Attack on SHealS and HealS: The Second Wave of GPST

Steven D. Galbraith and Yi-Fu Lai(✉)

University of Auckland, Auckland, New Zealand
s.galbraith@auckland.ac.nz, ylai276@aucklanduni.ac.nz

Abstract. We cryptanalyse the isogeny-based public key encryption schemes SHealS and HealS, and the key exchange scheme HealSIDH of Fouotsa and Petit from Asiacrypt 2021.

1 Introduction

An important problem is to have an efficient and secure static-static key exchange protocol or public key encryption (PKE) from isogenies. A static-static protocol enables participants to execute the desired primitives without changing the public keys from time to time. This is possible and natural using CSIDH [CLM+18], which has been used to construct several competitive isogeny-based cryptographic primitives [BKV19, MOT20, EKP20, LGd21, BDK+22] while the counterparts are missing in the SIDH-based constructions. However due to subexponential attacks on CSIDH based on the Kuperberg algorithm [Kup05, Pei20], SIDH-related assumptions [JD11] might provide a more robust foundation[1]. Hence, an efficient protocol with a robust underlying assumption from isogenies is still an open problem.

The main bottleneck for SIDH-family schemes to achieve the static-static property boils down to the adaptive GPST attack [GPST16]. The attack enables malicious Bob to extract Alice's secret key bit by bit from each handshake and vice versa. The known countermeasures against the attack are to embed a zero-knowledge proof [UJ20] or to utilize the k-SIDH method [AJL17]. However, these countermeasures also inevitably incur multiple parallel isogeny computations so that the deduced schemes are not practical. To resolve this, Fouotsa and Petit [FP21] (Asiacrypt'21) presented a variant of SIDH with a novel key validation mechanism by using the commutativity of the isogeny diagram [Leo20]. The scheme requires fewer isogeny computations than SIKE [ACC+17] with the prime number doubled in length which still is far more efficient than the other known abovementioned solutions. In [FP21], it is claimed that the work gives the static-static key exchange and PKE solutions from isogenies which are immune to any adaptive attacks.

[1] We remark that the confidence of the SIDH-based protocols is still under debate due to the recent advance given by Castryck and Decru [CD22].

J. H. Cheon and T. Johansson (Eds.): PQCrypto 2022, LNCS 13512, pp. 399–421, 2022.
https://doi.org/10.1007/978-3-031-17234-2_19

In this work we refute the claim by presenting an adaptive attack against the protocols presented in [FP21]. Our attack builds on the flaw in the key validation mechanism, which is the core result [FP21] to construct SHealS, HealS, and HealSIDH. The attack can be viewed as a simple tweak of the GPST attack and, surprisingly, it takes the same number of oracle queries as the GPST attack against SIDH to adaptively recover a secret key. In other words, the additional key validation mechanism not only slows down the protocol with respect to the original SIDH scheme but also gives no advantage to the scheme in preventing adaptive attacks.

1.1 Concurrent Works

An exciting advance in isogeny cryptanalysis given by Castryck and Decru [CD22] gives a polynomial time key-recovery attack against the original SIDH [JD11] by exploiting the torsion points and the known endomorphism ring of E_0. The current version of the attack does not run in polynomial time against SHealS and HealS where the endomorphism ring is assumed to be unknown, as a potential patch suggested in [CD22] using a trusted set-up for the public curve. Whether the Castryck-Decru attack can be extended to the unknown endomorphism and run in polynomial time, of course, is worthwhile to be investigated further before jumping to conclusions.

1.2 Technical Overview

The cornerstone of our attack is the flaw originating in the proof of the main theorems for the key validation mechanism (Theorems 1 and 2 in [FP21]). The main idea of the mechanism exploits the nontrivial commutativity of the SIDH diagram [Leo20] (i.e. $\phi'_A\phi_B = \phi'_B\phi_A$ when Alice and Bob both behave honestly). For a given curve E_0, a natural number b and a basis $\{P_2, Q_2\}$ for $E_0[4^a]$ from the public parameter, the key validation mechanism checks the validity of three following relations:

$$\begin{aligned} e_{4^a}(R_a, S_a) &= e_{4^a}(P_2, Q_2)^{3^b}, \\ \phi'_A(R_a) &= [e_1]R_{ab} + [f_1]S_{ab} \in E_{AB}, \\ \phi'_A(S_a) &= [e_2]R_{ab} + [f_2]S_{ab} \in E_{AB}, \end{aligned}$$

where ϕ'_A is an isogeny from E_B with kernel $\langle [2^a]R_a + [\alpha 2^a]S_a \rangle \subset E_B$, $\{R_a, S_a\}$ and $\{R_{ab}, S_{ab}\}$ are bases for $E_B[4^a]$ and $E_{AB}[4^a]$ respectively, $(R_a, S_a, R_{ab}, S_{ab}, E_B, E_{AB})$ is the input given by Bob, and $(\alpha, e_1, f_1, e_2, f_2)$ is Alice's secret key. The first equation comes from the relations between isogenies and the Weil pairing. The last two equations are derived from the commutativity of the SIDH diagram [Leo20].

These relations will be satisfied when Bob produces the input honestly. In the security analysis in [FP21], to make another valid input, which is not obtained by taking negations of the curve points, is equivalent to solve four linear equations with four unknown variables (e_1, f_1, e_2, f_2) over the ring $\mathbb{Z}/4^a\mathbb{Z}$. Furthermore, Bob's input also has the restriction that $e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$ and ϕ'_A might vary with the choice of R_a and S_a. Therefore, it is deduced that Bob, without knowing Alice's secret, is not able to produce another valid input, which is not obtained by taking negations of the original input. In this way, since Bob, restricted by the mechanism, behaves honestly, the cryptosystem will be secure based on the hardness assumption.

However, for an adaptive attack, what malicious Bob wants to exploit is that Alice's behaviour is dependent on the secret. The proof in [FP21] neglects the spirit of the adaptive attack where malicious Bob can learn the desired information adaptively. For example, write $\mathbf{M} = \begin{pmatrix} e_1 & f_1 \\ e_2 & f_2 \end{pmatrix} \in M_{2\times 2}(\mathbb{Z}/4^a\mathbb{Z}), \mathbf{u} = (R_a\ S_a)^T$ and $\mathbf{v} = (R_{ab}\ S_{ab})^T$. We may therefore abuse the notation by writing $\phi'_A\mathbf{u} = \mathbf{M}\mathbf{v}$. As we will show in Sect. 3, by considering matrices $\mathbf{P}_1 = \begin{pmatrix} 1 & 0 \\ 2^{2a-1} & 1 \end{pmatrix}$ and $\mathbf{P}_2 = \mathbf{I}_2$, the relation $\mathbf{P}_1\mathbf{M} = \mathbf{M}\mathbf{P}_2$ holds if and only if $e_1 = f_1 = 0 \mod 2$. Hence, on input $(R'_a, S'_a, R'_{ab}, S'_{ab}, E_B, E_{AB})$ where $(R'_a\ S'_a)^T = \mathbf{P}_1\mathbf{u}$ and $(R'_{ab}\ S'_{ab})^T = \mathbf{P}_2\mathbf{v}$ the key validation mechanism will pass if and only if $\phi'_A\mathbf{P}_1\mathbf{u} = \mathbf{M}\mathbf{P}_2\mathbf{v}$ if and only if $e_1 = f_1 = 0 \mod 2$. Note that because $\det(\mathbf{P}_1) = 1$ and $(2^a\ 2^a)\mathbf{P}_1 = (c\ c)$ for some $c \in \mathbb{Z}_{2^a}$, the Weil pairing check will also pass and the isogeny used by the mechanism is still ϕ'_A. In this way, Bob learns one bit information of e_1 and f_1. Moreover, as we will show in Sect. 3, this is enough to recover the least significant bit of α.

On top of that, Bob can utilize the GPST attack in a "reciprocal" sense to extract further information further. If the least significant bit of α, denoted by α_0, is 1, the secret α is invertible over the ring $\mathbb{Z}/2^a\mathbb{Z}$. By further replacing R_a with $R'_a = R_a + [2^{2a-2}]R_a - [2^{2a-2}\alpha_0]S_a$, the validity of the second relation in the mechanism depends on the second least significant bit of α. However, $e_{4^a}(R'_a, S_a)$ will never satisfy the first relation. To overcome this, Bob will replace S_a with $[\alpha_0^{-1}2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a$ which can be used to extract the second least significant bit of α^{-1}, because the equality of the third equation depends on the second least significant bit of α^{-1}. Remark that, the isogeny used in the key validation mechanism is not necessarily the same ϕ'_A if the kernel is not $\langle [2^a]R_a + [\alpha 2^a]S_a \rangle$. In Sect. 4, we present the attack in details including the case where α is even.

Structure of this Paper. We begin in Sect. 2 with some preliminary background on elliptic curves, isogenies, a brief outline the fundamental scheme of [FP21], together with a few immediate properties of the scheme. We then introduce the method of using commutativity of matrices to extract the least significant bit of Alice's secret in Sect. 3. Based on the least significant bit information, a tweak of the GPST attack to recursively and adaptively recover Alice's secret is then deduced in Sect. 4. A brief summary is made in Sect. 5. We also provide in Appendix A a generalized attack against mechanism using commutativity of isogenies.

2 Preliminaries

Notations. We begin by introducing some notations that will be used throughout the paper. Let $\mathbf{O}$ represent the point at infinity of an elliptic curve, $\mathbb{N}$ be the set of natural numbers, and $\mathbb{Z}$ be the set of integers. For $n \in \mathbb{N}$, let $\mathbb{Z}_n$ defined to be $\mathbb{Z}/n\mathbb{Z}$ and $\mathbb{F}_n$ be the finite field of order n. For convenience, when we write $u \in \mathbb{Z}_n$, we consider u is a representative taken from $\{0, \cdots, n-1\} \subset \mathbb{Z}$. Similarly, when we write $u \mod n$, we consider the unique representative taken from $\{0, \cdots, n-1\} \subset \mathbb{Z}$. Also, for $n \in \mathbb{N}$, $e_n(\cdot, \cdot)$ represents the Weil e_n-pairing.

2.1 Elliptic Curves and Isogenies

An elliptic curve is a rational nonsingular curve of genus one with a distinguished point at infinity denoted by $\mathbf{O}$. An elliptic curve with $\mathbf{O}$ forms an additive commutative group. Let p be an odd prime number and q be a power of p. If E is an elliptic curve defined over $\mathbb{F}_q$, then $E(\mathbb{F}_q)$, collecting $\mathbb{F}_q$-rational points of E and $\mathbf{O}$, is a finite subgroup of E. Moreover, E is said to be supersingular if the endomorphism ring of E is a maximal order in a quaternion algebra. For $n \in \mathbb{N}$ coprime with p, the n-torsion subgroup $E[n]$, collecting points of order dividing n, is isomorphic to $\mathbb{Z}_n \oplus \mathbb{Z}_n$. The Weil e_n-pairing $e_n(\cdot, \cdot)$ is bilinear, alternating and nondegenerate.

