) \equiv \mathrm{Trd}(\theta) \bmod 2^a$;
6: Compute the $x$-coordinate of $[C + s_1 D\deg(\phi_2)]P + [s_2 D\deg(\phi_2)]Q$ and set it as $x_R$;
7: return $x_R$.
```

1) Assume $P_1, P_2 \in E(\mathbb{F}_q)[n]$, $P_3, P_4 \in E(\mathbb{F}_q)/nE(\mathbb{F}_q)$. Then

$$e_n(P_1 + P_2, P_3) = e_n(P_1, P_3)e_n(P_2, P_3),$$
$$e_n(P_1, P_3 + P_4) = e_n(P_1, P_3)e_n(P_1, P_4).$$

2) Let $P \in E(\mathbb{F}_q)[n]$. If $e_n(P, Q) = 1$ for any $Q \in E(\mathbb{F}_q)/nE(\mathbb{F}_q)$, then $P = \infty_E$.
3) Let $Q \in E(\mathbb{F}_q)/nE(\mathbb{F}_q)$. If $e_n(P, Q) = 1$ for any $P \in E(\mathbb{F}_q)[n]$, then $Q \in nE(\mathbb{F}_q)$.
4) Let $P \in E(\mathbb{F}_q)[N]$ and $Q \in E(\mathbb{F}_q)$, where $N = nn'$. Then

$$e_n([n']P, Q) = e_N(P, Q)^{n'}.$$

## III. Efficient Elliptic Curve Discrete Logarithm Computations

In this section, we focus on how to solve the two elliptic curve discrete logarithms in Algorithm 2 and propose a more efficient approach to obtain $s_1$ and $s_2$. To be precise,

$$\begin{aligned} \phi_1(P) &= [s_1]\phi_2(P) + [s_2]\phi_2(Q), \\ \phi_1(Q) &= [s_3]\phi_2(P) + [s_4]\phi_2(Q). \end{aligned} \tag{1}$$

where $\phi_1$, $\phi_2$ are two isogenies of odd degree. For simplicity, we denote $P_i = \phi_i(P)$ and $Q_i = \phi_i(Q)$, $i = 1, 2$.

The authors in [13] used the Pohlig-Hellman algorithm [27] with a balanced strategy to reduce the above two elliptic curve discrete logarithms in the group $E_A[2^a]$ into multiple elliptic curve discrete logarithms in the group $E_A[2]$. For efficiency, they suggested using the $x$-only arithmetic to recover the absolute values of $s_1$, $s_2$, $s_3$ and $s_4$ by computing two elliptic curve discrete logarithms in Equation (1), and then determine the signs of them with the help of $\mathrm{Trd}(\theta)$. However, it still incurs large computational cost. This method has to compute the $x$-coordinates of $P_i + Q_j$ and $P_1 + P_2 + Q_i$ $(i, j = 1, 2)$ in advance and store all of them into a stack. During the computation, all the elements in the stack need to be updated frequently in order to entirely utilize the $x$-only arithmetic.

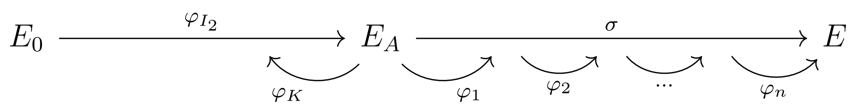

Fig. 2. Sketch of the ideal-to-isogeny translation. In the first execution to compute $\varphi_1$, we set $J = I_2$, $K = \overline{J} + 2^a\mathcal{O}_A$.

On the other hand, as we can see in Algorithm 2, the goal of computing the absolute values of $s_3$ and $s_4$ in the second elliptic curve discrete logarithm is merely to confirm the signs of $s_1$ and $s_2$. It is natural to ask whether one could compute only one elliptic curve discrete logarithm to obtain the exact values of $s_1$ and $s_2$.

In the following, we propose a more efficient method to obtain the exact values of $s_1$ and $s_2$. Firstly, we demonstrate how to avoid the second elliptic curve discrete logarithm computation in Equation (1) with the knowledge of $\theta$. Subsequently, inspired by previous works, we take full advantage of pairing computations to translate the first elliptic curve discrete logarithm into two discrete logarithms in the finite field $\mathbb{F}_{p^2}$. Finally, we show how to compute the two discrete logarithms in $\mathbb{F}_{p^2}$ efficiently.

## A. Saving One Elliptic Curve Discrete Logarithm Computation

Thanks to [13, Lemma 8], the ideal-to-isogeny translation applies Algorithm 1 to choose an endomorphism $\theta \in \mathcal{O}\backslash(\mathbb{Z} + K)$ for computing the kernel of $\varphi_I$. To optimize the implementation, we further exploit the specific properties of $\theta$. Now we propose Lemma 1, a key observation that eliminates the second elliptic curve discrete logarithm computation in Equation (1):

*Lemma 1:* Assume that $\varphi_J$ is a cyclic $2^{\bullet}$-isogeny from $E_0$ to $E$, and $J$ is the corresponding right $\mathcal{O}$-ideal. Suppose that $K = \overline{J} + 2\mathcal{O}$. If the endomorphism $\theta \in \mathcal{O}\backslash(\mathbb{Z} + K)$ and $P$ is a point of order $2^a$ such that $\langle P \rangle = E[2^a] \cap \ker(\hat{\varphi}_J)$, then $\theta([2^{a-1}]P) \neq [2^{a-1}]P$.

*Proof:* Clearly, the ideal corresponding to the isogeny $\hat{\varphi}_J$ is $\overline{J}$. Hence, for any $\delta \in K = \overline{J} + 2\mathcal{O}$, we have

$$\delta([2^{a-1}]P) = \infty_E.$$

Suppose that $\theta([2^{a-1}]P) = [2^{a-1}]P$. From $\theta([2^{a-1}]P) - [2^{a-1}]P = \infty_E$, we can deduce $\theta - 1 \in K$ from the Deuring Correspondence Theorem. It implies that $\theta \in \mathbb{Z} + K$. This contradicts the fact that $\theta \in \mathcal{O}\backslash(\mathbb{Z} + K)$. □

Lemma 1 implies that the endomorphism $\theta$ always maps $[2^{a-1}]P$ to a point which is not $[2^{a-1}]P$. Since the reduced norm of $\theta$ divides $T^2$ and $T$ is odd, $\theta([2^{a-1}]P)$ is not the point at infinity. This implies that the endomorphism $\theta$ maps $[2^{a-1}]P$ to another point of order 2.

In the following, we show that $s_2$ in Equation (1) is always odd. It confirms that $s_2^{-1} \bmod 2^a$ exists, which can be employed to accelerate the performance.

*Proposition 1:* At Step 3 of Algorithm 2, we have $s_2 \equiv 1 \bmod 2$.

