On the Security of Isogeny Based AKE in the Quantum Random Oracle Model

Haiyang Xue[1], Man Ho Au[1], Rupeng Yang[1], Bei Liang[2], Haodong Jiang[3]

[1] Department of Computer Science, The University of Hong Kong
haiyangxc@gmail.com, allen.au@gmail.com, orbbyrp@gmail.com
[2] Yanqi Lake Beijing Institute of Mathematical Sciences and Applications
lbei@bimsa.cn
[3] State Key Laboratory of Mathematical Engineering and Advanced Computing
hdjiang13@gmail.com

Abstract. Several quantum-resistant authenticated key exchange protocols (AKEs) have been proposed from supersingular isogenies. SIAKE [Xu et al. ASIACRYPT 2019] is one of the most efficient schemes for achieving strong security. However, its security analyses are conducted in the classical random oracle model, thereby leaving security in the quantum random oracle model (QROM) as an open problem. In this paper, we prove that SIAKE is also secure in the QROM with a slight modification by adding public keys in hash functions. Our approach provides a new AKE based on communicative supersingular isogeny. Our technique could be extended to show that a modified X3LH [Xue et al. ASIACRYPT 2018], a generic AKE based on double-key PKE, is QROM secure under the one-way assumption of double-key PKE.

Keywords: supersingular isogeny, AKE, quantum random oracle model

Version Note. This is an updated version of previous eprint [43] with the title "Compact Authenticated Key Exchange in the Quantum Random Oracle Model".

1 Introduction

Since the introduction of authenticated key exchange (AKE), there has been a series of works on security models and constructions from classical assumptions. Recently, one of the most important and appealing directions is to construct AKE against quantum attacks. Among them, those based on lattice [6] and supersingular isogeny [16,24] are the most attractive ones. We focus on AKE from supersingular isogeny in this paper.

Isogeny-based AKE can be traced back to Galbraith [20], Fujioka et al. [19], and LeGrow et al. [32]. However, they either achieve weak security or require considerable overhead (e.g. [32] relies on inefficient isogeny-based signatures). The following works seek to design an efficient scheme that also provides a strong security guarantee. Protocols gave by Longa [34, sec. 4, AKE-SIDH-SIKE] and Xu et al. [45] are the most prominent ones. Both of them instantiate generic frameworks, i.e., the framework is given by Fujioka et al. [18] and that of Xue et al. [44] respectively.

Fujioka et al. [18] generically construct CK^+ secure AKE (denoted as FSXY hereafter) by relaxing their work [17] in the standard model to the random oracle model. Particularly, FSXY consists of a one-way chosen-ciphertext (OW-CCA) secure KEM under the responder's static public key and parallel execution of OW-CCA secure KEM and passively-secure KEM under initiator's static and ephemeral public keys respectively. We could build FSXY from one-way chosen-plaintext (OW-CPA) secure PKE, since OW-CCA KEM is implied by OW-CPA PKE in the random oracle model (ROM) [17, Sec. 4].

Another noteworthy framework is due by Xue et al. [44] (denoted as X3LH hereafter). Abstracted from many well-known ad-hoc AKEs, X3LH devises AKE from a new primitive

called adaptively secure double-key (2-key) KEM, which could be obtained from a passively secure 2-key PKE [44, Sec. 6.2] under the assumption of the random oracle. Their observation is that, in FSXY, two independent ciphertexts under the initiator's static and ephemeral public keys could be incorporated into single encryption under these two public keys, and this incorporation may yield communication and computation advantage. They also formalized the security requirement of the 2-key PKE as [OW-CPA, OW-CPA]. More precisely, in the [OW-CPA, $\cdot$] (resp. [$\cdot$, OW-CPA]) game, the adversary attempts to invert the ciphertext of a random message which is encrypted under a pair of public keys, where the first (resp. second) public key is generated by the challenger, while the second (resp. first) public key is chosen by the adversary.

The advantage of X3LH is that 2-key PKE may allow computational and bandwidth saving alongside the parallel use of two standard PKEs. These savings are significant in the supersingular isogeny setting. Thus, following X3LH, Xu et al.'s protocol [45] (denoted as SIAKE) is much more compact than Longa's instantiation [34] of FSXY (denoted as FSXY-SI). Looking ahead, a concrete comparison is shown in Table 1.

To better understand SIAKE and FSXY-SI, let's intuitively explain constructions of 2-key PKE using classical ElGamal. Let g be a group generators and $h_1 = g^{x_1}, h_2 = g^{x_2}$ be the public keys of two ElGamal schemes. A 2-key PKE of message $m := m_1||m_2$ can be generated in two approaches, where H is a pair-wise independent hash function.

Types	Intuitions	Schemes
Type 1	$[\ g^{r_1}, \mathsf{H}(h_1^{r_1}) \oplus m_1\ ]\ ||\ [\ g^{r_2}, \mathsf{H}(h_2^{r_2}) \oplus m_2\ ]$	FSXY-SI [18,34]
Type 2	$[\ c_1 = g^{r_1}, c_2 = \mathsf{H}(h_1^{r_1}) \oplus m_1, c_3 = \mathsf{H}(h_2^{r_1}) \oplus m_2\ ]$	SIAKE [45]

Updating mathematical structure to supersingular isogeny key exchange (SIDH) [16], SIAKE and FSXY-SI follow Type 2 and Type 1 approach respectively. It is obvious that SIAKE benefits from randomness reuse.

Nevertheless, we should carefully handle this update. Compared with classical ElGamal, isogeny-based encryption has two essential differences. 1) The validation of SIDH's public key is itself a hard problem [41]; 2) The gap assumption[4] does not hold in the SIDH setting, due to the adaptive attack [21] given by Galbraith, Petit, Shani, and Ti. SIAKE and FSXY-SI circumvent these issues by upgrading the passively-secure PKE to chosen-ciphertext secure KEM in the random oracle model.