An isogeny is a morphism between elliptic curves preserving the point at infinity. The kernel of an isogeny is always finite and defines the isogeny up to a power of the Frobenius map. We restrict our attention to separable isogenies (which induce separable extensions of function fields over $\mathbb{F}_q$) between supersingular elliptic curves defined over $\mathbb{F}_q$. Given a finite subgroup S of E, there exists a unique separable isogeny with kernel S from E to the codomain denoted by E/S which can be computed via Vélu's formulas. We refer to [Sil09] to get more exposed to the elliptic curve theory.

2.2 Brief Outline of HealSIDH Key Exchange

Both SHealS and HealS, introduced in [FP21], are PKE schemes building on the key exchange scheme HealSIDH with a key validation mechanism. Concretely, SHealS is a PKE scheme using the padding to encrypt the message where the padding is the hash value of the shared curve (j-invariant) obtained from HealSIDH. HealS is a variant of SHealS by changing the parameters. In other words, our adaptive attack on HealSIDH is applicable to both SHealS and HealS.

We briefly introduce HealSIDH with the key validation mechanism as shown in Fig. 1. The public parameter $\mathsf{pp} = (E_0, P_2, Q_2, P_3, Q_3, p, a, b)$ contains a supersingular curve E_0 defined over $\mathbb{F}_{p^2}$ with an unknown endomorphism ring and $(p, a, b) \in \mathbb{N}^3$ where p is a prime of the form $2^{2a}3^{2b}f - 1$ such that $2^a \approx 3^b$. The requirement of the unknown endomorphism prevents the torsion-point attack [dQKL+21] (and also [CD22]). The sets $\{P_2, Q_2\}$, $\{P_3, Q_3\}$ are bases for $E_0[4^a]$ and $E_0[9^b]$ respectively and $P_A = [2^a]P_2, Q_A = [2^a]Q_2, P_B = [3^b]P_3$, and

$Q_B = [3^b]Q_3$. Alice and Bob sample α and β uniformly at random from $\mathbb{Z}_{2^a}$ and $\mathbb{Z}_{3^b}$ respectively. Also, Alice and Bob compute $\phi_A : E_0 \to E_A = E_0/\langle P_A+[\alpha]Q_A\rangle$ and $\phi_B : E_0 \to E_B = E_0/\langle P_B + [\beta]Q_B\rangle$, respectively. Alice and Bob compute $(\phi_A(P_2), \phi_A(Q_2), \phi_A(P_3), \phi_A(Q_3))$ and $(\phi_B(P_3), \phi_B(Q_3), \phi_B(P_2), \phi_B(Q_2))$ respectively. Alice's and Bob's public keys are $(E_A, \phi_A(P_3), \phi_A(Q_3))$ and $(E_B, \phi_B(P_2), \phi_B(Q_2))$ respectively. Alice computes the canonical basis $\{R_A, S_A\}$ for $E_A[4^a]$ and represents $\phi_A(P_2) = [e_1]R_A + [f_1]S_A$ and $\phi_A(Q_2) = [e_2]R_A + [f_2]S_A$. Bob computes the canonical basis $\{R_B, S_B\}$ for $E_B[9^a]$ and represents $\phi_B(P_3) = [g_1]R_B + [h_1]S_B$ and $\phi_B(Q_3) = [g_2]R_B + [h_2]S_B$. Alice's and Bob's secret keys are $\mathsf{sk}_A = (\alpha, e_1, f_1, e_2, f_2)$ and $\mathsf{sk}_B = (\beta, g_1, h_1, g_2, h_2)$ respectively.

To establish a shared secret with Alice, Bob collects Alice's public key, denoted by (E_A, R_b, S_b), and computes $\phi'_B : E_A \to E_{AB} = E_A/\langle[3^b]R_b + [\beta 3^b]S_b\rangle$ together with $(\phi'_B(R_A), \phi'_B(S_A), \phi'_B(R_b), \phi'_B(S_b))$. He sends $(R_{ab} = \phi'_B(R_A), S_{ab} = \phi'_B(S_A))$ to Alice.

Upon receiving (R_{ab}, S_{ab}) from Bob, Alice collects Bob's public key (E_B, R_a, S_a). She computes $\phi'_A : E_B \to E_{BA} = E_B/\langle[2^a]R_a + [\alpha 2^a]S_a\rangle$ together with $(\phi'_A(R_B), \phi'_A(S_B), \phi'_A(R_a), \phi'_A(S_a))$. If $e_{4^a}(R_a, S_a) \neq e_{4^a}(P_2, Q_2)^{3^b}$, $\phi'_A(R_a) \neq [e_1]R_{ab} + [f_1]S_{ab}$, or $\phi'_A(S_a) \neq [e_2]R_{ab} + [f_2]S_{ab}$, then Alice aborts (the session). Otherwise, she sends $(R_{ba} = \phi'_A(R_B), S_{ba} = \phi'_A(S_B))$ to Bob and keeps the j-invariant j_{BA} of E_{BA} as the shared secret.

Similarly, upon receiving (R_{ba}, S_{ba}), Bob aborts if $e_{9^b}(R_b, S_b) \neq e_{9^b}(P_3, Q_3)^{2^a}$, $\phi'_B(R_b) \neq [g_1]R_{ba} + [h_1]S_{ba}$, or $\phi'_B(S_b) \neq [g_2]R_{ba} + [h_2]S_{ba}$, If not he takes the j-invariant of E_{AB} as the shared secret.

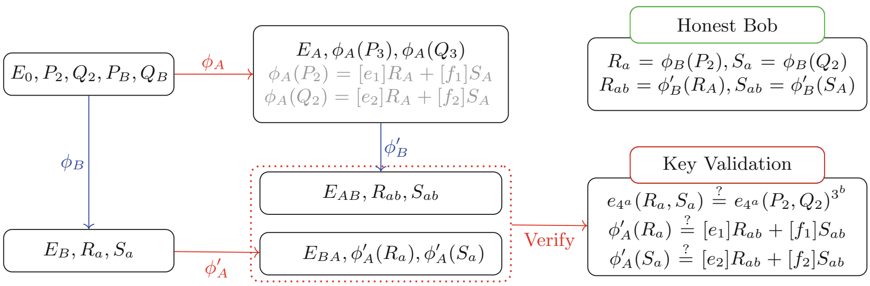

Fig. 1. The outline of HealSIDH with the key validation mechanism. The upper right box shows the points honest Bob will compute. The lower right box is the key validation process used by Alice to verify the public key given by Bob. The evaluations of $\phi_A(P_2), \phi_A(Q_2)$ are secretly computed by Alice and the coefficients e_1, f_1, e_2, f_2 are included in her secret key.

Remark 1. In the real protocol, instead of giveing R_{ab}, S_{ab} directly, Bob will give the coordinates of them with respect to the canonical basis of $E_{AB}[4^a]$. Otherwise, the secretly shared curve E_{AB} can be recontructed by an eavesdropper by computing its Montgomery coefficient $A_{E_{AB}} = (y(R_{ab})^2 - x(R_{ab})^3 -$

$x(R_{ab}))/x(R_{ab})^2$. For simplicity we ignore this detail and pretend Bob does send the points R_{ab} and S_{ab} to Alice. Hence, for the convenience, we may assume Bob sends the entire points R_{ab}, S_{ab} to Alice.

We have the following two immediate results.

Proposition 2. *If Bob honestly generates $R_a = \phi_B(P_2)$, $S_a = \phi_B(Q_2)$, $R_{ab} = \phi'_B(R_A)$ and $S_{ab} = \phi'_B(S_A)$, then $\{R_{ab}, S_{ab}\}$ is a basis of $E_{AB}[4^a]$ and $\{R_a, S_a\}$ is a basis of $E_B[4^a]$.*

Proof. Since $[4^a]R_a = \phi_B([4^a]P_2) = \mathbf{O}$ and $[4^a]S_a = \phi_B([4^a]Q_2) = \mathbf{O}$, both R_a and S_a are in $E_B[4^a]$. Due to $e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$, we know $e_{4^a}(R_a, S_a)$ is a primitive 4^a-th root of unity. Similarly, since $[4^a]R_{ab} = \phi'_B([4^a]R_A) = \mathbf{O}$ and $[4^a]S_{ab} = \phi'_B([4^a]S_A) = \mathbf{O}$, both R_{ab} and S_{ab} are in $E_{AB}[4^a]$. Due to $e_{4^a}(R_{ab}, S_{ab}) = e_{4^a}(R_A, S_A)^{3^b}$, we know $e_{4^a}(R_{ab}, S_{ab})$ is a primitive 4^a-th root of unity. Therefore, the result follows.

Lemma 3. *Let e_1, e_2, f_1, f_2 be defined as above and $\alpha \in \mathbb{Z}_{2^a}$ be Alice's secret key i.e.* $\ker(\phi_A) = \langle [2^a]P_2 + [\alpha 2^a]Q_2 \rangle$. *If Alice follows the protocol specification, then $e_1 + \alpha e_2 = f_1 + \alpha f_2 = 0 \mod 2^a$.*

Proof. Given $\phi_A(P_2) = [e_1]R_A + [f_1]S_A$ and $\phi_A(Q_2) = [e_2]R_A + [f_2]S_A$, we have $\mathbf{O} = \phi_A([2^a]P_2 + [\alpha 2^a]Q_2) = [2^a e_1 + \alpha 2^a e_2]R_A + [2^a f_1 + \alpha 2^a f_2]S_A = [e_1 + \alpha e_2]([2^a]R_a) + [f_1 + \alpha f_2]([2^a]S_A)$.

Note that $\{[2^a]R_A, [2^a]S_A\}$ is a basis for $E_A[2^a]$ due to $\{R_A, S_A\}$ being a basis for $E_A[4^a]$. Therefore, $e_1 + \alpha e_2 = f_1 + \alpha f_2 = 0 \mod 2^a$.

Modeling. We consider adaptive attacks against HealSIDH throughout this paper. Bob, as a malicious adversary, is given access to an oracle $\mathcal{O}_{\mathsf{sk}_A} \rightarrow 0/1$ taking as input $(R_a, S_a, R_{ab}, S_{ab}, E_B, E_{AB})$ with the relations specified as above. For simplicity, we denote the oracle by $\mathcal{O}$ and omit curves E_B, E_{AB} from the input when they are clear from the context. The oracle $\mathcal{O}$ returns 1 if and only if the following three equations hold:

$$e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}, \tag{1}$$

$$\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}, \tag{2}$$

$$\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}, \tag{3}$$

where ϕ'_A is an isogeny from E_B with kernel $\langle [2^a]R_a + [\alpha 2^a]S_a \rangle \in E_B$.

When Bob follows the protocol specification, the three equations hold naturally. The goal of malicious Bob in our attack is to recover Alice's core secret α by adaptively manipulating his input.