*Proof:* From $P_1 = [s_1]P_2 + [s_2]Q_2$, we have

$$[2^{a-1}]P_1 = [s_1]([2^{a-1}]P_2) + [s_2]([2^{a-1}]Q_2).$$

Suppose for contradiction that $s_2$ is even. Since the order of $[2^{a-1}]Q_2$ is 2, $[s_2]([2^{a-1}]Q_2)$ is the point at infinity. Therefore,

$$[2^{a-1}]P_1 = [s_1]([2^{a-1}]P_2).$$

Applying $\hat{\phi}_2$ to the above equation yields:

$$\theta([2^{a-1}]P) = [s_1 \deg(\phi_2)]([2^{a-1}]P).$$

From the deduction above, we know that the point $\theta([2^{a-1}]P)$ is of order 2. It implies that $\theta([2^{a-1}]P) = [2^{a-1}]P$, which is a contradiction with Lemma 1. Therefore, we have $s_2 \equiv 1 \bmod 2$. □

Now we demonstrate how to avoid the second elliptic curve discrete logarithm in Equation (1). From the knowledge of $\mathrm{Trd}(\theta)$ and $\mathrm{Nrd}(\theta)$, one can directly compute $s_3$ and $s_4$ with respect to $s_1$ and $s_2$. Note that

$$\theta \begin{pmatrix} P \\ Q \end{pmatrix} = \begin{pmatrix} s_1 \deg(\phi_2) & s_2 \deg(\phi_2) \\ s_3 \deg(\phi_2) & s_4 \deg(\phi_2) \end{pmatrix} \begin{pmatrix} P \\ Q \end{pmatrix},$$

and $s_2$, $\deg(\phi_2)$ are invertible in $\mathbb{Z}/2^a\mathbb{Z}$. Therefore, after recovering the absolute values of $s_1$ and $s_2$ in the first elliptic curve discrete logarithm computation, one can suppose

$$\begin{aligned} s_4 &= \frac{\mathrm{Trd}(\theta)}{\deg(\phi_2)} - s_1 \bmod 2^a, \\ s_3 &= \frac{s_1 s_4 \deg(\phi_2)^2 - \mathrm{Nrd}(\theta)}{s_2 \deg(\phi_2)^2} \bmod 2^a. \end{aligned} \tag{2}$$

Then, compute the $x$-coordinate of $[s_3]P_2 + [s_4]Q_2$. If the $x$-coordinate of $[s_3]P_2 + [s_4]Q_2$ is equal to that of $Q_1$, then the signs of $s_1$ and $s_2$ are correct. Otherwise, we need to change the signs of them. The main procedure is summarized in Algorithm 3:

At the beginning of this section, we reviewed the current implementation of computing discrete logarithms on elliptic curves in SQIsign. Even though the authors in [13] utilized the $x$-only arithmetic, the procedure remains expensive. A question raised here is how to compute the first elliptic curve discrete logarithm in Equation (1) more efficiently.

Our optimization is reminiscent of public-key compression in SIDH [28]. That is, applying pairings (note that the pairing we use should satisfy $e_{2^a}(R, R) = 1$ for any $R \in E(\mathbb{F}_{p^2})[2^a]$) to translate the elliptic curve discrete logarithm into two discrete logarithms in the cyclic group $\mu_{2^a} = \{h \in \mathbb{F}_{p^2} | h^{2^a} = 1\}$:

$$\begin{aligned} h_0 &= e_{2^a}(P_2, Q_2), \\ h_1 &= e_{2^a}(P_2, P_1) = e_{2^a}(P_2, [s_1]P_2 + [s_2]Q_2) = h_0^{s_2}, \\ h_2 &= e_{2^a}(Q_2, P_1) = e_{2^a}(Q_2, [s_1]P_2 + [s_2]Q_2) = h_0^{-s_1}. \end{aligned} \tag{3}$$

In Sections III-B and III-C, we show how to efficiently compute the pairings in Equation (3) and the two discrete logarithms in $\mu_{2^a}$ to recover $s_1$ and $s_2$, respectively.

**Algorithm 3** EndomorphismEvaluation($\varphi_1$, $\varphi_2$, $C$, $D$, $t$, $n$, $P$)

```
Require: Two isogenies $\phi_1$, $\phi_2$ from $E$ to $E'$, scalars $C$ and $D$, the reduced trace $\mathrm{Trd}(\theta)$ where $\theta = \hat{\phi_2} \circ \phi_1$, the reduced norm $\mathrm{Nrd}(\theta)$ and a point $P \in E[2^a]$.
Ensure: The $x$-coordinate of $[C]P + [D]\theta(P)$.
1: Compute $Q$ such that $\langle P, Q\rangle = E[2^a]$ and compute $P+Q$;
2: Compute $x_{\phi_1(P)}$, $x_{\phi_1(Q)}$, $x_{\phi_2(P)}$, $x_{\phi_2(Q)}$, $x_{\phi_2(P+Q)}$;
3: Compute $s_1, s_2$ such that $x_{\phi_1(P)}$ is equal to the $x$-coordinate of $[s_1]\phi_2(P) + [s_2]\phi_2(Q)$;
4: Let $s_4 = \mathrm{Trd}(\theta)/\deg(\phi_2) - s_1 \bmod 2^a$ and $s_3 = (s_1 s_4 \deg(\phi_2)^2 - \mathrm{Nrd}(\theta))/\left(s_2 \deg(\phi_2)^2\right) \bmod 2^a$;
5: Compute the $x$-coordinate of $[s_3]\phi_2(P) + [s_4]\phi_2(Q)$ and set it as $x_t$;
6: if $x_t \neq x_{\phi_1(Q)}$ then
7:     $s_1 \leftarrow -s_1$, $s_2 \leftarrow -s_2$;
8: end if
9: Compute the $x$-coordinate of $[C + s_1 D \deg(\phi_2)]P + [s_2 D \deg(\phi_2)]Q$ and set it as $x_R$;
10: return $x_R$.
```

## *B. Pairing Computations*

In this subsection, we show the feasibility of adapting the reduced Tate pairing in Equation (3) and explore how to compute $h_0$, $h_1$ and $h_2$ efficiently. Furthermore, we analyze the situation when using the Weil pairing. For simplicity, we adopt the notation $e_{T,n}(\cdot,\cdot)$ and $e_{W,n}(\cdot,\cdot)$ for the reduced Tate pairing and the Weil pairing, respectively.

Since the embedding degree is equal to 1, the property that $e_{T,2^a}(R,R) = 1$ does not necessarily hold for any $R \in E(\mathbb{F}_{p^2})[2^a]$. Hence, a natural question to ask is whether the deduction in Equation (3) is still correct when applying the reduced Tate pairing. Indeed, the fact $e_{T,2^a}(R,R) = 1$ has been applied to public-key compression in SIDH [29]. It seems that the correctness is well known to the experts, but we did not find a relevant proof in the literature. Therefore, we propose Theorem 1 for illustrating the special feature of the reduced Tate pairing in our scenario.

*Theorem 1:* Suppose that $E$ is a supersingular elliptic curve over $\mathbb{F}_{p^2}$ with cardinality $\#E(\mathbb{F}_{p^2}) = (p+1)^2$. Let $E[n] \subseteq E(\mathbb{F}_{p^2})$. Then $e_{T,n}(R,R) = 1$ for any $R \in E(\mathbb{F}_{p^2})[n]$.