The Quantum ROM. Since the quantum computer could execute all the off-line primitives, including hash functions, Boneh et al. [5] introduced the quantum ROM (QROM), in which the adversary can query random oracle with arbitrary superpositions. It is widely believed that proofs in the *quantum* ROM rather than *classical* ROM fulfill the security requirements against quantum adversaries.

Motivation. Although SIAKE and FSXY-SI lead to efficient AKEs from supersingular isogenies, analyses of their securities are conducted in the classical ROM, thereby leaving security in QROM as an open problem, which motivates us to investigate in this paper.

Hövelmanns et al. [23] made the first try on this problem. They proposed a modified FSXY (denoted as HKSU hereafter) and proved its QROM-security. They re-examined the puncture technique of [38] (which is recent progress on QROM security of Fujisaki-Okamoto) and applied it to FSXY. A shortcoming of their technique is that the underlying 1-key PKE should be IND-CPA secure (cf. OW-CPA secure PKE is sufficient in the original FSXY.)

Although HKSU's technique works well for FSXY-SI, it can not be applied to SIAKE, due to the following reasons.

[4] In the classical group, the gap assumption states the computational Diffie-Hellman is still hard even giving a (limited) oracle on decisional Diffie-Hellman problem.

Firstly, 2-key PKE of SIAKE only achieves one-wayness even under supersingular isogeny DDH (SI-DDH) assumption. Recall that SIAKE's 2-key ciphertext is analogous to Type 2's ciphertext. The infeasibility of IND security comes from that, in the security game, h_2 is chosen by the adversary. Specifically, assuming (c_1^*, c_2^*, c_3^*) is the challenge ciphertext, to prove IND security, we may embed DDH challenge into g, h_1, c_1^* and c_2^*. However, simulating c_3^* is challenging since neither $\log_g h_2$ nor $\log_g c_1^*$ is known. Some kind of gap assumption and random oracle may help to fix this. Unfortunately, as said before, the gap assumption does not hold for supersingular isogeny.

Secondly, HKSU's proof is built indispensably on the so-called injective mapping with encryption under a fixed public key to decouple the decapsulation-like oracle from the secret key. However, in SIAKE, the adversary may query decapsulation-like oracle under many pairs of public keys and could maliciously choose the second public key (without the awareness of any user). This highly influences the applicability of injective mapping with encryption since now the "injective" property may not hold.

Thus, to support the state-of-the-art scheme, more investigations should be conducted on the security of SIAKE in QROM, which is also an open problem raised in [45]. In this paper, we are motivated to address this.

1.1 Our Contributions

- We prove that, with a slight modification, SIAKE is secure under the SI-DDH assumption in the QROM. Actually, we give a more general result: modified SIAKE is an adaptively secure AKE in QROM if the underlying 2-key PKE is [OW-CPA, OW-CPA] secure with perfect correctness.
- This also provides an approach to design isogeny-based AKEs in QROM, i.e., we only need to focus on instantiating 2-key PKE. With this guide, CSIAKE, a new construction based on commutative supersingular isogeny DDH (CSI-DDH) [15], is given.
- By further taking decryption failure into account, we extend the result to show that modified X3LH transforms any [OW-CPA, OW-CPA] secure 2-key PKE into adaptively secure AKE in QROM. This re-answer the open problem on QROM security of FSXY. Our answer builds on the OW-CPA security of underlying PKE, as the original FSXY required, contrary to HKSU [23] which needs IND-CPA.
- Our proof technique, namely, domain separation and *injective mapping with encryption under many public keys*, is of independent interest. We believe this method is helpful for multi-user security of Fujisaki-Okamoto transformation in QROM.

See Fig. 1 and Table 1 for a summary of our contributions and a comparison with previous works.

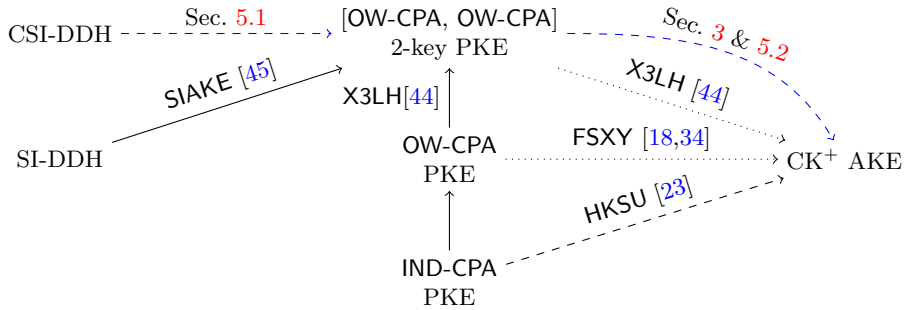

Fig. 1. Illustration of previous and our works. The dotted (resp. dashed) lines indicate works in the classical ROM (resp. QROM). The dashed blue lines indicate our work.

AKEs	Assumption	ROM	Model	Arbitrary Reg.	Bytes	Isogenies
Gal [20]	SI-CDH	c	CK	no	660	3+3
FTTY [19]	SI-DDH	c	CK	no	660	3+3
FSXY-SI [18,34]	SI-DDH	c	CK^+	yes	1384	6+6
SIAKE [45]	SI-DDH	c	CK^+	yes	1054	6+5
HKSU[23]	SI-DDH	q	CK^+	yes	1384	6+6
Ours	SI-DDH	q	CK^+	yes	1054	6+5
[28,31]	Strong-CSIDH	c	CCGJJ	yes	1320	4+4
HKSU[23]	CSI-DDH	q	CK^+	yes	2688	6+6
Ours	CSI-DDH	q	CK^+	yes	2028	6+5

Table 1. Comparison of isogeny-based AKEs. In the ROM column, "q" and "c" indicate the quantum and classical ROM respectively. "Arbitrary Reg." indicates arbitrary key registration. The communication in column "Bytes" are computed for 128 bits security with SIKEp403 [24] and CSIDHp-5280 [11]. "Isogenies" indicates the computation of isogenies and "$x + y$" means that the initiator (resp. responder) runs x (resp. y) isogenies. Strong-CSIDH is an analog of non-standard strong DH assumption for commutative supersingular isogeny.