The flaw of the claim in [FP21] comes from the main theorem (Theorem 2) for the key validation mechanism. Theorem 2 of [FP21] states that if on input $(\widetilde{R_a}, \widetilde{S_a}, \widetilde{R_{ab}}, \widetilde{S_{ab}})$ the oracle returns 1, then there are only 16 forms of $(\widetilde{R_a}, \widetilde{S_a}, \widetilde{R_{ab}}, \widetilde{S_{ab}})$ as follows:

$$(\widetilde{R_a}, \widetilde{S_a}, \widetilde{R_{ab}}, \widetilde{S_{ab}}) = ([\pm 1]\phi_B(P_2), [\pm 1]\phi_B(Q_2), [\pm 1]\phi'_B(R_A), [\pm 1]\phi'_B(S_A)),$$

where ϕ_B, ϕ'_B are the isogenies computed by Bob following the protocol specification. We will immediately show this is not true in the next section.

3 Parity Recovering

In this section, we consider the least significant bits of e_1, e_2, f_1, f_2 and α. We can recover the least significant bit of α with one oracle query by relying the relations given by Lemma 3.

Say Bob computes ϕ_B, ϕ'_B honestly. The attack presented in this section and the next section relies on following facts:

- $\{P_2, Q_2\}$, is a basis for $E_0[4^a]$.
- $\{R_{ab}, S_{ab}\} = \{\phi'_B(R_A), \phi'_B(S_A)\}$ is a basis of $E_{AB}[4^a]$ (Proposition 2).
- $\{R_a, S_a\} = \{\phi_B(P_2), \phi_B(Q_2)\}$ is a basis of $E_B[4^a]$ (Proposition 2).
- $e_1 + \alpha e_2 = f_1 + \alpha f_2 = 0 \mod 2^a$ (Lemma 3).

The high-level idea in this section is simple. Assume Alice and Bob follows the protocol specification. Write $\mathbf{M} = \begin{pmatrix} e_1 & f_1 \\ e_2 & f_2 \end{pmatrix} \in M_{2\times 2}(\mathbb{Z}_{4^a}), \mathbf{u} = (R_a \ \ S_a)^T$ and $\mathbf{v} = (R_{ab} \ \ S_{ab})^T$. Recall that $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$ where R_a, S_a, R_{ab}, S_{ab} are honestly generated by Bob. We may abuse the notation by writing $\phi'_A\mathbf{u} = \mathbf{M}\mathbf{v}$ based on Eqs. (2) and (3). The idea is to find a pair of particular square matrices $\mathbf{P}_1, \mathbf{P}_2 \in M_{2\times 2}(\mathbb{Z}_{4^a})$ where $\mathbf{P}_1$ is of determinant 1 such that the commutativity of $\mathbf{P}_1\mathbf{M} = \mathbf{M}\mathbf{P}_2$ is conditioned on the information (parity for instance) to be extracted from $\mathbf{M}$. Let $(R'_a \ \ S'_a)^T = \mathbf{P}_1\mathbf{u}$ and $(R'_{ab} \ \ S'_{ab})^T = \mathbf{P}_2\mathbf{v}$. On input $(R'_a, S'_a, R'_{ab}, S'_{ab})$ the oracle returns 1 if $\mathbf{M}$ satisfies the commutativity condition $\mathbf{P}_1\mathbf{M} = \mathbf{M}\mathbf{P}_2$, because $\mathbf{P}_1\phi'_A\mathbf{u} = \phi'_A\mathbf{P}_1\mathbf{u} = \mathbf{P}_1\mathbf{M}\mathbf{v} = \mathbf{M}\mathbf{P}_2\mathbf{v}$ holds. Remark that the determinant 1 of $\mathbf{P}_1$ ensures the new pair $(R'_a \ \ S'_a)$ will satisfy the Weil pairing verification Eq. (1). Futhermore, we require $(2^a \ \ \alpha 2^a)\mathbf{P}_1 = (c \ \ c)$ for some $c \in \mathbb{Z}_{2^a}$ so that the isogeny used by the oracle is still the one with the kernel $\langle [2^a]R_a + [\alpha 2^a]S_a \rangle$.

Though there are 2^4 combinations of the least significant bits of e_1, e_2, f_1, f_2. The following lemma shows that when Alice generates them honestly, there are only six patterns.

Lemma 4. *If Alice produces $\phi_A(P_2)$ and $\phi_A(Q_2)$ honestly, then there are only 6 possible patterns of parities of e_1, e_2, f_1, f_2:*

1. $f_2 = 1 \mod 2$ *and* $e_2 = e_1 = f_1 = 0 \mod 2$,
2. $e_2 = 1 \mod 2$ *and* $e_1 = f_1 = f_2 = 0 \mod 2$,
3. $e_2 = f_2 = 1 \mod 2$ *and* $e_1 = f_1 = 0 \mod 2$,
4. $f_1 = f_2 = 1 \mod 2$ *and* $e_1 = e_2 = 0 \mod 2$,

5. $e_1 = e_2 = 1 \mod 2$ *and* $f_1 = f_2 = 0 \mod 2$*,*
6. $e_1 = e_2 = f_1 = f_2 = 1 \mod 2$*.*

Proof. Recall $e_{4^a}(\phi_A(P_2), \phi_A(Q_2)) = e_{4^a}(P_2, Q_2)^{2^a} = e_{4^a}(R_A, S_A)^{e_1 f_2 - e_2 f_1}$. Since both $\{P_2, Q_2\}$ and $\{R_A, S_A\}$ are bases for $E_0[4^a], E_A[4^a]$ respectively, both $e_{4^a}(P_2, Q_2)$ and $e_{4^a}(R_A, S_A)$ are primitive 4^a-th roots of unity. Given

$$e_{4^a}(R_A, S_A)^{2^a(e_1 f_2 - e_2 f_1)} = 1,$$

we have $e_1 f_2 - e_2 f_1 = 0 \mod 2^a$.

Furthermore, e_2, f_2 cannot be both even. Recall $\phi(Q_2) = e_2 R_A + f_2 S_A$. Suppose for the purpose of contradiction that both e_2 and f_2 are even. Then, $[2^{2a-1}]\phi_A(Q_2) = \mathbf{O}$, which implies $\ker(\phi_A) = \langle P_2 + [\alpha]Q_2 \rangle$ contains $[2^{2a-1}]Q_2$. That is, $[k]P_2 + [k\alpha]Q_2 = [2^{2a-1}]Q_2$ for some $k \in \mathbb{Z}_{2^a}$, so $k = 0$. This contradicts the fact that $\{P_2, Q_2\}$ is a basis for $E_0[4^a]$. The result follows.

We order the six cases according to the lemma above. The following lemmata indicate that we can divide the overall cases into two partitions: $\{$*Case* 1, *Case* 2, *Case* 3$\}$ and $\{$*Case* 4, *Case* 5, *Case* 6$\}$ with 1 oracle query.

Lemma 5. *Assume Bob honestly generates* $R_a, S_a, R_{ab}, S_{ab}, E_B, E_{AB}$*. On input* $(\widetilde{R_a}, \widetilde{S_a}, R_{ab}, S_{ab})$*, where* $\widetilde{R_a} = R_a$ *and* $\widetilde{S_a} = [2^{2a-1}]R_a + S_a$ *the oracle returns* 1 *only for Cases 1 to 3.*

Proof. Firstly, the isogeny ϕ'_A computed by the oracle is the same one used by Alice in the honest execution. This is because both kernels are the same:

$$\langle [2^a]R_a + [\alpha 2^a]S_a \rangle = \langle [2^a]\widetilde{R_a} + [\alpha 2^a]\widetilde{S_a} \rangle.$$

Therefore, since R_a, S_a, R_{ab}, S_{ab} are honestly generated, we may assume $e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$, $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, and $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$.

For Eq. (1), since $e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$, we have

$$e_{4^a}(\widetilde{R_a}, \widetilde{S_a}) = e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}.$$

Given $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$ and $R_{ab}, S_{ab} \in E_{AB}[2^a]$, we have

$$\begin{aligned} \phi'_A(\widetilde{R_a}) - [e_1]R_{ab} - [f_1]S_{ab} &= \mathbf{O}, \\ \phi'_A(\widetilde{S_a}) - [e_2]R_{ab} - [f_2]S_{ab} &= [2^{2a-1}e_1]R_{ab} + [2^{2a-1}f_1]S_{ab}. \end{aligned}$$

Recall that $\{R_{ab}, S_{ab}\}$ is a basis. Therefore, the oracle returns 1 if and only if $[2^{2a-1}e_1]R_{ab} + [2^{2a-1}f_1]S_{ab} = \mathbf{O}$ or, equivalently, $e_1 = f_1 = 0 \mod 2$. The result follows.

Lemma 6. *Assume Bob honestly generates* $R_a, S_a, R_{ab}, S_{ab}, E_B, E_{AB}$*. On input* $(\widetilde{R_a}, \widetilde{S_a}, R_{ab}, S_{ab})$*, where* $\widetilde{R_a} = [1 + 2^{2a-1}]R_a - [2^{2a-1}]S_a$ *and* $\widetilde{S_a} = [2^{2a-1}]R_a + [1 - 2^{2a-1}]S_a$ *the oracle returns* 1 *only for Cases 4 to 6.*

Proof. Firstly, the isogeny ϕ'_A computed by the oracle is the same one used by Alice in the honest execution. This is because both kernels are the same:

$$\langle [2^a]R_a + [\alpha 2^a]S_a\rangle = \langle [2^a]\widetilde{R_a} + [\alpha 2^a]\widetilde{S_a}\rangle.$$

Therefore, since R_a, S_a, R_{ab}, S_{ab} are honestly generated, we may assume $e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$, $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, and $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$.

For Eq. (1), since $e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$, we have

$$\begin{aligned}
&e_{4^a}(\widetilde{R_a}, \widetilde{S_a})\\
&= e_{4^a}([1 + 2^{a-1}]R_a - [2^{a-1}]S_a, [2^{a-1}]R_a + [1 - 2^{a-1}]S_a)\\
&= e_{4^a}(R_a, S_a)^{1-2^{2a-2}+2^{2a-2}}\\
&= e_{4^a}(P_2, Q_2)^{3^b}.
\end{aligned}$$

Given $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$ and $R_{ab}, S_{ab} \in E_{AB}[2^a]$, we have

$$\begin{aligned}
\phi'_A(\widetilde{R_a}) - [e_1]R_{ab} - [f_1]S_{ab} &= [2^{2a-1}]([e_1]R_{ab} + [f_1]S_{ab} + [e_2]R_{ab} + [f_2]S_{ab}),\\
\phi'_A(\widetilde{S_a}) - [e_2]R_{ab} - [f_2]S_{ab} &= [2^{2a-1}]([e_1]R_{ab} + [f_1]S_{ab} + [e_2]R_{ab} + [f_2]S_{ab}).
\end{aligned}$$

Recall that $\{R_{ab}, S_{ab}\}$ is a basis of $E_{AB}[2^a]$. Therefore, the oracle returns 1 if and only if $e_1 = e_2 \mod 2$ and $f_1 = f_2 \mod 2$. The result follows.