*Proof:* From [30, Theorem 3.17], we have

$$e_{T,n}(R,R) = e_{W,n}(R, R' - \pi(R')),$$

where $R' \in E(\overline{\mathbb{F}_p})$ such that $[n]R' = R$ and $\pi$ is the $p^2$-th power Frobenius map. Since $\#E(\mathbb{F}_{p^2}) = (p+1)^2$, we have $\pi = [-p]$. Therefore,

$$R' - \pi R' = [p+1]R' = \left[\frac{p+1}{n}\right] R.$$

By $E[n] \subseteq E(\mathbb{F}_{p^2})$, it follow that $n|p+1$ [30, Corollary 3.11]. From the bilinearity and the alternating property of the Weil pairing, we have

$$e_{T,n}(R,R) = e_{W,n}(R, R' - \pi(R')) = e_{W,n}(R,R)^{\frac{p+1}{n}} = 1,$$

which concludes the proof. □ □

It remains to explore how to efficiently compute the reduced Tate pairings. In the SIDH/SIKE implementation [31], Naehrig and Renes [32] used the dual isogeny to pull back the pairing computations from the image curve to the starting curve. However, this technique does not work here because

$$\begin{aligned} h_0 &= e_{2^a}(\hat{\varphi}_J(P), \hat{\varphi}_J(Q)) \\ &= e_{2^a}(P,Q)^{\deg(\varphi_J)} = e_{2^a}(P,Q)^{2^{a+\bullet}} = 1. \end{aligned}$$

Similarly, we have $h_1 = h_2 = 1$. Therefore, we have to compute the three pairings in Equation (3) on the image curve $E$, as done in [28] and [29].

The reduced Tate pairing computations involve two procedures: Miller function construction and the final exponentiation. Compared to the latter, the former consumes more computational resources because of the low embedding degree. In the SIDH/SIKE implementation, the state-of-the-art improves the Miller loop computation by the following formula with precomputation [33]:

$$\begin{aligned} &\mathrm{div}(f_{4^{n+1},R}) \\ &\quad = \mathrm{div}\left(\frac{\left[f_{4^n,R}^2 \cdot \left(\lambda_1(x - x_{[2\cdot4^n]R}) - (y + y_{[2\cdot4^n]R})\right)\right]^2}{\lambda_2(x - x_{[2\cdot4^n]R}) - (y + y_{[2\cdot4^n]R})}\right), \end{aligned} \tag{4}$$

where the function $f_{N,R}$ is rational with divisor $\mathrm{div}(f_{N,R}) = N(R) - ([N]R) - (N-1)(\infty_E)$, the values $\lambda_1$ and $\lambda_2$ are the slopes of the lines passing through $[4^n]R$ and $[-2\cdot4^n]R$ twice, respectively. In our case, we are not able to apply the precomputation technique since the two arguments are unknown. However, one can still use Equation (4) instead of adapting the usual doubling step:

$$\mathrm{div}(f_{2^{n+1},R}) = \mathrm{div}\left(f_{2^n,R}^2 \frac{\lambda_1(x - x_{[2^n]R}) - (y - y_{[2^n]R})}{x - x_{[2^{n+1}]R}}\right), \tag{5}$$

where $\lambda_1$ is the slope of the line passing through $[2^n]R$ twice.

For efficiency, we use modified Jacobian coordinates to compute the pairings. The doubling operation requires only $3\boldsymbol{M} + 5\boldsymbol{S}$ [34], where $\boldsymbol{S}, \boldsymbol{M}$ are the cost of an $\mathbb{F}_{p^2}$ field squaring and multiplication, respectively. Another advantage of using modified Jacobian coordinates is that during the computation of doubling/quadrupling of $R$ one can also obtain $\lambda_1$ and $\lambda_2$ easily.

According to our estimate, each quadrupling Miller loop using Equation (4) with modified Jacobian coordinates costs $17\boldsymbol{M} + 13\boldsymbol{S}$. It saves $3\boldsymbol{M} + 1\boldsymbol{S}$ compared to computing two doubling Miller loops using Equation (5) with modified Jacobian coordinates.

The final exponentiation is an exponentiation to the power $\frac{p^2-1}{2^a} = (p-1)\cdot\frac{p+1}{2^a}$. Raising to the power $p-1$ is straightforward, requiring only one application of the Frobenius map and one inversion in $\mathbb{F}_{p^2}$. Conversely, the exponentiation to the power $\frac{p+1}{2^a}$ is a hard part. One can utilize the efficient formulas in the cyclotomic subgroup $\mu_{p+1} = \{h^{p+1} = 1 | h \in \mathbb{F}_{p^2}\}$ [29, Section 5.1]. Another effective method, which is proposed by Scott and Barreto [35], is to raise the power with the help of Lucas sequences [36, Section 3.6.3]. In the implementation, we employ the latter one for the hard part since it performs better.

In fact, we can further optimize the computation from the relations of $h_0$ and $h_1$. Adapting the reduced Tate pairings in Equation (3),

$$\begin{aligned} h_0 &= e_{T,2^a}(P_2, Q_2) = f_{2^a,P_2}(Q_2)^{\frac{p^2-1}{2^a}}, \\ h_1 &= e_{T,2^a}(P_2, P_1) = f_{2^a,P_2}(P_1)^{\frac{p^2-1}{2^a}}, \\ h_2 &= e_{T,2^a}(Q_2, P_1) = f_{2^a,Q_2}(P_1)^{\frac{p^2-1}{2^a}}. \end{aligned} \tag{6}$$

Note that the first two pairing computations share the same first argument. Therefore, we can merge the computations of $h_0$ and $h_1$ to eliminate the redundant Miller function construction.

*Remark 1:* The techniques proposed above cannot be directly applied to the case when the order of the pairing is $2^{a'}$ with $a' < a$. This is because, given a class in $E(\mathbb{F}_{p^2})/2^{a'}E(\mathbb{F}_{p^2})$, we may not find an element in $E(\mathbb{F}_{p^2})[2^{a'}]$ to be the representative of the class. Assume that $\langle P, Q\rangle = E(\mathbb{F}_{p^2})[2^{a'}]$. Since $\langle [2^{a-a'}]P, [2^{a-a'}]Q\rangle \in [2^{a'}]E(\mathbb{F}_{p^2})$ and the second argument of the reduced Tate pairing is a representative of the class in $E(\mathbb{F}_{p^2})/2^{a'}E(\mathbb{F}_{p^2})$, the order of the reduced Tate pairing $e_{2^{a'}}(P, Q)$ is $2^{2a'-a}$ in $\mathbb{F}_{p^2}$. For instance, set $a' = a - 1$. In this situation, all the points in $E(\mathbb{F}_{p^2})[2]$ represent the same class $[\infty_E]$ in $E(\mathbb{F}_{p^2})/2E(\mathbb{F}_{p^2})$. If the second argument is a point of order $2^{a-1}$, then $e_{2^{a-1}}(P, Q)$ is of order $2^{a-2}$ in $\mathbb{F}_{p^2}$. Especially, if we consider the pairing of order $2^{a'}$ satisfying $2a' < a$, the value $e_{2^{a'}}(P, Q)$ is always equal to 1. Fortunately, we always handle the case $a' = a$ except for the last step of the ideal-to-isogeny translation.