1.2 Technique Overview

We first review SIAKE in detail and discuss challenges in proving its security in QROM. Then, our solution follows.

Review of SIAKE. SIAKE starts from an [OW-CPA, OW-CPA] secure 2-key PKE (with perfect correctness). Specifically, using the crypto-friendly notion of isogeny (refer to Sec. 2.2), $\mathfrak{g}$ is the public parameter, $(\mathfrak{g}^{\mathfrak{a}_1}, \mathfrak{a}_1)$ and $(\mathfrak{g}^{\mathfrak{a}_2}, \mathfrak{a}_2)$ are two public-secret key pairs, the ciphertext of $m_1||m_2$ with randomness $\mathfrak{b}$ is

$$\mathsf{Enc}(\mathfrak{g}^{\mathfrak{a}_1}, \mathfrak{g}^{\mathfrak{a}_2}, m_1||m_2; \mathfrak{b}) := \mathfrak{g}^{\mathfrak{b}}, \mathsf{H}((\mathfrak{g}^{\mathfrak{a}_1})^{\mathfrak{b}}) \oplus m_1, \mathsf{H}((\mathfrak{g}^{\mathfrak{a}_2})^{\mathfrak{b}}) \oplus m_2,$$

where H is a pair-wise independent hash function. It could be taken as the randomness-reuse of two standard one-key encryptions in SIKE [24]. SIAKE achieves CK^+ secure AKE via a two-step process: firstly from passively secure 2-key PKE to adaptively secure 2-key KEM, and then to AKE. Given hash functions G, h and H, SIAKE could be integrated as below, where Enc_1 is a normal one key encryption, i.e., SIKE [24].

Alice (pk_A, sk_A)		**Bob** (pk_B, sk_B)
	$pk_{2A},\ C_A = \mathsf{Enc}_1(pk_B, m_A; G(m_A))$ $\longrightarrow$	
$K_A = h(m_A)$	$\longleftarrow$ $C_B = \mathsf{Enc}(pk_A, pk_{2A}, m_B; G(m_B))$	$K_B = h(pk_{2A}, m_B)$
$K = H(sid, K_A, K_B)$		$K = H(sid, K_A, K_B)$

Recall that, a CK^+ secure AKE guarantees that no PPT adversary can distinguish the session key of test-session from a random string, even if it could send any message, make SessionKeyReveal queries on any non-test-session to obtain a session key, query SessionStateReveal to gain the internal state, and corrupt some users to get their secret keys.

Security of underlying 2-key PKE could be separated as [OW-CPA, $\cdot$] and [$\cdot$, OW-CPA] games. In the first (resp. second) game, the adversary attempts to invert the ciphertext of a

random message where the first (resp. second) public key is generated by challenger, while the second (resp. first) public key is chosen by the adversary.

Roughly speaking, in SIAKE, $[\cdot, \mathsf{OW\text{-}CPA}]$ is used to provide weak perfect forward security for AKE, and $[\mathsf{OW\text{-}CPA}, \cdot]$ security is appropriately utilized to handle all the other attacks. To fill up the gap between AKE and 2-key PKE, two issues should be resolved.

Issue I: Answer SessionKeyReveal query (which is a decapsulation-like query) without the static secret key (i.e., the first secret key of 2-key PKE).

Issue II: Decouple the session key K^* of test-session from its challenge ciphertext C^* (which is the encryption of a message m^*).

In classical ROM, hash lists could effectively handle these issues. Concretely, we could answer the SessionKeyReveal queries by searching the hash lists and could decouple the test-session key from its challenge ciphertext when the hash lists do not contain m^*. In case hash lists contain m^*, we could transfer it into solving the $[\mathsf{OW\text{-}CPA}, \cdot]$ problem.

However, maintaining hash lists is not an easy task in QROM. For 1-key PKE, several recent works [39,22,25,38,2,8,47,30] develop new techniques to circumvent recording hash lists when re-exam QROM security of Fujisaki-Okamoto, which replaces randomness with the hash of the message, i.e., $G(m)$, and sets the encapsulated key as $K = h(m)$. Unfortunately, applying them to SIAKE is challenging. In the following, we analyze the challenges and give our solution.

Challenges and Our Solution. To simplify the presentation, we only analyze the authentication of Alice. Let $C^* = \mathsf{Enc}(pk_1^*, pk_2^*, m_B^*; G(m_B))$ be the 2-key ciphertext of test-session, where pk_1^* is Alice's static key and pk_2^* (may be chosen by adversary) is the ephemeral key. Furthermore, we abuse the notion of $(\mathsf{Enc}_1, \mathsf{Dec}_1)$ as the encryption and decryption of standard PKE or SIKE [24].

***Issue I**: Answering SessionKeyReveal without Secret Key.* In Fujisaki-Okamoto, there is a similar problem of answering the decapsulation oracle without knowing the secret key. To solve it, Targhi and Unruh [39] appended a length-preserving hash of plaintext to the ciphertext. Although it could be directly applied to SIAKE, it incurs a considerable overhead which we want to avoid.