The cases $\{Case\,1,\ Case\,2,\ Case\,3\}$ occur if and only if the least significant bit of α is 0 by Lemma 4. In fact, by choosing particular matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, one can precisely recover all parities of e_1, e_2, f_1 and f_2. However, by Lemma 4, we do not bother to find them all since the information given in Lemma 5 already is sufficient to recover the least significant bit of α. In the next section, we will present a variant of the GPST attack. We start with the least significant bit of α to recover each higher bit with one oracle query for each.

4 Recover the Secret

In this section, we present a variant of the GPST attack to recover the secret α based on the knowledge extracted from the previous section. The high-level idea is to use the GPST attack in a "reciprocal" manner. Recall that Bob has two following equations when he generates the points $(R_a, S_a, R_{ab}, S_{ab})$ honestly:

$$\begin{aligned}
\phi'_A(R_a) &= [e_1]R_{ab} + [f_1]S_{ab},\\
\phi'_A(S_a) &= [e_2]R_{ab} + [f_2]S_{ab},
\end{aligned}$$

where $\ker(\phi'_A) = \langle [2^a]R_a + [2^a\alpha]S_a\rangle$.

To extract the second least significant bit of $-\alpha$, denoted by α_1, based on the least bit α_0, we consider $\phi'_A(R_a + [2^{2a-2}]R_a - [2^{2a-2}\alpha_0]S_a) = [e_1]R_{ab} + [f_1]S_{ab} + ([2^{2a-2}e_1 - 2^{2a-2}\alpha_0 e_2]R_{ab} + [2^{2a-2}f_1 - 2^{2a-2}\alpha_0 f_2]S_{ab})$ where the purpose of

$[2^{2a-2}\alpha_0]S_a$ is to eliminate the lower bit. Note that $([2^{2a-2}e_1 - 2^{2a-2}\alpha_0 e_2]R_{ab} + [2^{2a-2}f_1 - 2^{2a-2}\alpha_0 f_2]S_{ab}) = ([\alpha_1 2^{2a-1}][e_2]R_{ab} + [\alpha_1 2^{2a-1}][f_2]S_{ab})$ because $e_1 + \alpha e_2 = f_1 + \alpha f_2 = 0 \mod 2^a$ and $\{R_a, S_a\}$ is a basis for $E_B[4^a]$ (Lemma 3 and Proposition 2). By Lemma 4, since e_2 and f_2 cannot be both even, at least one of $[2^{2a-1}e_2]R_{ab}$ and $[2^{2a-1}f_2]S_{ab}$ is of order 2. It follows that the equation

$$\phi'_A(R_a + [2^{2a-2}]R_a - [2^{2a-2}\alpha_0]S_a) = [e_1]R_{ab} + [f_1]S_{ab}$$

holds if and only if $\alpha_1 = 0$.

Unfortunately, querying the oracle on input $(R_a+[2^{2a-2}]R_a-[2^{2a-2}\alpha_0]S_a, S_a, R_{ab}, S_{ab})$ will always return 0 so that Bob cannot obtain any useful information. This is because $e_{4^a}(R_a+[2^{2a-2}]R_a-[2^{2a-2}\alpha_0]S_a, S_a)$ never equals $e_{4^a}(P_2, Q_2)^{3^b}$. In other words, if Bob does so, he will always get $\perp$ from Alice. To resolve this, we use the idea of "reciprocal". Assume α is invertible modulo 2^a. Bob will craft a point replacing S_a for *recovering* $\alpha^{-1} \mod 2^a$ *at the same time*. Concretely, Bob computes $\hat{\alpha} = \alpha_0^{-1} \mod 4$. For the same reasoning as above, the equation

$$\phi'_A(\hat{\alpha}[2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a) = [e_2]R_{ab} + [f_2]S_{ab}$$

holds if and only if $\alpha^{-1} = \hat{\alpha} \mod 4$ if and only if $\alpha_1 = 0$.

Moreover, $e_{4^a}(R_a + [2^{2a-2}]R_a - [2^{2a-2}\alpha_0]S_a, \hat{\alpha}[2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a) = e_{4^a}(R_a, S_a)$. Therefore, by sending $(R_a + [2^{2a-2}]R_a - [2^{2a-2}\alpha_0]S_a, \hat{\alpha}[2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a, R_{ab}, S_{ab})$ to Alice, Bob can know whether $\alpha_1 = 0$. However, α is not necessarily odd. We have to use unbalanced powers of 2 on each query and introduce the concept of *quasi-inverse* elements.

Remark 7. On input $(R_a + [2^{2a-2}]R_a - [2^{2a-2}\alpha_0]S_a, \hat{\alpha}[2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a, R_{ab}, S_{ab})$, honest Alice will use the same isogeny ϕ'_A because $\langle[2^a](R_a + [2^{2a-2}]R_a - [2^{2a-2}\alpha_0]S_a) + [\alpha 2^a](\hat{\alpha}[2^{2a-2}]R_a + [1 - 2^{2a-2}]S_a)\rangle = \langle[2^a]R_a + [\alpha 2^a]S_a\rangle$. The same kernel will therefore derive the same isogeny ϕ'_A.

4.1 Quasi-Inverse Element

Definition 8. *Let p be a prime and $a \in \mathbb{N}$. For an element $u \in \mathbb{Z}$, a p^a-*quasi-inverse element *of u is a non-zero element $v \in \mathbb{Z}_{p^a}$ such that $uv = p' \mod p^a$ where p' is the maximal power of p dividing u.*

When $a = 1$, every element obviously has a *p-quasi-inverse element* by taking either its inverse over $\mathbb{Z}_p$ or 1. Unlike the inverse over a ring, a quasi-inverse is not necessarily unique. For instance, $1, 9, 17$ and 25 are 2^5-quasi-inverse elements of 4 over $\mathbb{Z}_{32}$. Also, if $u = 0$, any non-zero element can be its quasi-inverse.

A non-zero element being not a unit of $\mathbb{Z}_{p^a}$ can still have a p^a-quasi-inverse element. However, a non-zero element v in $\mathbb{Z}_{p^a}$ being a p^a-quasi-inverse element for a non-zero integer in $\mathbb{Z}_{p^a}$ implies v is a unit of $\mathbb{Z}_{p^a}$.

Proposition 9. *Let p be a prime and $a \in \mathbb{N}$. For $u \in \mathbb{Z}$, a non-zero element over $\mathbb{Z}_{p^a}$, any p^a-quasi-inverse element of u is a unit of $\mathbb{Z}_{p^a}$.*

Proof. Write $u = u'p^j$ where $u', j \in \mathbb{Z}$ and u' is not divisible by p and $j < a$. Say there exists $v \in \mathbb{Z}_{p^a}$ such that $uv = p^j \mod p^a$. Since u is a non-zero element over $\mathbb{Z}_{p^a}$, we know $a > j$ so that $(u/p^j)v = 1 \mod p^{j-a}$. It follows that v is not divided by p, so v is a unit of $\mathbb{Z}_{p^a}$.

In fact, for any $u \in \mathbb{Z}_{p^a}$ where $p^j \mid u$ and $p^{j+1} \nmid u$ for some non-negative integer j, one can always find a p^a-quasi-inverse by taking $v = (u/p^j)^{-1} \in \mathbb{Z}_{p^{a-j}}$ and naturally lifting v to $\mathbb{Z}_{p^a}$ Therefore, we may let $\mathsf{QuasiInv}(u, p, i)$ be an efficient algorithm outputting a p^i-quasi-inverse element of u and restrict it to output 1 when $p^i \mid u$.

Remark 10. Looking ahead, in our attack, we need to compute 2^{i+1}-quasi-inverse elements for either α_l or $\alpha_l + 2^i$ in the i-th iteration, where $\alpha_l = \alpha \mod 2^i$ has been extracted in the previous iterations. In a more general case where the prime 2 is replaced by $q \in \mathbb{N}$, the attack enumerates q^{i+1}-quasi-inverse elements for $\alpha_l + tq^i$ for every $t \in \{0, \cdots, q-1\}$, which corresponds to guess whether the next digit is t or not. See Appendix A for more details.

4.2 Attack on HealS and SHealS

The algorithm in Fig. 2 together with Theorem 12 provides an iterative approach for recovering α. It requires one oracle query to recover each bit of α in each iteration. We need the following lemma to prove the main theorem.

Lemma 11. *Let $(\alpha, e_1, f_1, e_2, f_2)$ denote Alice's HealSIDH secret key as Sect. 2.2. For any $i \in \{1, \ldots, a-1\}$, write $-\alpha = \alpha_l + 2^i\alpha_i \pmod{2^{i+1}}$ where $\alpha_l \in \mathbb{Z}_{2^i}$ and $\alpha_i \in \mathbb{Z}_2$. Let $\hat{\alpha}_l$ be a 2^{i+1}-quasi-inverse element of α_l such that $\hat{\alpha}_l\alpha_l = 2^j \mod 2^{i+1}$. Then, $\alpha_i = 0$ if and only if each of the following two equations is true:*

$$e_1 - \alpha_l e_2 = f_1 - \alpha_l f_2 \ = 0 \mod 2^{i+1} \tag{4}$$
$$\hat{\alpha}_l e_1 - 2^j e_2 = \hat{\alpha}_l f_1 - 2^j f_2 \ = 0 \mod 2^{i+1} \tag{5}$$

Proof. By Lemma 3, we have $e_1 - \alpha_l e_2 = -\alpha e_2 - \alpha_l e_2 \mod 2^{i+1}$ and $f_1 - \alpha_l f_2 = -\alpha f_2 - \alpha_l f_2 \mod 2^{i+1}$. By Lemma 4, not both e_2 and f_2 are divisible by 2. Therefore, the first equation is zero if and only if $\alpha_i = 0$.

Similarly, by Lemma 3, we have $\hat{\alpha}_l e_1 - 2^j e_2 = \hat{\alpha}_l \alpha e_2 - 2^j e_2 = \hat{\alpha}(\alpha_l + \alpha_i 2^i)e_2 - 2^j e_2 = \hat{\alpha}\alpha_i e_2 2^i \mod 2^{i+1}$. Also, $\hat{\alpha}_l f_1 - 2^j f_2 = \hat{\alpha}\alpha_i f_2 2^i \mod 2^{i+1}$. By Lemma 4 and Proposition 9, not both $e_2\hat{\alpha}$ and $f_2\hat{\alpha}$ are divisible by 2. Therefore, the second equation is zero if and only if $\alpha_i = 0$.

Theorem 12. *Assume Alice follows the protocol specification. The algorithm in Fig. 2 returns α in Alice's secret key.*

Proof. We are going to prove the theorem by induction on i for the i-th bit of α where $i < a$. Write $-\alpha = \alpha_l + 2^i\alpha_i \in \mathbb{Z}_{2^{i+1}}$ for some $i \in \{1, \ldots, a-1\}$ where

Algorithm: $\mathsf{Recover}(\mathsf{pp}, \mathsf{sk}_B, \alpha_0)$

Input: pp public parameter of the protocol, sk_B the secret key of Bob, $\alpha_0 = \alpha \mod 2$

Given: access to an oracle $\mathcal{O}(R_a, S_a, R_{ab}, S_{ab}; E_B, E_{AB}) \to 0/1$ returns 1 iff the following equations hold:

$$e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2),$$
$$\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab},$$
$$\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab},$$

where ϕ'_A is an isogeny from E_B with kernel $\langle [2^a]R_a + [\alpha 2^a]S_a \rangle \in E_B$.