An alternative approach to compute pairings in Equation (3) is to utilize the Weil pairing [37, Proposition 8]:

$$\begin{aligned} e_{W,2^a}(P_2, P_1) &= \frac{f_{2^a,P_2}(P_1)}{f_{2^a,P_1}(P_2)}, \\ e_{W,2^a}(P_2, Q_2) &= \frac{f_{2^a,P_2}(Q_2)}{f_{2^a,Q_2}(P_2)}, \\ e_{W,2^a}(Q_2, P_1) &= \frac{f_{2^a,Q_2}(P_1)}{f_{2^a,P_1}(Q_2)}. \end{aligned} \tag{7}$$

Clearly, we need to construct three Miller functions. For the reduced Tate pairing computation, Miller function construction is more expensive than the final exponentiation. Additionally, only two Miller function constructions are needed in Equation (6), while there are three Miller functions to be constructed in Equation (7). Consequently, the Weil pairing computation is still not as efficient as the reduced Tate pairing computation. But in parallel implementation, the Weil pairing computation becomes more competitive since it does not need the final exponentiation and all Miller function evaluations could be executed simultaneously. Another advantage compared to the reduced Tate pairing is that one can apply the Weil pairing to the situation when the order of the pairing is less than $2^a$.

## *C. Discrete Logarithm Computations in $\mu_{2^a}$*

Since the order of $\mu_{2^a}$ is smooth, one can use the Pohlig-Hellman algorithm with an optimal strategy to translate discrete logarithms in $\mu_{2^a}$ into discrete logarithms in $\mu_{2^w}$, where $w$ is a small integer. It remains to compute discrete logarithms in $\mu_{2^w}$ efficiently.

The authors in [38] proposed two methods to accelerate discrete logarithm computations. The first one is to compute a lookup table with respect to the base $h_0$:

$$\begin{aligned} &T_1^{sgn}[r][c] = (h_0)^{(c+1)2^{wr+m}}, \\ &\qquad r = 0, 1, \cdots, \lfloor \frac{a}{w} \rfloor - 1, c = 0, 1, \cdots, 2^{w-1} - 1, \end{aligned} \tag{8}$$

where $m \equiv a \bmod w$. Since $h_0$ is not fixed, we cannot compute the lookup table in advance. As the base power $w$ increases, the lookup table construction becomes more expensive, and it requires more storage at the same time, while the discrete logarithm computations would be more efficient.

The second method proposed in [38] is to compute only the first column and the last row of the lookup table in Equation (8):

$$\begin{aligned} FC &= \left\{T_1^{sgn}[r][0] = (h_0)^{2^{wr+m}}, i = 0, 1, \cdots, \lfloor \frac{a}{w} \rfloor - 1\right\}, \\ LR &= \left\{T_1^{sgn}[\lfloor \frac{a}{w} \rfloor - 1][c] = (h_0)^{(c+1)2^{a-w}}, c = 0, 1, \cdots, 2^{w-1}\right\}. \end{aligned} \tag{9}$$

The discrete logarithm computations with Equation (9) is more expensive compared to that of the former method. However, the construction of Equation (9) is more efficient than the entire lookup table construction. Furthermore, the latter method would be preferred in storage restrained environments.

In the following, we give another effective approach to improve the performance of discrete logarithms in $\mu_{2^a}$.

At first glance, as $h_0$ is not fixed, it seems that we cannot use precomputation in the setup phase to reduce the computational cost. However, since the cyclotomic group $\mu_{2^a}$ in $\mathbb{F}_{p^2}$ is fixed, one can find a primitive element $g$ of $\mu_{2^a}$ in advance. Instead of computing the two discrete logarithms of $h_1$, $h_2$ to the base $h_0$, we compute three discrete logarithms of $h_0$, $h_1$, $h_2$ to the base $g$:

$$h_0 = g^{s_0'}, h_1 = g^{s_1'}, h_2 = g^{s_2'}. \tag{10}$$

Hence, when the storage is available, we can use the precomputation in the setup phase to further speed up the discrete logarithm computations in Equation (10). In this case, the lookup table with respect to $g$ is as follows:

$$\begin{aligned} &T_1^{sgn}[r][c] = g^{(c+1)2^{wr+m}}, \\ &\qquad r = 0, 1, \cdots, \lfloor \frac{a}{w} \rfloor - 1, c = 0, 1, \cdots, 2^{w-1} - 1. \end{aligned} \tag{11}$$

Note that $h_0$ is also a primitive element in $\mu_{2^a}$. Therefore, we can recover the solutions by one inversion and two multiplications in $\mathbb{Z}/2^a\mathbb{Z}$:

$$s_1 = (s_0')^{-1}s_1', s_2 = (s_0')^{-1}s_2'.$$

**Algorithm 4** PH_DLP($h$, $g$, $w$, $T_1^{sgn}$, $Str$)

**Require:** The challenge $h$, a primitive element $g$ in the multiplicative group $\mu_{2^a}$, the base power $w$, the lookup table $T_1^{sgn}$ in Equation (11), the optimal strategy $Str$.

**Ensure:** The array $D$ such that $h = g^{\left(D[\lfloor \frac{a}{w} \rfloor -1]\cdots D[1]D[0]\right)_{2^w}}$.

```
1:  Initialize a Stack $Stack$, which contains tuples of the form $(h_t, e_t, l_t)$, where $h_t \in \mu_{2^a}$, $e_t, l_t \in \mathbb{N}$.
2:  $LR \leftarrow$ the last row of the lookup table $T_1^{sgn}$;
3:  $i \leftarrow 0$, $j \leftarrow 0$, $k \leftarrow 0$, $m \leftarrow 2^a \bmod w$, $h_t \leftarrow h$, $y \leftarrow 1$;
4:  $h_t \leftarrow (h_t)^{2^m}$;
5:  Push the tuple $(h_t, j, k)$ into $Stack$;
6:  while $k \neq \lfloor \frac{e_\ell}{w} \rfloor - 1$ do
7:      while $j + k \neq \lfloor \frac{e_\ell}{w} \rfloor - 1$ do
8:          $j \leftarrow j + Str[i]$;
9:          $h_t \leftarrow (h_t)^{2^{w \cdot Str[i]}}$;
10:         Push the tuple $(h_t, j + k, Str[i])$ into $Stack$;
11:         $i \leftarrow i + 1$;
12:     end while
13:     Pop the top tuple $(h_t, e_t, l_t)$ from $Stack$;
14:     $j \leftarrow j - l_t$, $k \leftarrow k + 1$;
15:     Find $x_t$ such that $h_t = (LR[0])^{x_t}$ with the help of $LR$;
16:     $D[k] \leftarrow x_t$;
17:     for each tuple $(h_t, e_t, l_t)$ in $Stack$ do
18:         if $x_t \neq 0$ then
19:             if $x_t > 0$ then
20:                 $h_t \leftarrow h_t \cdot \overline{T_1^{sgn}[e_t][x_t - 1]}$;
21:             else
22:                 $h_t \leftarrow h_t \cdot T_1^{sgn}[e_t][-x_t - 1]$;
23:             end if
24:         end if
25:         $e_t \leftarrow e_t + 1$;
26:     end for
27: end while
28: Pop the top tuple $(h_t, e_t, l_t)$ from $Stack$;
29: Find $x_t$ such that $h_t = (LR[0])^{x_t}$ with the help of $LR$;
30: $D[k] \leftarrow x_t$;
31: if $m \neq 0$ then
32:     $y_0 \leftarrow g^{D[0]}$;
33:     for $i_2$ from 1 to $\lfloor \frac{e_\ell}{w} \rfloor - 1$ do
34:         if $D[i_2] < 0$ then
35:             $y \leftarrow y \cdot \overline{T_1^{sgn}[i_2 - 1][-D[i_2] - 1]}$;
36:         end if
37:         if $D[i_2] > 0$ then
38:             $y \leftarrow y \cdot T_1^{sgn}[i_2 - 1][D[i_2] - 1]$;
39:         end if
40:     end for
41:     $y \leftarrow y^{2^{w-m}}$;
42:     $y \leftarrow y_0 \cdot y$, $y \leftarrow h \cdot \overline{y}$;
43:     Find $x_t$ such that $y = (LR[0])^{x_t}$ with the help of $LR$;
44:     $D[k + 1] \leftarrow \frac{x_t}{2^{w-m}}$;
45: end if
46: return $D$.