Another elegant technique is the injective mapping with encryption [5,38,25,2], which is to replace h with the composition of a private QRO h_q and an injective map f. As illustrated in Fig. 2, f is taken to be $m \mapsto \mathsf{Enc}_1(pk, m; G(m))$. By such replacement, we could directly return $h_q(C)$ as the key encapsulated in ciphertext C, since $h(\mathsf{Dec}_1(sk, C)) = h_q \circ \mathsf{Enc}_1\big(pk, \mathsf{Dec}_1(sk, C); G\left(\mathsf{Dec}_1(sk, C)\right)\big) = h_q(C)$, where pk, sk are the public and secret keys.

Note that to apply the injective mapping with encryption, the public key should be known at the beginning. However, in SIAKE, an adversary could query SessionKeyReveal on many public key pairs of 2-key encryption. This technique can not be straightly applied to the scenario of SIAKE, since under which public key pairs the function Enc is applied is unknown, needless to say, the required injective property. We could narrow the gap by embedding a public key pair into h to specify Enc's public keys, i.e., setting the encapsulated key as $K = h(pk_1, pk_2, m)$. In spit of this, the proof is still challenging, since now adversary could query h with any pk_1 and pk_2 of its choice, which still contradict with the requirement that $\mathsf{Enc}(pk_1, pk_2, m; G(m))$ should be *injective*.

Our solution. One may come up with the idea of checking the validity of public keys in $h(pk_1, pk_2, m)$. However, as said before, public key validation is itself a hard problem in SIDH [41]. Our observation is that, in AKE, SessionKeyReveal queries are applied to many but bounded public keys, i.e., those honestly generated static and ephemeral public keys. Thus, although maintaining hash lists is not an easy task in QROM, preparing lists of these public keys is feasible.

let N be the number of AKE users and l be the up-bound number of sessions between two users. Let $\{(pk_{1,i}, sk_{1,i})_{1\le i\le N}\}$ be the list of prepared static public-secret keys where the pair with index i is for user U_i, and let L_1 be that only contains public keys. Let $\{(pk^j_{2,i}, sk^j_{2,i})_{1\le i\le N, 1\le j\le Nl}\}$ be the list of ephemeral public-secret keys, where $pk^j_{2,i}$ is prepared as the ephemeral public key for U_i's j-th session, and let L_2 be that only contains public keys. Note that not all the prepared keys are used in the real game. For example, the adversary may register an invalid public key for U_i, in which case $(pk_{1,i}, sk_{1,i})$ is not used. Fortunately, we do not need to answer SessionKeyReveal queries on those sessions.

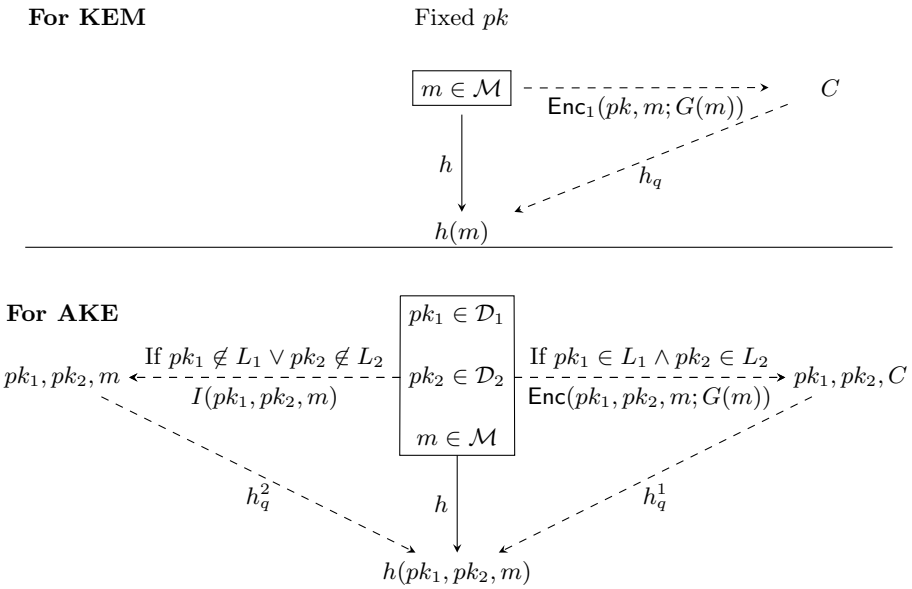

Fig. 2. The injective mapping by Enc_1 under *fixed* public key and the injective mapping by Enc under *many* public keys. pk is a fixed public key. I is the identity map.

With prepared L_1, L_2, domain of h, i.e., $\mathcal{D}_1 \times \mathcal{D}_2 \times \mathcal{M}$ could be divided as $L_1 \times L_2 \times \mathcal{M}$ and its complement. With such a domain separation, our technique, i.e., injective mapping with encryption under many public keys, is illustrated in Fig. 2. Namely, $h(pk_1, pk_2, m)$ is defined according to the domain separation as

$$h(pk_1, pk_2, m) = \begin{cases} h^1_q(pk_1, pk_2, \mathsf{Enc}(pk_1, pk_2, m; G(m))) & \text{if } \ pk_1 \times pk_2 \in L_1 \times L_2 \\ h^2_q(pk_1, pk_2, m) & \text{otherwise,} \end{cases}$$

where h^1_q, h^2_q are private random oracles. Now, we could answer SessionKeyReveal queries on $(pk_1 \in L_1, pk_2 \in L_2, C, \cdots)$ by using $h^1_q(pk_1, pk_2, C)$ as the key encapsulated in C, obviously, without the knowledge of secret key.

***Issue II**: Decoupling Session Key from Challenge Ciphertext.* Further investigation should be done on this issue, since 1) we start from one-way security of 2-key PKE; 2) an adversary could choose the second public key of C^*, i.e., pk^*_2.

The One Way to Hiding (OW2H) lemma [40] plays an essential role to decouple encapsulated key from challenge ciphertext. Informally, the OW2H lemma states that: if a quantum distinguisher, issuing queries to QRO $\mathcal{O}_1$ or $\mathcal{O}_2$ which only differ on a set S, could distinguish them from each other, then there exists a one-wayness attacker to find some element in S.