Ensure: Alice's secret key α

Compute $(R_a, S_a, R_{ab}, S_{ab}) \leftarrow (\phi_B(P_2), \phi_B(Q_2), \phi'_B(R_A), \phi'_B(S_A))$ by following the protocol specification using sk_B.
Obtain a from pp.
Obtain $\alpha_l \leftarrow \alpha_0$.
$i = 1$
$j = \perp$ ▷ j will indicate the maximal power of 2 dividing α.
if $\alpha_l = 1$ **then** $j \leftarrow 0$
while $i < a$ **do**
  if $\alpha_l = 0$ **then**
    $(\widetilde{R_a}, \widetilde{S_a}) \leftarrow ([1 + 2^{2a-1}]R_a, [2^{2a-i-1}]R_a + [1 - 2^{2a-1}]S_a)$
    $c \leftarrow \mathcal{O}(\widetilde{R_a}, \widetilde{S_a}, R_{ab}, S_{ab})$
    $c \leftarrow 1 - c$
    if $c = 0$ **then** $j \leftarrow i$ ▷ Assert 2^j is the maximal power of 2 dividing α.
  else
    $\hat{\alpha}_l \leftarrow \mathsf{QuasiInv}(\alpha_l, 2, i+1)$ ▷ $\hat{\alpha}_l(\alpha_l) = 0$ or $2^j \mod 2^{i+1}$
    $\widetilde{R_a} \leftarrow [1 + 2^{2a-i+j-1}]R_a - [\alpha_l 2^{2a-i+j-1}]S_a$
    $\widetilde{S_a} \leftarrow [\hat{\alpha}_l 2^{2a-i-1}]R_a + [1 - 2^{2a-i+j-1}]S_a$
    $c \leftarrow \mathcal{O}(\widetilde{R_a}, \widetilde{S_a}, R_{ab}, S_{ab})$
  if $c \neq 1$ **then** ▷ Assert i-th bit of α is 1.
    $\alpha_l \leftarrow \alpha_l + 2^i$
return $-\alpha_l$

Fig. 2. An algorithm to recover the secret α in $\mathsf{sk}_A = (\alpha, e_1, f_1, e_2, f_2)$.

$\alpha_l \in \mathbb{Z}_{2^i}$ and $\alpha_i \in \mathbb{Z}_2$ represent the bits that have been recovered and the next bit to be recovered respectively. Since we have assumed the correctness of the given least significant bit of α, it suffices to show that given α_l the extraction of α_i, the i-th bit of α, is correct in each iteration of the while-loop of Fig. 2.

Firstly, within each query, the isogeny ϕ'_A computed by the oracle is the same because the kernels are all identical:

$$\begin{aligned}\langle[2^a]R_a + [\alpha 2^a]S_a\rangle &= \langle[2^a]([1+2^{2a-1}]R_a - [t2^{2a-i-1}]S_a)\\ &\quad + [\alpha 2^a]([t'2^{2a-i-1}]R_a + [1-2^{2a-1}]S_a)\rangle\\ &= \langle[2^a]([1+2^{2a-i+j-1}]R_a - [t2^{2a-i+j-1}]S_a)\\ &\quad + [\alpha 2^a]([t'2^{2a-i-1}]R_a + [1-2^{2a-i+j-1}]S_a)\rangle,\end{aligned}$$

for any $t, t' \in \mathbb{Z}_{2^a}$ where $i, j \in \mathbb{Z}_a$. Therefore, since R_a, S_a, R_{ab}, S_{ab} are honestly generated, we may assume $e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$, $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, and $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$.

Also, every input satisfies Eq. (1). Since $e_{4^a}(R_a, S_a) = e_{4^a}(P_2, Q_2)^{3^b}$, we have for any $\hat{\alpha}_l \in \mathbb{Z}_{2^a}$, and $i, j \in \mathbb{Z}_a$,

$$\begin{aligned}&e_{4^a}([1+2^{2a-1}]R_a - [\alpha_l 2^{2a-i-1}]S_a, [\hat{\alpha}_l 2^{2a-i-1}]R_a + [1-2^{2a-1}]S_a)\\ &= e_{4^a}([1+2^{2a-i+j-1}]R_a - [\alpha_l 2^{2a-i+j-1}]S_a, [\hat{\alpha}_l 2^{2a-i-1}]R_a + [1-2^{2a-i+j-1}]S_a)\\ &= e_{4^a}(R_a, S_a)\\ &= e_{4^a}(P_2, Q_2)^{3^b}.\end{aligned}$$

To prove the correctness of the extraction of α_i, we claim that Eqs. (2) and (3) are both satisfied if and only if α_i is 1 in the if-loop of $\alpha_l = 0$ or is 0 in the if-loop of $\alpha_l \neq 0$. We therefore consider these two cases.

Case1: the if-loop of $\alpha_l = 0$. Being in this loop in the i-th iteration means $\alpha = 0 \mod 2^i$. The oracle takes $(\widetilde{R_a}, \widetilde{S_a}, R_{ab}, S_{ab})$ as input where $(\widetilde{R_a}, \widetilde{S_a}) = ([1+2^{2a-1}]R_a, [2^{2a-i-1}]R_a + [1-2^{2a-1}]S_a)$. Recall $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, and $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$. For Eq. (2), we have

$$\begin{aligned}&\phi'_A(\widetilde{R_a}) - [e_1]R_{ab} - [f_1]S_{ab}\\ &= [(1+2^{2a-1})e_1]R_{ab} + [(1+2^{2a-1})f_1]S_{ab} - [e_1]R_{ab} - [f_1]S_{ab}\\ &= [2^{2a-1}e_1]R_{ab} + [2^{2a-1}f_1]S_{ab}\\ &= [-\alpha 2^{2a-1}e_2]R_{ab} + [-\alpha 2^{2a-1}f_2]S_{ab}\\ &= \mathbf{O}.\end{aligned}$$

That is, Eq. (2) will always hold. Remark the third equation comes from Lemma 3 and the fact that i is less than a. The fourth equation comes from the fact that $\alpha = 0 \mod 2^i$ and $i \geq 1$ and $\{R_{ab}, S_{ab}\}$ is a basis for $E_{AB}[4^a]$.

Also, since $\alpha_l = 0$, $\hat{\alpha}_l$ is 1 by the specification of QuasiInv. Recall $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, and $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$. For Eq. (3), we have

$$\begin{aligned}&\phi'_A(\widetilde{S_a}) - [e_2]R_{ab} - [f_2]S_{ab}\\ &= [2^{2a-i-1}e_1 - 2^{2a-1}e_2]R_{ab} + [2^{2a-i-1}f_1 - 2^{2a-1}f_2]S_{ab}\\ &= [-\alpha 2^{2a-i-1}e_2 - 2^{2a-1}e_2]R_{ab} + [-\alpha 2^{2a-i-1}f_2 - 2^{2a-1}f_2]S_{ab}\\ &= [\alpha_i 2^{2a-1} - 2^{2a-1}][e_2]R_{ab} + [\alpha_i 2^{2a-1} - 2^{2a-1}][f_2]S_{ab}.\end{aligned}$$

Similarly, the third equation comes from Lemma 3 and the fact that i is less than a. The fourth equation comes from the fact that $\alpha = 0 \mod 2^i$ and $\{R_{ab}, S_{ab}\}$ is a basis for $E_{AB}[4^a]$. By Lemma 4, e_2 and f_2 cannot be both even so that at least one of $[2^{2a-1}e_2]R_{ab}$ and $[2^{2a-1}f_2]S_{ab}$ is of order 2. Equation (3) holds if and only if α_i is 1.

Therefore, by combining conditions of Eqs. (1) to (3), in the if-loop of $\alpha_l = 0$, the oracle outputs $c = 1$ if and only if $\alpha_i = 1$.

Case2: the if-loop of $\alpha_l \neq 0$**.** The condition is equivalent to 2^j is the maximal power of 2 dividing α. The oracle takes $(\widetilde{R_a}, \widetilde{S_a}, R_{ab}, S_{ab})$ as input where $(\widetilde{R_a}, \widetilde{S_a})$ $= ([1 + 2^{2a-i+j-1}]R_a - [\alpha_l 2^{2a-i+j-1}]S_a, [\hat{\alpha}_l 2^{2a-i-1}]R_a + [1 - 2^{2a-i+j-1}]S_a)$.

Recall $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, and $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$. For Eq. (2), we have

$$\begin{aligned}&\phi'_A(\widetilde{R_a}) - [e_1]R_{ab} - [f_1]S_{ab}\\ =\ &[(2^{2a-i+j-1})e_1 - \alpha_l 2^{2a-i+j-1}e_2]R_{ab} + [(2^{2a-i+j-1})f_1 - \alpha_l 2^{2a-i+j-1}f_2]S_{ab}\end{aligned}$$

Recall that $\{R_{ab}, S_{ab}\}$ is a basis for $E_{AB}[4^a] \simeq \mathbb{Z}_{4^a} \times \mathbb{Z}_{4^a}$. By Lemma 11 (Eq. (4)), we know $\phi'_A(\widetilde{R_a}) - [e_1]R_{ab} - [f_1]S_{ab} = \mathbf{O}$ if and only if $\alpha_i 2^j = 0 \mod 2$.

Also, for Eq. (3), we have $\hat{\alpha}$

$$\begin{aligned}&\phi'_A(\widetilde{S_a}) - [e_2]R_{ab} - [f_2]S_{ab}\\ =\ &[\hat{\alpha}_l 2^{2a-i-1}e_1 + (-2^{2a-i+j-1})e_2]R_{ab} + [\hat{\alpha}_l 2^{2a-i-1}f_1 + (-2^{2a-i+j-1})f_2]S_{ab}\end{aligned}$$

Recall that $\{R_{ab}, S_{ab}\}$ is a basis for $E_{AB}[4^a] \simeq \mathbb{Z}_{4^a} \times \mathbb{Z}_{4^a}$. By Lemma 11 (Eq. (5)), we know $\phi'_A(\widetilde{S_a}) - [e_2]R_{ab} - [f_2]S_{ab} = \mathbf{O}$ if and only if $\alpha_i = 0$.

Therefore, by combining conditions of Eqs. (1) to (3), in the if-loop of $j \neq \perp$, the oracle outputs $c = 1$ if and only if $\alpha_i = 0$.