```

Compared with the first two methods, our method offers the advantage of precomputing the entire lookup table with respect to $g$ in the setup phase to enhance the performance. However, it requires one more discrete logarithm computation in $\mu_{2^a}$. We respectively estimate the computational costs by utilizing the three methods presented above when setting the prime $p$ as $p_{1973}$ ($a = 75$). For simplicity, we only consider multiplications and squarings, and assume that their computational costs are approximately equal. As shown in Table I, the previous methods proposed in [38] are more efficient than our new method when the base power is small. As the base power $w$ increases, our new method saves more computational resources. When the storage is limited, one can adapt Method 2 proposed in [38] since it requires the least storage for the lookup table.

TABLE I
COST ESTIMATES (IN $\mathbb{F}_p$ MULTIPLICATIONS) FOR THE DISCRETE LOGARITHM COMPUTATION BY DIFFERENT METHODS

| Method | $w = 1$ | $w = 2$ | $w = 3$ | $w = 4$ | $w = 5$ | $w = 6$ |
|---|---|---|---|---|---|---|
| Method 1 proposed in [37] | 2068 | 1510 | 1219 | 1180 | 1171 | 1438 |
| Method 2 proposed in [37] | 2068 | 1560 | 1338 | 1375 | 1184 | 1389 |
| Our method | 2910 | 1980 | 1421 | 1179 | 855 | 825 |

Based on [38, Algorithm 6], we present Algorithm 4 to solve discrete logarithms. Since the algorithm is non-recursive, it would be more attractive in parallel environments.

## IV. Other Improvements

In this section, we propose other techniques to speed up the signing phase. Some of the improvements also benefit the performance of key generation and verification.

### A. Torsion Point Generation

To accelerate torsion basis generation in compressed SIDH, Costello et al. [29] proposed a method to find out a torsion basis of $E(\mathbb{F}_{p^2})[2^a]$. The main idea is as follows: Firstly, precompute a list $L$ of non-squares in $\mathbb{F}_{p^2}$. Then, randomly select $v_1 \in L$ until $v_1^3 + Av_1^2 + v_1$ is a square. It confirms that $(v_1, \sqrt{v_1^3 + Av_1^2 + v_1})$ is a point on $E(\mathbb{F}_{p^2})$. According to [39, Ch. 1((§4))], the order of the point is divided by $2^a$, thus one can perform scalar multiplication to obtain a point $P$ of order $2^a$. Similarly, one can generate a point $Q$ of order $2^a$ until $\langle P, Q\rangle = E(\mathbb{F}_{p^2})[2^a]$, which can be checked by $[2^{a-1}]P \neq [2^{a-1}]Q$. In this subsection, we will show how to adapt this method to benefit the implementation of SQIsign.

Note that the $2^\bullet$-isogeny $\sigma \circ \varphi_{I_2}$ can be composed by multiple 2-isogenies. In addition, for any 2-isogeny $\phi$ whose kernel is $\langle (x_P, 0)\rangle$ with $x_P \neq 0$, i.e.,

$$\phi : (x, y) \mapsto \left(x \cdot \left(\frac{x_P x - 1}{x - x_P}\right), \sqrt{x_P} \cdot y \cdot \left(\frac{x_P x^2 - 2x_P^2 x + x_P}{(x - x_P)^2}\right)\right),$$

we have

$$\phi((0, 0)) = (0, 0).$$

When the kernel is $\langle (0, 0)\rangle$, the isogeny can be defined by

$$\phi : (x, y) \mapsto \left(\frac{1}{\sqrt{A^2 - 4}} \cdot \frac{x^2 + Ax + 1}{x}, \frac{1}{\sqrt[4]{A^2 - 4}} \cdot y \cdot \frac{x^2 - 1}{x^2}\right),$$

where $A$ is the coefficient of the initial curve $E_A : y^2 = x^3 + Ax^2 + x$ [20, Section 2.3.1]. In this case, the point $(0, 0)$ on the image curve is in the kernel of the dual isogeny.

To summarize, in both cases the dual of 2-isogeny has kernel $\langle(0, 0)\rangle$. Furthermore, we can deduce that a cyclic $2^{\bullet}$-isogeny $\varphi$ computed by the above formulas has the property that $(0, 0) \in \ker(\hat{\varphi})$. This implies that the first step of the dual isogeny has kernel $\langle(0, 0)\rangle$.

Note that in the first execution of the ideal-to-isogeny translation, we have $\varphi_J = \varphi_{I_2}$. Since $\varphi_{I_2}$ is cyclic, we can imply that $(0, 0) \in E[2^a] \cap \ker(\hat{\varphi}_J) = \langle P\rangle$, which means $[2^{a-1}]P = (0, 0)$.

Now we consider the second execution. If $\sigma \circ \varphi_{I_2}$ is cyclic, we can imply that the isogeny $\varphi_J = \varphi_1 \circ \varphi_{I_2}$ is also cyclic. Hence, $E[2^a] \cap \ker(\hat{\varphi}_J) = \ker(\hat{\varphi}_1) = \langle P\rangle$ is a point of order $2^a$, and from the deduction above we have $[2^{a-1}]P = (0, 0)$. However, if $\sigma \circ \varphi_{I_2}$ is not cyclic, then $E[2^a] \cap \ker(\hat{\varphi}_J)$ is not a cyclic group. It follows that $[2^{a-1}]P = \infty$. In this situation, one can set $\langle P\rangle$ to be the kernel of $\hat{\varphi}_1$ instead of $E[2^a] \cap \ker(\hat{\varphi}_J)$. In the meantime, set $K$ to be the ideal corresponding to $\hat{\varphi}_1$ in Algorithm 1. Since the isogeny $\varphi_1$ is cyclic and it has degree $2^a$, we have $[2^{a-1}]P = (0, 0)$.

Similarly, if $E[2^a] \cap \ker(\hat{\varphi}_J) = \ker(\hat{\varphi}_i) = \langle P\rangle$ in the $i$-th ideal-to-isogeny translation with $i > 2$, then from $\varphi_i$ being cyclic we have $[2^{a-1}]P = (0, 0)$. Otherwise we set $\langle P\rangle$ to be the kernel of $\hat{\varphi}_i$ and $K$ as the ideal corresponding to $\hat{\varphi}_i$ in Algorithm 1.