In the QROM evaluation of Fujisaki-Okamoto, the puncture technique, combining OW2H on a set S containing a single point, is usually applied to decouple key from challenge ciphertext (e.g. [38] and HKSU [23]). The puncture technique requires the underlying encryption to be IND-CPA, which SIAKE does not provide. Furthermore, in the scenario of SIAKE, S could not be a set of single point since the adversary could arbitrarily choose pk_2^*.

We handle this by using the unified oracle trick and a well-chosen S.

The Unified Oracle Trick. Introduced in [39] and later utilized by [25], the unified oracle trick, treats queries to G, h as a unified $G \times h$, provides a possibility to reduce security to the one-wayness of underlying encryption. We adopt this technique and replace $G(m_B)$ in SIAKE with $G(pk_{2A}, m_B)$. Looking ahead, after guessing the test-session, say pk_1^*, any query (pk_2, m) to G could be handled as a query with (pk_1^*, pk_2, m) which makes G and h share the same domain.

The Choice of S. In Fujisaki-Okamoto transformation, S (the set of different elements between $\mathcal{O}_1$ and $\mathcal{O}_2$) is a set of a single point, i.e., the challenge message. In the scenario of SIAKE, since ephemeral public key pk_2^* may be generated by the adversary, S should be carefully chosen to make sure that 1) it is large enough such that pk_2^* is covered; 2) it is not too large such that answering SessionKeyReveal query in Issue I is not affected.

Let $L_{2\mathsf{after}} \subset L_2$ be the list of ephemeral public keys utilized after the test-session. We could do this by doubling the size of L_2. Then, we set $S = \{pk_1^*\} \times \mathcal{D}_2 \setminus L_{2\mathsf{after}} \times \{m_B^*\}$, which is exactly the set satisfying all requirements. 1) If the ephemeral public key has high entropy, it holds that $pk_2^* \in \mathcal{D}_2 \setminus L_{2\mathsf{after}}$ with overwhelming probability, which satisfies the first requirement. 2) Since m_B^* is randomly chosen by simulator, any SessionKeyReveal query before the test-session meets m_B^* with negligible probability. Furthermore, by the definition of $L_{2\mathsf{after}}$, ephemeral public keys of SessionKeyReveal queries after the test-session will be in $L_{2\mathsf{after}}$ (and L_2). Thus, the second requirement is satisfied.

Putting All Together. Finally, we modify SIAKE by embedding both static and ephemeral public keys into h and adding ephemeral public key into G. The cost of this modification is negligible, while the gain is QROM security.

1.3 Extending and Generalization.

Following the approach of SIAKE and our modification, we propose a new QROM secure AKE from commutative supersingular isogeny Diffie-Hellman (CSIDH) [15]. Actually, we only need to instantiate 2-key PKE based on CSIDH. Compared with HKSU, our scheme has a significant bandwidth advantage, since a 2-key PKE ciphertext in this setting is only 5.3% longer than an ordinary PKE ciphertext.

Note that the above analysis works for any [OW-CPA, OW-CPA] secure 2-key PKE with perfect correctness. The result could be generalized to the QROM security of X3LH by further considering decryption failure.

1.4 Related Works

AKEs from Supersingular Isogeny. Galbraith [20], Fujioka et al. [19] directly construct AKEs from supersingular isogeny. Their schemes only achieve weak security (e.g., CK, Honest key registration). Longa [34] showed how to adapt FSXY (which achieves CK^+ security with arbitrary key registration) to supersingular isogeny. Xu et al. [45] proposed a more efficient construction, SIAKE, by following X3LH [44]. Very recently, De Kock et al. [28] and Kawashima et al. [31] proposed two AKEs from commutative supersingular isogenies (CSI). They focus on tight reduction but rely on a non-standard strong Diffie-Hellman assumption in the CSI set. All these works conduct their security in the classical ROM. (Fujioka et al.'s scheme [19] is the only exception, however, its proof is insufficient.)

HKSU in QROM. Hövelmanns et al. [23] proposed a modular HKSU framework (a modification of FSXY) from IND-CPA secure PKE in QROM. Based on (commutative) supersingular isogeny, our scheme has better computation and communication performance than HKSU (refer to Table 1). One may think [OW-CPA, OW-CPA] secure 2-key PKE is not a standard primitive, such as IND-CPA secure PKE. However, 2-key PKE could be built from standard primitives and assumptions (e.g. DDH in supersingular isogeny setting). By instantiating our modified X3LH with two parallel PKEs, we re-answer the QROM security of FSXY. We note that our solution needs one more re-encryption than HKSU, while we start from the same security as FSXY, i.e., OW-CPA.

Fujisaki-Okamoto in QROM. Several works [22,25,38,2,8,47,30] have re-examined Fujisaki-Okamoto transformation in QROM. They utilized injective mapping with encryption or additional hash to avoid recording hash queries and also proposed different variants of the OW2H lemma. Please refer to Appendix A for more details. Zhandry [47] showed a possibility for lazy sampling and recording queries. However, as he said, his proof might be looser than those using OW2H.

Hashing with Public Key. The technique of hashing with the public key has been used to analyze the multi-user security of Schnorr signature [3]. Recently, several submissions for the NIST Post-Quantum Cryptography Standardization (e.g., Kyber [6]) have also employed such a technique. Kyber proposed heuristic analysis from the perspective of multi-target attacks. The necessity of putting the public key into hashing is still heavily debated [10]. Our analysis in this work shows that hashing with the public key seems necessary to prove the multi-user security of Fujisaki-Okamoto in QROM.