Remark 13. It seems that in our attack, both the satisfaction of Eq. (1) and the identical kernels of ϕ'_A used by the oracle the proof of Theorem 12 are derived from the fact that the kernel is of the form $\langle [2^a]P_2 + [2^a\alpha]Q_2\rangle$. Hence, one may guess that relaxing the kernel to be $\langle [2^i]P_2 + [2^i\alpha]Q_2\rangle$ for some $i \in \{0, \cdots, a-1\}$ can give a variant secure against the attack we presented. However, in the appendix, we consider a more generic situation for HealSIDH covering the concern, and the prime 2 can be replaced by any small natural number q. The algorithm takes $2a(q-1)$ oracle queries to fully recover Alice's secret key $\alpha \in \mathbb{Z}_{q^{2a}}$.

5 Summary

This work presents an adaptive attack on the isogeny-based key exchange and PKE schemes in [FP21], which were claimed to have the static-static property against any adaptive attack. Our attack is based on the subtle flaws in the main theorems (Theorems 1 and 2) in [FP21] for the key validation mechanism used in each scheme, which states that Bob can pass the key validation mechanism only if his input is correctly formed. We not only show that multiple non-trivial solutions can pass the check but also derive a concrete and efficient adaptive attack against the static-static proposals by tweaking the GPST attack. Furthermore, we provide a generalized attack in the appendix on any immediate repairs to the mechanism exploiting the commutativity of the SIDH evaluations.

Hence, our result points out that having an efficient static-static key exchange or PKE from a robust isogeny assumption remains an open problem. We look forward to future work in the community to resolve this problem.

Acknowledgement. This project is supported by the Ministry for Business, Innovation and Employment in New Zealand. We thank Shuichi Katsumata and Federico Pintore (alphabetically ordered) for pointing out errors in the previous version and helpful comments to improve clarity. Also, we thank anonymous reviewers from PQCrypto2022 for their detailed comments and suggestions.

A A Generalized Attack

This section presents a generalized result. We consider a more generic condition where Alice uses q^n torsion subgroup for some natural numbers n, q to replace 2^{2a}. Furthermore, we do not restrict the secret kernel to be of the form $\langle [q^{n/2}]P_q + [\alpha][q^{n/2}]Q_q \rangle$ where $\{P_q, Q_q\}$ is a basis of $E_0[q^n]$ and $\alpha \in \mathbb{Z}_{q^a}$. Instead, we permit α to be drawn arbitrarily from $\mathbb{Z}_{q^a}$ and the kernel to be $\langle [q^{n-a}](P_q + [\alpha]Q_q) \rangle$. When n is even and $q = 2$, taking $a = n/2$ is the case considered in Sect. 2.2. The generalization captures any straightforward modification of the HealSIDH cryptosystem. The final algorithm takes $a(q-1)$ oracle queries to fully recover Alice's secret key $\alpha \in \mathbb{Z}_{q^a}$. Therefore, as long as q is small, the HealSIDH cryptosystem and the key validation algorithm are vulnerable to our new variant of GPST attack.

To be more specific, the public parameter $\mathsf{pp} = (E_0, P_q, Q_q, P_{q'}, Q_{q'}, p, q, q')$ where $q, q' \in \mathbb{N}$ are coprime, $p = fq^n q'^{n'} - 1$ is prime, $q^n \approx q'^{n'}$, and $\{P_q, Q_q\}$ and $\{P_{q'}, Q_{q'}\}$ are bases for $E_0[q^n]$ and $E_0[q'^{n'}]$, resp. Alice samples a secret α uniformly at random from $\mathbb{Z}_{q^a}$, computes $\phi_A : E_0 \to E_A = E_0/\langle [q^{n-a}](P_q + [\alpha]Q_q) \rangle$ and representing $\phi_A(P_q) = [e_1]R_A + [f_1]S_A$ and $\phi_A(Q_q) = [e_2]R_A + [f_2]S_A$ where $\{R_A, S_A\}$ is a canonical basis for $E_A[q^a]$. Alice's secret key is $\mathsf{sk}_A = (\alpha, e_1, f_1, e_2, f_2)$ and public key is $(E_A, \phi_A(P_{q'}), \phi_A(Q_{q'}))$.

The high-level idea of the generalized attack is similar. Different from the "reciprocal" GPST attack presented in Sect. 4, one can view the generalized attack as the "triple" GPTS attack. Similarly, we use the equalities of Eq. (2) and Eq. (3) to extract the information of α and a quasi-inverse of α simultaneously. Additionally, on input $(R'_a, S'_a, R'_{ab}, S'_{ab})$, the oracle computes the isogeny with kernel $\langle R'_a + \alpha S'_a \rangle$. We will use the equality between $\langle R'_a + \alpha S'_a \rangle$ and $\langle \phi_B(P_q) + \alpha \phi_B(Q_q) \rangle$ to extract α again (see Lemma 18). We will show three equalities hold if and only if the extraction of a digit of α is correct.

Heuristic Assumption. We assume that the oracle will return 0 with an overwhelming probability if the input does not induce the same kernel as the honest input. Since we do not restrict the secret kernel to be of the form $\langle [q^{n/2}]P_q + [\alpha][q^{n/2}]Q_q \rangle$, the isogeny used by the oracle might therefore vary with each query[2]. We thereby require this assumption. Given the randomness of isogeny evaluation, the assumption is reasonable. Assume a new induced isogeny used by the oracle mapping R_a and S_a uniformly at random over $\mathbb{F}_p^2$. Then both equations (Eqs. (2) and (3)) are satisfied with probability around $1/p^2$ even if we only focus on the x-coordinate.

We start with following three simple facts similar to Proposition 2 and Lemmas 3 and 4.

Proposition 14. *If Bob honestly generates R_a, S_a, R_{ab}, S_{ab} by $R_a = \phi_B(P_q)$, $S_a = \phi_B(Q_q)$, $R_{ab} = \phi'_B(R_A)$ and $S_{ab} = \phi'_B(S_A)$, then $\{R_{ab}, S_{ab}\}$ is a basis of $E_{AB}[q^n]$ and $\{R_a, S_a\}$ is a basis of $E_B[q^n]$.*

Proof. Since $[q^n]R_a = \phi_B([q^n]P_q) = \mathbf{O}$ and $[q^n]S_a = \phi_B([q^n]Q_q) = \mathbf{O}$, both R_a and S_a are in $E_B[q^n]$. Due to $e_{q^n}(R_a, S_a) = e_{q^n}(P_q, Q_q)^{q'^{n'}}$, we know $e_{q^n}(R_a, S_a)$ is a primitive q^n-th root of unity. Similarly, Since $[q^n]R_{ab} = \phi'_B([q^n]R_A) = \mathbf{O}$ and $[q^n]S_{ab} = \phi'_B([q^n]S_A) = \mathbf{O}$, both R_{ab} and S_{ab} are in $E_{AB}[q^n]$. Due to $e_{q^n}(R_{ab}, S_{ab}) = e_{q^n}(R_A, S_A)^{q'^{n'}}$, we know $e_{q^n}(R_{ab}, S_{ab})$ is a primitive q^n-th root of unity. Therefore, the result follows.

Lemma 15. *Let e_1, e_2, f_1, f_2 defined as above and $\alpha \in \mathbb{Z}_{q^a}$ be the secret key of Alice such that $\ker(\phi_A) = \langle [q^{n-a}](P_q + [\alpha]Q_q) \rangle$. If Alice follows the protocol specification, then $e_1 + \alpha e_2 = f_1 + \alpha f_2 = 0 \mod q^a$.*

Proof. Given $\phi_A(P_2) = [e_1]R_A + [f_1]S_A$ and $\phi_A(Q_2) = [e_2]R_A + [f_2]S_A$, we have $\mathbf{O} = \phi_A([q^{n-a}](P_q + [\alpha]Q_q)) = [q^{n-a}][e_1 + \alpha e_2]R_A + [q^{n-a}][f_1 + \alpha f_2]S_A = [e_1 + \alpha e_2]R_A + [f_1 + \alpha f_2]S_A$. Recall that $\{[q^{n-a}]R_A, [q^{n-a}]S_A\}$ is a basis of $E_A[q^a]$. Therefore, $e_1 + \alpha e_2 = f_1 + \alpha f_2 = 0 \mod q^a$.

Lemma 16. *If Alice produces $\phi_A(P_q)$ and $\phi_A(Q_q)$ honestly, then e_2 and f_2 cannot be both divisible by q.*

Proof. Suppose for the purpose of contradiction that both e_2 and f_2 are divisible by q. Then, $[q^{n-1}]\phi_A(Q_q) = \mathbf{O}$, which implies $\ker(\phi_A) = \langle [q^{n-a}](P_q + [\alpha]Q_q) \rangle$ contains $[q^{n-1}]Q_q$. That is, $[kq^{n-a}]P_q + [kq^{n-a}\alpha]Q_q = [q^{2a-1}]Q_q$ for some $k \in \mathbb{Z}_{q^a}$, so $k = 0$. This contradicts the fact that $\{P_q, Q_q\}$ is a basis for $E_0[q^n]$. The result follows.

The algorithm in Fig. 3 together with Theorem 17 provides an iterative approach for recovering α. It requires $q - 1$ oracle queries to recover each digit of α in each iteration.

Theorem 17. *Assume Alice follows the protocol specification. The algorithm in Fig. 3 returns α in Alice's secret key.*

[2] For instance, on input $(R_a, [2^{a-1}]R_a + S_a, R_{ab}, S_{ab})$ as Lemma 5 for $q = 2$ and $n = a$, the isogeny used by the oracle is with kernel $\langle R_a + [\alpha]S_a + [\alpha 2^{2a-1}]R_a \rangle$. The kernel is the same if and only if α is divisible by 2.

Proof. We are going to prove the theorem by induction on i for the i-th digit of α where $i < a$. Write $-\alpha = \alpha_l + q^i\alpha_i \mod q^{i+1}$ for some $i \in \{0, \ldots, a-1\}$ where $\alpha_l \in \mathbb{Z}_{q^i}$ and $\alpha_i \in \mathbb{Z}_q$ represent the digits that have been recovered and the next digit to be recovered respectively.

First of all, we will show that within each query in each loop with respect to i, the isogeny ϕ'_A computed by the oracle is of the kernel $\langle R_a + \alpha S_a \rangle$ if $t = \alpha_i$.

Lemma 18 *(Kernel analysis)***.** *For each query made in Fig. 3 in each loop with respect to i, the kernel used by the oracle internally is identical to $\langle [q^{n-a}](P_q + [\alpha]Q_q)\rangle$ if $t = \alpha_i$.*

Proof. **Case1: the if-loop of $i = 0$.** For the queries in the if-loop of $i = 0$, if $t = \alpha_i$, we have

$$\begin{aligned} &\langle [q^{n-a}](([1 + q^{n-1}]R_a - [tq^{n-1}]S_a) + [\alpha]([\hat{\alpha}_{tl} q^{n-1}]R_a + [1 - q^{n-1}]S_a))\rangle \\ =& \langle [q^{n-a}](P_q + [\alpha]Q_q)\rangle \end{aligned}$$

Remark that here $\alpha_i = \alpha_0$ and the quasi-inverse $\hat{\alpha}_{tl} = t^{-1} \mod q$ for $t \neq 0$. Therefore, $1 + \alpha\hat{\alpha}_{tl} = 0 \mod q$ and $-\alpha_0 - \alpha = 0 \mod q$, and the second equation follows.