Therefore, in each ideal-to-isogeny translation, we can always set the point $P$ such that $[2^{a-1}]P = (0, 0)$. Now we need another point $Q$ such that $\langle P, Q\rangle = E(\mathbb{F}_{p^2})[2^a]$, i.e., $[2^{a-1}]Q \neq (0, 0)$. Obviously, the above method presented by Costello et al. is exactly suitable for speeding up the generation of $Q$. Further, there is no need to check $[2^{a-1}]Q \neq (0, 0)$ when applying this method since $P$ and $Q$ are always linearly independent, according to Theorem 2.

*Theorem 2:* Assume that $E_A : y^2 = x^3 + Ax^2 + x$ is a supersingular elliptic curve defined on the finite field $\mathbb{F}_{p^2}$, where $2^a \| p + 1$ and $E_A[2^a] \subseteq E_A(\mathbb{F}_{p^2})$. Suppose that $Q = (x_Q, y_Q) \in E_A(\mathbb{F}_{p^2})$ and denote $\mathrm{ord}(Q)$ the order of $Q$. If $2^a \| \mathrm{ord}(Q)$, then $(x_Q)^{\frac{p^2-1}{2}} = -1$ if and only if $\left[\frac{\mathrm{ord}(Q)}{2}\right] Q \neq (0, 0)$.

*Proof:* Suppose that $P$ is a point of order $2^a$ defined on $E_A/\mathbb{F}_{p^2}$. Firstly, we prove that $P$ and $Q$ are linearly independent if and only if $e_{T,2^a}(P, Q)$ is a primitive element of the group $\mu_{2^a}$.

Let $Q \in E_A(\mathbb{F}_{p^2})$ be a rational point such that $P$ and $Q$ are linearly independent. Suppose for contradiction that $e_{T,2^a}(P, Q)$ is not a primitive element of the group $\mu_{2^a}$. Then we have

$$e_{T,2}([2^{a-1}]P, Q) = e_{T,2^a}(P, Q)^{2^{a-1}} = 1,$$

From Theorem 1 we can deduce that

$$e_{T,2}([2^{a-1}]P, P) = e_{T,2^a}(P, P)^{2^{a-1}} = e_{T,2^a}(P, [2^{a-1}]P) = 1. \tag{12}$$

Since $E_A(\mathbb{F}_{p^2})/2E_A(\mathbb{F}_{p^2}) = \{\infty_{E_A} + 2E_A(\mathbb{F}_{p^2}), P + 2E_A(\mathbb{F}_{p^2}), Q + 2E_A(\mathbb{F}_{p^2}), P + Q + 2E_A(\mathbb{F}_{p^2})\}$, we have $e_{T,2}([2^{a-1}]P, R) = 1$ for any $R \in E_A(\mathbb{F}_{p^2})/2E_A(\mathbb{F}_{p^2})$. According to the non-degeneracy property of the reduced Tate pairing, $[2^{a-1}]P = \infty_{E_A}$. This is a contradiction and thus $e_{T,2^a}(P, Q)$ is a primitive element of the group $\mu_{2^a}$.

On the other hand, if $e_{T,2^a}(P, Q)$ is of order $2^a$, then

$$\begin{aligned} e_{T,2^a}(P, Q)^{2^{a-1}} &= e_{T,2^a}(P, [2^{a-1}]Q) \\ &= e_{T,2^a}\left(P, \left[\frac{\mathrm{ord}(Q)}{2}\right] Q\right) = -1. \end{aligned}$$

It follows from Equation (12) that $\left[\frac{\mathrm{ord}(Q)}{2}\right] Q \neq [2^{a-1}]P$. Hence, we can deduce that $P$ and $Q$ are linearly independent.

Now assume that $[2^{a-1}]P = (0, 0)$. If $[\frac{\mathrm{ord}(Q)}{2}]Q \neq (0, 0)$, then $Q$ and $P$ are linearly independent. Therefore, $e_{T,2^a}(P, Q)$ is a primitive element of $\mu_{2^a}$, i.e.,

$$e_{T,2^a}(P, Q)^{2^{a-1}} = e_{T,2}((0, 0), Q) = (x_Q)^{\frac{p^2-1}{2}} = -1. \tag{13}$$

Conversely, if $(x_Q)^{\frac{p^2-1}{2}} = -1$, from Equation (13) we can imply that $P$ and $Q$ are linearly independent. It ensures that $[\frac{\mathrm{ord}(Q)}{2}]Q \neq (0, 0)$. This completes the proof. □ □

With Theorem 2, we can efficiently generate the point $Q$ in Algorithm 5. It should be noted that this improvement benefits all the procedures of SQIsign, especially the verifying phase.

**Algorithm 5** DeterministicSecondPoint($A$)

```
Require: The coefficient $A$ of the Montgomery curve $E_A : y^2 = x^3 + Ax^2 + x$.
Ensure: A point $Q$ defined on $E_A$ of order $2^a$ such that $[2^{a-1}]Q \neq (0, 0)$.
1: Select a non-square element $x_Q \in \mathbb{F}_{p^2}$ such that $x_Q^3 + Ax_Q^2 + x_Q$ is a square;
2: $Q \leftarrow (x_Q, \sqrt{x_Q^3 + Ax_Q^2 + x_Q})$;
3: $Q \leftarrow [\frac{p+1}{2^a}]Q$;
4: return $Q$.
```

## B. Image Curve Recovery With Three Points in Isogeny Computations

In each ideal-to-isogeny translation, we need to construct the large degree isogeny $\phi_2$ and evaluate it at $P$, $Q$ and $P + Q$. Invoking efficiency reasons, the computation of $\phi_2$ is composed by multiple odd degree isogeny computations. At each odd degree isogeny computation, not only we need to evaluate it at the three points, but the image curve should also be obtained. Fortunately, one can use Equation (14) to recover the image curve coefficient $A$ [40, Remark 4]:

$$A = \frac{(1 - x_{P'}x_{Q'} - x_{P'}x_{Q'-P'} - x_{Q'}x_{Q'-P'})^2}{4x_{P'}x_{Q'}x_{Q'-P'}} - x_{P'} - x_{P'} - x_{Q'-P'}, \tag{14}$$

where $P'$ and $Q'$ are two points defined on the image curve. For large prime degree isogeny computations, applying Equation (14) to obtain the image curve coefficient is much more efficient than computing it with Vélu's formula [23],

**Algorithm 6** FirstIdealToIsogenyEichler$_{2^a}(\mathcal{O}_A, I, K, P, \theta, Q)$

```
Require: A left $\mathcal{O}_A$-ideal $I$ of reduced norm $2^a$, a left $\mathcal{O}_A$-ideal $K = \overline{I_2} + 2\mathcal{O}_A$, a generator $P$ of $E[2^a] \cap \ker(\hat{\varphi}_{I_2})$, an endomorphism $\theta \in \mathcal{O}_A \backslash (\mathbb{Z} + K)$ and the point $Q = \theta(P)$.
Ensure: $\varphi_1$ of degree $2^a$.
1: Select $\alpha \in I$ such that $I = \mathcal{O}_A\langle \alpha, 2^a \rangle$;
2: Compute $C, D$ such that $\alpha(C + D\theta) \in K$ and $\gcd(C, D, 2) = 1$;
3: Compute $\varphi_1$ of kernel $\langle [C]P + [D]Q \rangle$;
4: return $\varphi_1$.