2 Preliminary

2.1 2-Key PKE

Recall [44] that 2-key PKE 2PKE is a quadruple of PPT algorithms $(\mathsf{KGen1}, \mathsf{KGen2}, \mathsf{Enc}, \mathsf{Dec})$ with public key space $\mathcal{D}_{pk_1} \times \mathcal{D}_{pk_2}$, plaintext space $\mathcal{M}$, randomness space $\mathcal{R}$ and ciphertext space $\mathcal{C}$. Let $\mathcal{D}_1$ (resp. $\mathcal{D}_2$) be some superset of $\mathcal{D}_{pk_1}$ (resp. $\mathcal{D}_{pk_2}$) such that the membership problem of $\mathcal{D}_1$ (resp. $\mathcal{D}_2$)is easy.

- $\mathsf{KGen1}$: on input security parameter, output public-secret key (pk_1, sk_1).
- $\mathsf{KGen2}$: on input security parameter, output public-secret key (pk_2, sk_2).
- $\mathsf{Enc}(pk_1, pk_2, m; r)$: on input public keys pk_1, pk_2, plaintext $m \in \mathcal{M}$, and randomness $r \in \mathcal{R}$, output the ciphertext $C \in \mathcal{C}$. Sometimes, we eliminate the randomness r and denote it as $\mathsf{Enc}(pk_1, pk_2, m)$ for simplicity.
- $\mathsf{Dec}(sk_1, sk_2, C)$: on input sk_1, sk_2 and cipheretext C, output a plaintext m.

Correctness and Decryption failure. The decryption failure is defined as

$$\delta := \mathrm{E}\left(\max_{m \in \mathcal{M}} \Pr\left[\mathsf{Dec}\left(sk_1, sk_2, \mathsf{Enc}(pk_1, pk_2, m)\right) \neq m\right]\right),$$

where the expectation is taken over $(pk_1, sk_1) \leftarrow \mathsf{KGen1}$ and $(pk_2, sk_2) \leftarrow \mathsf{KGen2}$. The scheme is said to be perfectly correct, if $\delta = 0$.

Entropy of second public key. For any $pk_2' \in \mathcal{D}_2$, $\Pr[pk_2 = pk_2' | (pk_2, sk_2) \leftarrow \mathsf{KGen2}] \leq \varepsilon_{\mathsf{pk2}}$.

One-way Security. [OW-CPA, OW-CPA] security [44] is defined against two adversaries, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ attacking pk_1 and $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ attacking pk_2. The $[\mathsf{OW\text{-}CPA}, \cdot]$ and $[\cdot, \mathsf{OW\text{-}CPA}]$ games are shown in Fig. 3. The advantage $\mathcal{A}$ wins in game $[\mathsf{OW\text{-}CPA}, \cdot]$ is defined as $\mathrm{Adv}^{[\mathsf{OW\text{-}CPA}, \cdot]}_{\mathsf{2PKE}}(\mathcal{A}) = \Pr\left[[\mathsf{OW\text{-}CPA}, \cdot]^{\mathcal{A}} \Rightarrow 1\right]$. We say 2PKE is $[\mathsf{OW\text{-}CPA}, \cdot]$ secure, if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}^{[\mathsf{OW\text{-}CPA}, \cdot]}_{\mathsf{2PKE}}(\mathcal{A})$ is negligible. The advantage $\mathrm{Adv}^{[\cdot, \mathsf{OW\text{-}CPA}]}_{\mathsf{2PKE}}(\mathcal{B})$

Game $[\mathsf{OW\text{-}CPA}, \cdot]$	Game $[\cdot, \mathsf{OW\text{-}CPA}]$
$(pk_1, sk_1) \leftarrow \mathsf{KGen1}$	$(pk_2, sk_2) \leftarrow \mathsf{KGen2}$
$(state; pk_2^*) \leftarrow \mathcal{A}_1(pk_1)$	$(state; pk_1^*) \leftarrow \mathcal{B}_1(pk_2)$
$m \leftarrow \mathcal{M}, c^* \leftarrow \mathsf{Enc}(pk_1, pk_2^*, m)$	$m \leftarrow \mathcal{M}, c^* \leftarrow \mathsf{Enc}(pk_1^*, pk_2, m)$
$m' \leftarrow \mathcal{A}_2(state, c^*)$	$m' \leftarrow \mathcal{B}_2(state, c^*)$
return $m' \stackrel{?}{=} m$	return $m' \stackrel{?}{=} m$

Fig. 3. The one-way security games for 2-key PKE.

and $[\cdot, \mathsf{OW\text{-}CPA}]$ security can be defined in the same manner. 2PKE is $[\mathsf{OW\text{-}CPA}, \mathsf{OW\text{-}CPA}]$ secure if it is both both $[\mathsf{OW\text{-}CPA}, \cdot]$ and $[\cdot, \mathsf{OW\text{-}CPA}]$ secure.

1-key PKE. Let $\mathsf{PKE} = (\mathsf{KGen1}, \mathsf{Enc}_1, \mathsf{Dec}_1)$ be a 1-key PKE with randomness space $\mathcal{R}_1$, message space $\mathcal{M}_1$ and ciphertext space $\mathcal{C}_1$. It can be taken as a special 2-key PKE where KGen2 does nothing (e.g. $(-,-) \leftarrow \mathsf{KGen2}$).

2.2 Supersingular Isogeny with crypto-friendly notion and isogeny-based 2-key PKE

We briefly recall the SIDH protocol [16] with crypto-friendly notions from [19], and present the isogeny-based 2-key PKE of [45].