Case2: the if-loop of $\alpha_l = 0$. For the queries in the while-loop of $\alpha_l = 0$, we have

$$\begin{aligned} &\langle [q^{n-a}](([1 + q^{n-1}]R_a) + [\alpha]([\hat{\alpha}_{tl} q^{n-i-1}]R_a + [1 - q^{n-1}]S_a))\rangle \\ =& \langle [q^{n-a}](P_q + [\alpha]Q_q)\rangle \end{aligned}$$

Remark that being in the if-loop of $\alpha_l = 0$ implies $i \geq 1$ and $q^i \mid \alpha$. Hence, in this case the kernel computed by the oracle is always $\langle [q^{n-a}](P_q + [\alpha]Q_q)\rangle$.

Case3: the if-loop of $\alpha_l \neq 0$. For the queries in the while-loop of $\alpha_l \neq 0$, if $t = \alpha_i$, we have

$$\begin{aligned} &\langle [q^{n-a}](([1 + q^{n-i+j-1}]R_a - [(\alpha_l + tq^i)q^{n-i+j-1}]S_a) \\ &+ [\alpha]([\hat{\alpha}_{tl} q^{n-i-1}]R_a + [1 - q^{n-i+j-1}]S_a))\rangle \\ =& \langle [q^{n-a}](P_q + [\alpha]Q_q)\rangle \end{aligned}$$

Remark that we have $\hat{\alpha}_{tl}(\alpha_l + tq^i) = q^j \mod q^n$ and $-\alpha = \alpha_l + q^i\alpha_i \mod q^{i+1}$ where $i > j$. Therefore, when $t = \alpha_i$, we have $q^j + \alpha\hat{\alpha}_{tl} = 0 \mod q^{i+1}$ and $(\alpha_l + tq^i) + \alpha = 0 \mod q^{i+1}$. The second equation follows.

Similarly, we analyze the satisfaction of Eq. (1) (the Weil pairing check) for the oracle input. The following lemma shows that all oracle inputs will satisfy Eq. (1).

Lemma 19 *(Eq. (1) analysis)***.** *Each query made in Fig. 3 in each loop satisfies Eq. (1).*

Proof. Recall that we have $e_{q^a}(R_a, S_a) = e_{q^a}(P_2, Q_2)^{q'^b}$[3].

Case1: the if-loop of $i = 0$. For the queries in the if-loop of $i = 0$, we always have

$$\begin{aligned} & e_{q^a}([1+q^{n-1}]R_a - [tq^{n-1}]S_a, [\hat{\alpha}_{tl}q^{n-1}]R_a + [1-q^{n-1}]S_a) \\ =\ & e_{q^a}(R_a, S_a) \\ =\ & e_{q^a}(P_q, Q_q)^{q'^b}. \end{aligned}$$

Case2: the if-loop of $j = \perp$. For the queries in the while-loop of $j = \perp$, we always have

$$\begin{aligned} & e_{q^a}([1+q^{n-1}]R_a, [\hat{\alpha}_{tl}q^{n-i-1}]R_a + [1-q^{n-1}]S_a) \\ =\ & e_{q^a}(R_a, S_a) \\ =\ & e_{q^a}(P_q, Q_q)^{q'^b}. \end{aligned}$$

Case3: the if-loop of $j \neq \perp$. For the queries in the while-loop of $j \neq \perp$, we always have

$$\begin{aligned} & e_{q^a}(R_a, S_a)^{1-q^{2n-2i+2j-2}+\hat{\alpha}_{tl}(\alpha_l+tq^i)q^{2n-2i+j-2}} \\ =\ & e_{q^a}(R_a, S_a) \\ =\ & e_{q^a}(P_q, Q_q)^{q'^b}. \end{aligned}$$

Note that since $\hat{\alpha}_{tl}(\alpha_l + tq^i) = q^j \mod q^n$, we have

$$1 - q^{2n-2i+2j-2} + \hat{\alpha}_{tl}(\alpha_l + tq^i)q^{2n-2i+j-2} = 1 \mod q^n.$$

Therefore, all oracle queries made in Fig. 3 satisfy Eq. (1).

For the case $i = 0$ of induction, we have to show the correctness of the extraction of α_0, the least significant digit of $-\alpha$. We restrict our attention to the if-loop of the condition $i = 0$. Recall $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, and $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$. For Eq. (2) $t \in \mathbb{Z}_q$, we have

$$\begin{aligned} & \phi'_A([1+q^{n-1}]R_a - [tq^{n-1}]S_a) - [e_1]R_{ab} - [f_1]S_{ab} \\ =\ & [(1+q^{n-1})e_1 - tq^{n-1}e_2]R_{ab} + [(1+q^{n-1})f_1 - tq^{n-1}f_2]S_{ab} - [e_1]R_{ab} - [f_1]S_{ab} \\ =\ & [q^{n-1}e_1 - tq^{n-1}e_2]R_{ab} + [q^{n-1}f_1 - tq^{n-1}f_2]S_{ab} \\ =\ & [-\alpha q^{n-1}e_2 - tq^{n-1}e_2]R_{ab} + [-\alpha q^{n-1}f_2 - tq^{n-1}f_2]S_{ab} \\ =\ & [\alpha_0 q^{n-1}e_2 - tq^{n-1}e_2]R_{ab} + [\alpha_0 q^{n-1}f_2 - tq^{n-1}f_2]S_{ab} \end{aligned}$$

That is, Eq. (2) will always hold. Remark the third equation comes from Lemma 15. Therefore, the condition of Eq. (2) is satisfied if and only if $t = \alpha_0$.

[3] Since we allow to use q^a- and q'^b-isogenies here, the exponent thereby is q'^b here.

Similarly, for Eq. (3), we have

$$\begin{aligned}
&\phi'_A([\hat{\alpha}_{tl}q^{n-i-1}]R_a + [1-q^{n-1}]S_a) - [e_2]R_{ab} - [f_2]S_{ab} \\
&= [\hat{\alpha}_{tl}q^{n-1}e_1 - q^{n-1}e_2]R_{ab} + [\hat{\alpha}_{tl}q^{n-1}f_1 - q^{n-1}f_2]S_{ab} \\
&= [-\alpha\hat{\alpha}_{tl}q^{n-1}e_2 - q^{n-1}e_2]R_{ab} + [-\alpha\hat{\alpha}_{tl}q^{n-1}f_2 - q^{n-1}f_2]S_{ab} \\
&= [\alpha_0\hat{\alpha}_{tl}q^{n-1} - q^{n-1}][e_2]R_{ab} + [\alpha_0\hat{\alpha}_{tl}q^{n-1} - q^{n-1}][f_2]S_{ab}.
\end{aligned}$$

That is, Eq. (3) will always hold. Remark the third equation comes from Lemma 15. Therefore, the condition of Eq. (3) is satisfied if and only if $\alpha_0\hat{\alpha}_{tl} = 1 \mod q$. Equivalently, $t = \alpha_0$, because $t\hat{\alpha}_{tl} = 1 \mod q$. If $\alpha_0\hat{\alpha}_{tl} \neq 1 \mod q$ for all $t \in \{1, \cdots, q-1\}$, then $\alpha_0 = 0$. Therefore, by combining conditions of Eqs. (1) to (3), the extraction of α_0 is correct.

It suffices to show that given α_l the extraction of α_i, the i-th digit of $-\alpha \mod q^a$ for $i \geq 1$, is correct in each iteration of the while-loop of Fig. 3. To prove the correctness of the extraction of α_i, in either the if-loop of $\alpha_l = 0$ or the else-loop ($\alpha_l \neq 0$), we claim that Eqs. (2) and (3) are both satisfied if and only if the output of the oracle is $c = 1$ for $t \in \{1, \cdots, q-1\}$ used in the loop if and only if $\alpha_i = t$ for some $t \in \{1, \cdots, q-1\}$. We therefore consider two cases.

Case1: the if-loop of $\alpha_l = 0$. The condition is equivalent to $\alpha_l = 0$ which means $-\alpha = 0 \mod q^i$. We require the following to show the result.

Lemma 20. *Assume $\alpha_i \neq 0$. Then, both of the following two equations are true if and only if $\alpha_i = t$ for some $t \in \{1, \cdots, a-1\}$:*

$$q^{n-1}e_1 = q^{n-1}f_1 \ = 0 \mod q^n \tag{6}$$

$$\hat{\alpha}_{tl}e_1 - q^ie_2 = \hat{\alpha}_{tl}f_1 - q^if_2 \ = 0 \mod q^{i+1} \tag{7}$$

Proof. By Lemma 15, we have $q^{n-1}e_1 = -\alpha q^{n-1}e_2 \mod q^n$. Also, $q^{n-1}f_1 = -\alpha q^{n-1}f_2 \mod q^n$. The execution of this loop implies α is divisible by q. Therefore, the first equation always holds.

By Lemma 15, we have $\hat{\alpha}_{tl}e_1 - q^ie_2 = -\hat{\alpha}_{tl}\alpha e_2 - q^ie_2 \mod q^{i+1}$. Since $(\alpha_l + tq^i)\hat{\alpha}_{tl} = q^i \mod q^{i+1}$, we have $-\hat{\alpha}_{tl}\alpha e_2 - q^ie_2 = (\alpha_i - t)q^i\hat{\alpha}_{tl}e_2 \mod q^{i+1}$. Similarly, we have $\hat{\alpha}_{tl}f_1 - q^if_2 = (\alpha_i - t)q^i\hat{\alpha}_{tl}f_2 \mod q^{i+1}$. By Lemma 16 and Proposition 9, $e_2\hat{\alpha}_{tl}$ and $f_2\hat{\alpha}_{tl}$ cannot both be divisible by q. Therefore, the second equation is zero if and only if $\alpha_i = t$.

Hence, both of the following two equations are true if and only if $\alpha_i = t$.

Recall $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, and $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$. For Eq. (2), we have

$$\begin{aligned}
&\phi'_A([1+q^{n-1}]R_a) - [e_1]R_{ab} - [f_1]S_{ab} \\
&= [q^{n-1}e_1]R_{ab} + [q^{n-1}f_1]S_{ab}
\end{aligned}$$

Recall that $\{R_{ab}, S_{ab}\}$ is a basis for $E_{AB}[q^n] \simeq \mathbb{Z}_{q^n} \times \mathbb{Z}_{q^n}$. By using Lemma 20 (Eq. (6)), this condition always holds.

Also, for Eq. (3), we have

$$\begin{aligned}&\phi'_A([\hat{\alpha}_{tl}q^{n-i-1}]R_a + [1-q^{n-1}]S_a) - [e_2]R_{ab} - [f_2]S_{ab}\\ &= [\hat{\alpha}_{tl}q^{n-i-1}e_1 - q^{n-1}e_2]R_{ab} + [\hat{\alpha}_{tl}q^{n-i-1}f_1 - q^{n-1}f_2]S_{ab}\end{aligned}$$

Recall that $\{R_{ab}, S_{ab}\}$ is a basis for $E_{AB}[q^n] \simeq \mathbb{Z}_{q^n} \times \mathbb{Z}_{q^n}$. By using Lemma 20 (Eq. (7)), this condition holds if and only if $\alpha_i = t$ for some $t \in \{1, \cdots, a-1\}$.