```

[24]. This trick not only accelerates the signing phase but also the key generation phase. Note that this technique could also be adapted in the implementation of other isogeny-based protocols, such as M-SIDH and MD-SIDH [41].

### C. Precomputation for $\varphi_1$

In the first execution of the ideal-to-isogeny translation for $\sigma$, we compute $\varphi_1$ with $\mathcal{O}_A$, $I_2$ and $\varphi_{I_2}$. All of these are obtained in the key generation phase, and thus some procedures used to compute $\varphi_1$ can be saved via precomputation in the key generation phase. For example, the endomorphism $\theta \in \mathcal{O}_A$ can be computed in advance, allowing us to evaluate $Q = \theta(P)$ before signing. Indeed, except for $C$ and $D$, all the other information does not depend on the ideal $I_\sigma$. Therefore, one can precompute them to speed up the translation from the ideal $\langle I_\sigma, 2^a \rangle$ to the isogeny $\varphi_1$. As a result, we can compute $\varphi_1$ efficiently by Algorithm 6, which avoids large degree isogeny computations. Although the precomputation increases the required computational resources of key generation, it reduces the signing cost.

## V. Implementation Results

In this section, we present the implementation results of the procedures we have improved in the signing phase, and report the performance of the instantiation with $p_{1973}$ using our techniques. We also give a concrete comparison between the previous work and ours on efficiency. Based on the code[2] provided in [13], we compile and benchmark our code[3] on Intel(R) Core(TM) i9-12900K 3.20 GHz with TurboBoost and hyperthreading features disabled. Except for the improvements we mentioned in this paper, we also adapt some techniques proposed in the literature to further improve the implementation. For example, we employ the three-point ladder algorithm [42] when computing the kernel generator of the isogeny.

Table II reports the performance of our improved procedures in the signing phase. For elliptic curve discrete logarithm computations, we apply our new method to compute discrete logarithms in the group $\mu_{2^a}$ and set the base power $w = 5$. The results show that the performance is significantly accelerated with our techniques. It should be noted that all the procedures are executed multiple times during the key generation and signing phase. In addition, the verification phase frequently generate the second torsion point, thus our improved algorithms for torsion point generation also reduce the verifying cost.

TABLE II
Implementation Results of the Improved Procedures in the Signing Phase of SQIsign. The Results Are Expressed in Thousands of Clock Cycles

| Phase | $p_{6983}$ | | | $p_{3923}$ | | |
|---|---|---|---|---|---|---|
| | SQIsign2 [13] | This work | Speedup | SQIsign2 [13] | This work | Speedup |
| Computations for $s_1$ and $s_2$ | 4177 | 1260 | 69.8% | 6149 | 1287 | 79.1% |
| Torsion point generation | 881 | 604 | 31.4% | 605 | 400 | 33.9% |
| Isogeny computation of $\varphi_2$ | 30254 | 27439 | 9.3% | 23872 | 21628 | 9.4% |

TABLE III
Implementation Results of Each Phase in SQIsign. The Results Are Reported in Millions of Clock Cycles. We Execute 10 Times and Record the Average Costs

| Setting | Phase | SQIsign2 [13] | This work | | | |
|---|---|---|---|---|---|---|
| | | | without precomp. | Speedup | precomp. | Speedup |
| $p_{6983}$ | Keygen | 2029.1 | 1965.0 | 3.16% | 2039.1 | -0.5% |
| | Sign | 3671.1 | 3398.5 | 7.43% | 3312.5 | 9.77% |
| | Verify | 50.2 | 38.6 | 23.11% | 38.6 | 23.11% |
| $p_{3923}$ | Keygen | 361.5 | 329.6 | 8.82% | 399.6 | -10.54% |
| | Sign | 1677.7 | 1535.1 | 8.50% | 1477.6 | 11.93% |
| | Verify | 26.4 | 21.4 | 18.94% | 21.4 | 18.94% |

As shown in Table III, we improve the performance of all the procedures in SQIsign without the technique proposed in Section IV-C. When using the precomputation technique, the key generation phase is less efficient, but we further improve the signing phase. In particular, the signing performance is up to 18.02% faster than that of the previous work. This would be preferred in the case when the signer needs to sign a number of messages using the same secret key.

## VI. Conclusion

In this paper, we mainly focused on the ideal-to-isogeny translation in the signing phase of SQIsign, and proposed several novel techniques to enhance the performance. For each procedure we have considered, the improvements led to a significant speedup. The implementation results showed that we also improved the key generation phase and the verification phase of SQIsign. As a future work, we would like to explore how to further accelerate the implementation of SQIsign.

## Acknowledgment

The authors thank all the reviewers for their constructive comments. They also thank Jintai Ding for his valuable suggestions and proofreading an earlier version of this work.

## References

[1] D. Jao and L. De Feo, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies," in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2011, pp. 19–34.
[2] E. V. Flynn and Y. B. Ti, "Genus two isogeny cryptography," in *Post-Quantum Cryptography*. Cham, Switzerland: Springer, 2019, pp. 286–306.
[3] W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes, "CSIDH: An efficient post-quantum commutative group action," in *Advances in Cryptology—ASIACRYPT*. Cham, Switzerland: Springer, 2018, pp. 395–427.

[2]https://github.com/SQISign/the-sqisign
[3]https://github.com/LinKaizhan/FasterSQIsign

[4] L. Colo and D. Kohel, "Orienting supersingular isogeny graphs," *J. Math. Cryptol.*, vol. 14, no. 1, pp. 414–437, Oct. 2020.

[5] H. Onuki, "On oriented supersingular elliptic curves," *Finite Fields Their Appl.*, vol. 69, Jan. 2021, Art. no. 101777.

[6] L. De Feo, S. Dobson, S. D. Galbraith, and L. Zobernig, "SIDH proof of knowledge," in *Advances in Cryptology—ASIACRYPT*. Cham, Switzerland: Springer, 2022, pp. 310–339.

[7] J.-J. Chi-Dominguez. (2022). *A Note on Constructing SIDH-PoK-Based Signatures After Castryck-Decru Attack*. [Online]. Available: https://eprint.iacr.org/2022/1479

[8] L. De Feo and S. D. Galbraith, "SeaSign: Compact isogeny signatures from class group actions," in *Advances in Cryptology—EUROCRYPT 2019*. Cham, Switzerland: Springer, 2019, pp. 759–789.

[9] W. Beullens, T. Kleinjung, and F. Vercauteren, "CSI-FiSh: Efficient isogeny based signatures through class group computations," in *Advances in Cryptology—ASIACRYPT*. Cham, Switzerland: Springer, 2019, pp. 227–247.

[10] S. Atapoor, K. Baghery, D. Cozzo, and R. Pedersen, "CSI-SharK: CSI-FiSh with sharing-friendly keys," in *Information Security and Privacy*. Cham, Switzerland: Springer, 2023, pp. 471–502.

[11] S. D. Galbraith, C. Petit, and J. Silva, "Identification protocols and signature schemes based on supersingular isogeny problems," in *Advances in Cryptology—ASIACRYPT*. Cham, Switzerland: Springer, 2017, pp. 3–33.