Let p be a large prime with a form $p = \ell_1^{e_1}\ell_2^{e_2} \cdot f \pm 1$, where ℓ_1 and ℓ_2 are two small primes, and f is an integer cofactor. Then we can construct a supersingular elliptic curve E_0 defined over $\mathbb{F}_{p^2}$ with order $|E_0(\mathbb{F}_{p^2})| = (\ell_1^{e_1}\ell_2^{e_2} \cdot f)^2$. Let $\mathbb{Z}_m$ be the ring of residue classes modulo m. The subgroup $E_0[m]$ of m-torsion points is isomorphic to $\mathbb{Z}_m \times \mathbb{Z}_m$ for $m \in \{\ell_1^{e_1}, \ell_2^{e_2}\}$. Let P_1, Q_1 be two points that generate $E_0[\ell_1^{e_1}]$ and P_2, Q_2 be two points that generate $E_0[\ell_2^{e_2}]$.

We present the crypto-friendly notions following the treatment of Fujioka et al. [19]. We assume $\{t, s\} = \{1, 2\}$, and denote the public parameters by $\mathfrak{g} = (E_0; P_1, Q_1, P_2, Q_2)$ and $\mathfrak{e} = (\ell_1, \ell_2, e_1, e_2)$. We define the sets of supersingular curves and those with an auxiliary basis as

$$\begin{aligned}
\mathrm{SSEC}_p &= \{\text{supersingular elliptic curves } E \text{ over } \mathbb{F}_{p^2} \text{ with } E(\mathbb{F}_{p^2}) \simeq (\mathbb{Z}_{\ell_1^{e_1}\ell_2^{e_2} f})^2\};\\
\mathrm{SSEC}_A &= \{(E; P'_t, Q'_t) | E \in \mathrm{SSEC}_p, (P'_t, Q'_t) \text{ is basis of } E[\ell_t^{e_t}]\};\\
\mathrm{SSEC}_B &= \{(E; P'_s, Q'_s) | E \in \mathrm{SSEC}_p, (P'_s, Q'_s) \text{ is basis of } E[\ell_s^{e_s}]\}.
\end{aligned}$$

Let $\mathfrak{a} = k_a$ be element from $\mathbb{Z}_{\ell_s^{e_s}}$, and $\mathfrak{b} = k_b$ be element from $\mathbb{Z}_{\ell_t^{e_t}}$. Define isogenies $\phi_A : E_0 \to E_A = E_0/\langle P_s + [k_a]Q_s\rangle$, $\phi_B : E_0 \to E_B = E_0/\langle P_t + [k_b]Q_t\rangle$, $\phi_{BA} : E_B \to E_{BA} = E_B/\langle \phi_B(P_s) + [k_a]\phi_B(Q_s)\rangle$, and $\phi_{AB} : E_A \to E_{AB} = E_A/\langle \phi_A(P_t) + [k_b]\phi_A(Q_t)\rangle$. Then, we define

$$\begin{aligned}
\mathfrak{g}^{\mathfrak{a}} &= (E_A; \phi_A(P_t), \phi_A(Q_t)) \in \mathrm{SSEC}_A, (\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}} = j(E_{BA}),\\
\mathfrak{g}^{\mathfrak{b}} &= (E_B; \phi_B(P_s), \phi_B(Q_s)) \in \mathrm{SSEC}_B, (\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}} = j(E_{AB}).
\end{aligned}$$

With notions defined above, SIDH key exchange is: Alice computes $\mathfrak{g}^{\mathfrak{a}}$ with random $\mathfrak{a}$, while Bob computes $\mathfrak{g}^{\mathfrak{b}}$ with random $\mathfrak{b}$. They could compute the shared key as $j = (\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}} = (\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}}$.

Definition 1 (SI-DDH). *SI-DDH assumption says given $\mathfrak{e}$ and $(\mathfrak{g}, \mathfrak{g}^{\mathfrak{a}}, \mathfrak{g}^{\mathfrak{b}})$, any PPT adversary could not efficiently distinguish $(\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}}$ from a random j-invariant.*

Isogeny-based (2-key) PKE. Let H be a pair-wise independent hash function. Fig. 4 presents the $[\mathsf{OW\text{-}CPA}, \mathsf{OW\text{-}CPA}]$ secure 2-key PKE $\mathsf{2PKE}_{\mathsf{sidh}}$ [45] and $\mathsf{OW\text{-}CPA}$ secure PKE from [24] under SI-DDH assumption. Note that the two encryption schemes are both perfectly correct. The entropy of public key is $\varepsilon_{\mathsf{pk2}} = 1/\ell_s^{e_s}$.

$\mathsf{KGen1}$	$\mathsf{Enc}_1(pk_1, m_1)$	$\mathsf{Dec}_1(sk_1, C)$
$\mathfrak{a}_1 \leftarrow \mathbb{Z}_{\ell_s^{e_s}}$ $pk_1 = \mathfrak{g}^{\mathfrak{a}_1}$, $sk_1 = \mathfrak{a}_1$	$\mathfrak{b} \leftarrow \mathbb{Z}_{\ell_t^{e_t}}$, $c_1 = \mathfrak{g}^{\mathfrak{b}}$ $c_2 = \mathsf{H}((\mathfrak{g}^{\mathfrak{a}_1})^{\mathfrak{b}}) \oplus m_1$	$(c_1, c_2) \leftarrow C$ $m_1 = c_2 \oplus \mathsf{H}(c_1^{\mathfrak{a}_1})$
$\mathsf{KGen2}$	$\mathsf{Enc}(pk_1, pk_2, m_1 \Vert m_2)$	$\mathsf{Dec}(sk_1, sk_2, C)$
$\mathfrak{a}_2 \leftarrow \mathbb{Z}_{\ell_s^{e_s}}$ $pk_2 = \mathfrak{g}^{\mathfrak{a}_2}$, $sk_2 = \mathfrak{a}_2$	$\mathfrak{b} \leftarrow \mathbb{Z}_{\ell_t^{e_t}}$, $c_1 = \mathfrak{g}^{\mathfrak{b}}$ $c_2 = \mathsf{H}((\mathfrak{g}^{\mathfrak{a}_1})^{\mathfrak{b}}) \oplus m_1$ $c_3 = \mathsf{H}((\mathfrak{g}^{\mathfrak{a}_2})^{\mathfrak{b}}) \oplus m_2$	$(c_1, c_2, c_3) \leftarrow C$ $m_1 = c_2 \oplus \mathsf{H}(c_1^{\mathfrak{a}_1})$ $m_2 = c_3 \oplus \mathsf{H}(c_1^{\mathfrak{a}_2})$

Fig. 4. Isogeny-based PKE $\mathsf{PKE} = (\mathsf{KGen1}, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and isogeny-based 2-key PKE scheme $\mathsf{2PKE}_{\mathsf{sidh}} = (\mathsf{KGen1}, \mathsf{KGen2}, \mathsf{Enc}, \mathsf{Dec})$.