Therefore, by combining conditions of Eqs. (1) to (3), in the if-loop of $\alpha_l = 0$, the oracle outputs $c = 1$ for $t \in \{1, q-1\}$ used in the loop if and only if $\alpha_i = t$. Moreover, if all outputs of the oracle in the loop are 0, then $\alpha_i = 0$. The extraction of α_i is correct in this case.

Case2: the if-loop of $\alpha_l \neq 0$. The condition is equivalent to q^j is the maximal power of q dividing α.

Lemma 21. *Let notation be as above. Both of the following two equations are true if and only if $\alpha_i = t$:*

$$e_1 - (\alpha_l + tq^i)e_2 = f_1 - (\alpha_l + tq^i)f_2 \ = 0 \mod q^{i-j+1} \tag{8}$$

$$\hat{\alpha}_{tl}e_1 - q^j e_2 = \hat{\alpha}_{tl}f_1 - q^j f_2 \ = 0 \mod q^{i+1} \tag{9}$$

Proof. By Lemma 15, we have $e_1 - (\alpha_l + tq^i)e_2 = -\alpha e_2 - (\alpha_l + tq^i)e_2 = (\alpha_i - t)q^i e_2 \mod q^{i-j+1}$ and $f_1 - (\alpha_l + tq^i)f_2 = -\alpha f_2 - (\alpha_l + tq^i)f_2 = (\alpha_i - t)q^i f_2 \mod q^{i-j+1}$. By Lemma 16, not both e_2 and f_2 are divisible by q. Therefore, the first equation is zero if and only if $\alpha_i = t$ or $j \geq 1$.

Similarly, by Lemma 15, we have $\hat{\alpha}_{tl}e_1 - q^j e_2 = -\alpha\hat{\alpha}_{tl}e_2 - q^j e_2 \mod q^{i+1}$. Since $(\alpha_l + tq^i)\hat{\alpha}_{tl} = q^j \mod q^{i+1}$, we have $-\alpha\hat{\alpha}_{tl}e_2 - q^j e_2 = (\alpha_i - t)q^i\hat{\alpha}_{tl}e_2 \mod q^{i+1}$. Similarly, we have $\hat{\alpha}_{tl}f_1 - q^j f_2 = (\alpha_i - t)q^i\hat{\alpha}_{tl}f_2 \mod q^{i+1}$. By Lemma 16 and Proposition 9, not both $e_2\hat{\alpha}_{tl}$ and $f_2\hat{\alpha}_{tl}$ are divisible by q. Therefore, the second equation is zero if and only if $\alpha_i = t$.

Hence, both of the following two equations are true if and only if $\alpha_i = t$.

Recall $\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$, and $\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$. For Eq. (2), we have

$$\begin{aligned}&\phi'_A([1 + q^{n-i+j-1}]R_a - [(\alpha_l + tq^i)q^{n-i+j-1}]S_a) - [e_1]R_{ab} - [f_1]S_{ab}\\ &= [(q^{n-i+j-1})e_1 - (\alpha_l + tq^i)q^{n-i+j-1}e_2]R_{ab}\\ &\quad + [(q^{n-i+j-1})f_1 - (\alpha_l + tq^i)q^{n-i+j-1}f_2]S_{ab}\end{aligned}$$

For Eq. (3), we have $\hat{\alpha}$

$$\begin{aligned}&\phi'_A([\hat{\alpha}_{tl}q^{n-i-1}]R_a + [1 - q^{n-i+j-1}]S_a) - [e_2]R_{ab} - [f_2]S_{ab}\\ &= [\hat{\alpha}_{tl}q^{n-i-1}e_1 + (-q^{n-i+j-1})e_2]R_{ab} + [\hat{\alpha}_{tl}q^{n-i-1}f_1 + (-q^{n-i+j-1})f_2]S_{ab}\end{aligned}$$

Recall that $\{R_{ab}, S_{ab}\}$ is a basis for $E_{AB}[q^n] \simeq \mathbb{Z}_{q^n} \times \mathbb{Z}_{q^n}$. By Lemma 21, we know both conditions (Eqs. (2) to (3)) hold if and only if $\alpha_i = t$.

Algorithm: $\mathsf{Recover}(\mathsf{pp}, \mathsf{sk}_B)$
Input: pp public parameter of the protocol, sk_B the secret key of Bob,
Given: an oracle $\mathcal{O}_\alpha(R_a, S_a, R_{ab}, S_{ab}; E_B, E_{AB}) \to 0/1$ returns 1 if and only if the following equations hold:

$e_{q^n}(R_a, S_a) = e_{q^n}(P_q, Q_q)$,
$\phi'_A(R_a) = [e_1]R_{ab} + [f_1]S_{ab}$,
$\phi'_A(S_a) = [e_2]R_{ab} + [f_2]S_{ab}$,

where ϕ'_A is an isogeny from E_B with kernel $\langle [q^{n-a}](P_q + [\alpha]Q_q) \rangle \in E_B$.
Ensure: Alice's secret key α

Obtain $(R_a, S_a, R_{ab}, S_{ab}) \leftarrow (\phi_B(P_q), \phi_B(Q_q), \phi'_B(R_A), \phi'_B(S_A))$ by following the protocol specification using sk_B.
Obtain a from pp.
$i = 0$
$j = \perp$
$\alpha_l = 0$
while $i < a$ **do**
  $c = 0$
  $t = q$
  for $t \in \{0, \cdots, q-1\}$ **do**
    $\hat{\alpha}_{tl} \leftarrow \mathsf{QuasiInv}(\alpha_l + tq^i, q, n)$
  if $i = 0$ **then** ▷ Extract α_0.
    while $c = 0$ or $t > 0$ **do**
      $t -= 1$
      $\widetilde{R_a}, \widetilde{S_a} \leftarrow [1 + q^{n-1}]R_a - [tq^{n-1}]S_a, [\hat{\alpha}_{tl} q^{n-1}]R_a + [1 - q^{n-1}]S_a$
      $c \leftarrow \mathcal{O}(\widetilde{R_a}, \widetilde{S_a}, R_{ab}, S_{ab})$
    $\alpha_l \leftarrow t$
    $i += 1$
    if $t \neq 0$ **then** $j \leftarrow i$ ▷ Assert q is the maximal power of q dividing α.
    Continue
  if $\alpha_l = 0$ **then** ▷ Assert $\hat{\alpha}_{tl} t = 1$ or $0 \mod q$.
    while $c = 0$ or $t > 0$ **do**
      $t -= 1$
      $\widetilde{R_a}, \widetilde{S_a} \leftarrow [1 + q^{n-1}]R_a, [\hat{\alpha}_{tl} q^{n-i-1}]R_a + [1 - q^{n-1}]S_a$
      $c \leftarrow \mathcal{O}(\widetilde{R_a}, \widetilde{S_a}, R_{ab}, S_{ab})$
    $\alpha_l \leftarrow \alpha_l + tq^i$ ▷ Assert i-th digit of $-\alpha$ is t.
    if $t \neq 0$ **then** $j \leftarrow i$ ▷ Assert q^j is the maximal power of q dividing α.
  else ▷ Assert $\hat{\alpha}_{tl}(\alpha_l + tq^i) = q^j \mod q^n$.
    while $c = 0$ or $t > 0$ **do**
      $t -= 1$
      $\widetilde{R_a} \leftarrow [1 + q^{n-i+j-1}]R_a - [(\alpha_l + tq^i)q^{n-i+j-1}]S_a$
      $\widetilde{S_a} \leftarrow [\hat{\alpha}_{tl} q^{n-i-1}]R_a + [1 - q^{n-i+j-1}]S_a$
      $c \leftarrow \mathcal{O}(\widetilde{R_a}, \widetilde{S_a}, R_{ab}, S_{ab})$
    $\alpha_l \leftarrow \alpha_l + tq^i$ ▷ Assert i-th digit of $-\alpha$ is t.
  $i += 1$
return $-\alpha_l \mod q^a$

Fig. 3. A general algorithm to recover the secret α.

Therefore, by combining conditions of Eqs. (1) to (3), in the else-loop, the oracle outputs $c = 1$ for $t \in \{1, \cdots, q-1\}$ used in the loop if and only if $\alpha_i = t$. If all outputs of the oracle in the loop is 0, then $\alpha_i = 0$. The extraction in this case is correct. Hence, the algorithm in Fig. 3 successfully extracts Alices's secret key.

References

[ACC+17] Azarderakhsh, R., et al.: Supersingular isogeny key encapsulation. Submission NIST Post-Quantum Standard. Proj. **152**, 154–155 (2017)

[AJL17] Azarderakhsh, R., Jao, D., Leonardi, C.: Post-quantum static-static key agreement using multiple protocol instances. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 45–63. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72565-9_3

[BDK+22] Beullens, W., Dobson, S., Katsumata, S., Lai, Y.F., Pintore, F.: Group signatures and more from isogenies and lattices: generic, simple, and efficient. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 95–126. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-07085-3_4

[BKV19] Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_9

[CD22] Castryck, W., Decru, T.: An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Paper 2022/975 (2022). https://eprint.iacr.org/2022/975

[CLM+18] Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03332-3_15

[dQKL+21] de Quehen, V., et al.: Improved torsion-point attacks on SIDH variants. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 432–470. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84252-9_15

[EKP20] El Kaafarani, A., Katsumata, S., Pintore, F.: Lossy CSI-FiSh: efficient signature scheme with tight reduction to decisional CSIDH-512. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 157–186. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_6

[FP21] Fouotsa, T.B., Petit, C.: SHealS and HealS: isogeny-Based PKEs from a key validation method for SIDH. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 279–307. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92068-5_10

[GPST16] Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 63–91. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_3

[JD11] Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_2

[Kup05] Kuperberg, G.: A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. SIAM J. Comput. **35**(1), 170–188 (2005)

[Leo20] Leonardi, C.: A note on the ending elliptic curve in SIDH. Cryptology ePrint Archive, Report 2020/262 (2020). https://eprint.iacr.org/2020/262

[LGd21] Lai, Y.-F., Galbraith, S.D., Delpech de Saint Guilhem, C.: Compact, efficient and UC-secure isogeny-based oblivious transfer. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 213–241. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_8

[MOT20] Moriya, T., Onuki, H., Takagi, T.: SiGamal: a supersingular isogeny-based PKE and its application to a PRF. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 551–580. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_19

[Pei20] Peikert, C.: He gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 463–492. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_16

[Sil09] Silverman, J.H.: The Arithmetic of Elliptic Curves, vol. 106. Springer, Heidelberg (2009). https://doi.org/10.1007/978-0-387-09494-6

[UJ20] Urbanik, D., Jao, D.: New techniques for SIDH-based NIKE. J. Math. Cryptol. **14**(1), 120–128 (2020)