[12] L. De Feo, D. Kohel, A. Leroux, C. Petit, and B. Wesolowski, "SQISign: Compact post-quantum signatures from quaternions and isogenies," in *Advances in Cryptology—ASIACRYPT*. Cham, Switzerland: Springer, 2020, pp. 64–93.

[13] L. De Feo, A. Leroux, P. Longa, and B. Wesolowski, "New algorithms for the deuring correspondence: Towards practical and secure SQISign signatures," in *Advances in Cryptology—EUROCRYPT*. Cham, Switzerland: Springer, 2023, pp. 659–690.

[14] P. Dartois, A. Leroux, D. Robert, and B. Wesolowski, "SQIsignHD: New dimensions in cryptography," in *Advances in Cryptology—EUROCRYPT*. Cham, Switzerland: Springer, 2024, pp. 3–32.

[15] W. Castryck and T. Decru, "An efficient key recovery attack on SIDH," in *Advances in Cryptology—EUROCRYPT 2023*. Cham, Switzerland: Springer, 2023, pp. 423–447.

[16] L. Maino, C. Martindale, L. Panny, G. Pope, and B. Wesolowski, "A direct key recovery attack on SIDH," in *Advances in Cryptology—EUROCRYPT*. Cham, Switzerland: Springer, 2023, pp. 448–471.

[17] D. Robert, "Breaking SIDH in polynomial time," in *Advances in Cryptology—EUROCRYPT 2023*. Cham, Switzerland: Springer, 2023, pp. 472–503.

[18] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology—CRYPTO' 86*. Cham, Switzerland: Springer, 1987, pp. 186–194.

[19] D. Kohel, K. Lauter, C. Petit, and J.-P. Tignol, "On the quaternion-isogeny path problem," *LMS J. Comput. Math.*, vol. 17, no. 1, pp. 418–432, 2014.

[20] J. Chavez-Saab et al. (2023). *SQIsign*. [Online]. Available: http://sqisign.org

[21] J. Voight, *Quaternion Algebras*. Cham, Switzerland: Springer, 2021.

[22] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Cham, Switzerland: Springer, 2009.

[23] J. Vélu, "Isogénies entre courbes elliptiques," *C. R. Acad. Sci., Paris, Sér. A*, vol. 273, pp. 238–241, 1971.

[24] D. J. Bernstein, L. de Feo, A. Leroux, and B. Smith, "Faster computation of isogenies of large prime degree," in *Proc. 14th Algorithmic Number Theory Symp.*, vol. 4, Jun. 2020, pp. 39–55.

[25] A. K. Pizer, "Ramanujan graphs and hecke operators," *Bull. Amer. Math. Soc.*, vol. 23, no. 1, pp. 127–137, 1990.

[26] S. Galbraith, *London Mathematical Society Lecture Note Series*. Cambridge, U.K.: Cambridge Univ. Press, 2005, pp. 183–214.

[27] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over GF($p$) and its cryptographic significance," *IEEE Trans. Inf. Theory*, vols. IT-24, no. 1, pp. 106–110, Jul. 1978.

[28] R. Azarderakhsh, D. Jao, K. Kalach, B. Koziel, and C. Leonardi, "Key compression for isogeny-based cryptosystems," in *Proc. 3rd ACM Int. Workshop ASIA Public-Key Cryptography*, May 2016, pp. 1–10.

[29] C. Costello, D. Jao, P. Longa, M. Naehrig, J. Renes, and D. Urbanik, "Efficient compression of SIDH public keys," in *Advances in Cryptology—EUROCRYPT*. Cham, Switzerland: Springer, 2017, pp. 679–706.

[30] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*. Boca Raton, FL, USA: CRC Press, 2008.

[31] R. Azarderakhsh et al. (2020). *Supersingular Isogeny Key Encapsulation*. [Online]. Available: http://sike.org

[32] M. Naehrig and J. Renes, "Dual isogenies and their application to public-key compression for isogeny-based cryptography," in *Advances in Cryptology—ASIACRYPT*. Cham, Switzerland: Springer, 2019, pp. 243–272.

[33] K. Lin, J. Lin, W. Wang, and C.-A. Zhao, "Faster public-key compression of SIDH with less memory," *IEEE Trans. Comput.*, vol. 72, no. 9, pp. 2668–2676, 2023.

[34] D. J. Bernstein and T. Lange. *Explicit-Formulas Database*. Accessed: 2007. [Online]. Available: http://www.hyperelliptic.org/EFD

[35] M. Scott and P. S. L. M. Barreto, "Compressed pairings," in *Advances in Cryptology—CRYPTO*. Cham, Switzerland: Springer, 2004, pp. 140–156.

[36] C. B. P. R. Crandall, *Prime Numbers: A Computational Perspective*. Cham, Switzerland: Springer, 2005.

[37] V. S. Miller, "The weil pairing, and its efficient calculation," *J. Cryptol.*, vol. 17, no. 4, pp. 235–261, Sep. 2004.

[38] K. Lin, W. Wang, L. Wang, and C.-A. Zhao. (2021). *An Alternative Approach for Computing Discrete Logarithms in Compressed SIDH*. [Online]. Available: https://eprint.iacr.org/2021/1528

[39] D. Husemöller, *Elliptic Curves*. Cham, Switzerland: Springer, 2004.

[40] C. Costello, P. Longa, and M. Naehrig, "Efficient algorithms for supersingular isogeny Diffie–Hellman," in *Advances in Cryptology—CRYPTO*. Cham, Switzerland: Springer, 2016, pp. 572–601.

[41] T. B. Fouotsa, T. Moriya, and C. Petit, "M-SIDH and MD-SIDH: Countering SIDH attacks by masking information," in *Advances in Cryptology—EUROCRYPT*. Cham, Switzerland: Springer, 2023, pp. 282–309.

[42] A. Faz-Hernández, J. López, E. Ochoa-Jiménez, and F. Rodríguez-Henríquez, "A faster software implementation of the supersingular isogeny Diffie–Hellman key exchange protocol," *IEEE Trans. Comput.*, vol. 67, no. 11, pp. 1622–1636, Nov. 2018.

**Kaizhan Lin** received the B.S. degree from Sun Yat-sen University, Guangdong, China, where he is currently pursuing the Ph.D. degree in mathematics. His research interests include isogeny-based cryptography and public-key cryptography.

**Weize Wang** received the B.S. and M.S. degrees from Sun Yat-sen University, Guangdong, China. He is currently pursuing the Ph.D. degree in cybersecurity with Fudan University, Shanghai, China. His research interests include isogeny-based cryptography and public-key cryptography.

**Zheng Xu** received the B.S. degree in mathematics from Shandong University and the Ph.D. degree in pure mathematics from USTC. He further enriched his academic portfolio through post-doctoral appointments at Tsinghua University and BIMSA. He is currently a Post-Doctoral Researcher with Hefei National Laboratory. His research interests include isogeny-based cryptography and computational number theory.

**Chang-An Zhao** received the bachelor's degree in electronical engineering, the master's degree in applied mathematics, and the Ph.D. degree in information science and technology from Sun Yat-sen University, Guangzhou, China, in 2001, 2005, and 2008, respectively. He is currently with the School of Mathematics, Sun Yat-sen University. His research interests include elliptic curve cryptography, post-quantum cryptography, and algebraic coding theory.