2.3 CK$^+$ Security Model

Here, we recall the CK$^+$ model introduced by [17,18], which is a modified CK model [13] integrated with the weak perfect forward security (wPFS), resistant to key compromise impersonation (KCI) and (maximal exposure (MEX) attacks. We focus on the two-pass protocol in this definition. Please refer to Appendix B for a discussion on security models.

U_i denotes a party indexed by i, which is modeled as probabilistic polynomial time (PPT) interactive Turing machines. We assume that each party U_i owns a static pair of secret-public keys (ssk_i, spk_i), where spk_i is linked to U_i's identity such that the other parties can verify the authentic binding between them. We do not require the well-formedness of the static public key, in particular, a corrupted party can adaptively register any static public key of its choice.

Session. Each party can be activated to run an instance called a *session*. A party can be activated to initiate the session by an incoming message of the form $(\Pi, \mathcal{I}, U_A, U_B)$ or respond to an incoming message of the form $(\Pi, \mathcal{R}, U_B, U_A, X_A)$, where Π is a protocol identifier, $\mathcal{I}$ and $\mathcal{R}$ are role identifiers corresponding to *initiator* and *responder*, and X_A is the communication message. Activated with $(\Pi, \mathcal{I}, U_A, U_B)$, U_A is called the session *initiator*. Activated with $(\Pi, \mathcal{R}, U_B, U_A, X_A)$, U_B who will responds with X_B is the session *responder*.

According to the specification of AKE, the party creates session specified randomness which is generally called *ephemeral secret key*, computes and maintains a *session state*, generates outgoing messages, and completes the session by outputting a session key and erasing the session state. Here we require that the session state at least contains the ephemeral secret key.

A session may also be aborted without generating a session key. The initiator U_A creates a session state and outputs X_A, and may receive an incoming message of the forms $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ from the responder U_B, and may compute the session key SK. On the contrary, the responder U_B outputs X_B, and may compute the session key SK. We state that a session is *completed* if its owner computes the session key.

A session is associated with its owner, a peer, and a session identifier. If U_A is the initiator, the session identifier is $\mathsf{sid} = (\Pi, \mathcal{I}, U_A, U_B, X_A)$ or $\mathsf{sid} = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$, which denotes U_A as an owner and U_B as a peer. If U_B is the responder, the session is identified by $\mathsf{sid} = (\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$, which denotes U_B as an owner and U_A as a peer. The *matching session* of $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ is $(\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$ and vice versa.

Adversary. Adversary $\mathcal{A}$ is modeled as following to capture real attacks, including the control of communication and access to some secret information.

– Send(message): $\mathcal{A}$ sends messages in one of the following forms: $(\Pi, \mathcal{I}, U_A, U_B)$, $(\Pi, \mathcal{R}, U_B, U_A, X_A)$, or $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$, and obtains the response.

– $\mathsf{SessionKeyReveal}(\mathsf{sid})$: if the session sid is completed, $\mathcal{A}$ obtains the session key SK for sid.
– $\mathsf{SessionStateReveal}(\mathsf{sid})$: $\mathcal{A}$ obtains the session state of the owner of sid if the session is not completed. The session state should be specified by the concrete protocols. We require it returns the ephemeral secret keys and some intermediate computation results except for immediately erased information.
– $\mathsf{Corrupt}(U_i)$: this query allows the adversary to learn the static secret key of user U_i. After this query, U_i is said to be corrupted.

Freshness. Let $\mathsf{sid}^* = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ or $(\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$ be a completed session between U_A and U_B. If the matching session of sid^* exists, denote it by $\overline{\mathsf{sid}^*}$. We say session sid^* is fresh if $\mathcal{A}$ does not query: 1) $\mathsf{SessionStateReveal}(\mathsf{sid}^*)$, $\mathsf{SessionKeyReveal}(\mathsf{sid}^*)$, $\mathsf{SessionStateReveal}(\overline{\mathsf{sid}^*})$, or $\mathsf{SessionKeyReveal}(\overline{\mathsf{sid}^*})$ when $\overline{\mathsf{sid}^*}$ exists; 2) $\mathsf{SessionStateReveal}(\mathsf{sid}^*)$ or $\mathsf{SessionKeyReveal}(\mathsf{sid}^*)$ when $\overline{\mathsf{sid}^*}$ does not exist.

Security Experiment. (Quantum) adversary $\mathcal{A}$ could make a sequence of queries described above. During the experiment, $\mathcal{A}$ makes the query of $\mathsf{Test}(\mathsf{sid}^*)$, where sid^* must be a fresh session. $\mathsf{Test}(\mathsf{sid}^*)$ selects a random bit $b \in \{0, 1\}$, and returns the session key held by sid^* if $b = 0$; and returns a random key if $b = 1$. The experiment continues until $\mathcal{A}$ returns b'. The advantage of adversary $\mathcal{A}$ is defined as $\mathsf{Adv}^{ck+}_{\Pi}(\mathcal{A}) = \Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]$.

Definition 2. *We state that a AKE protocol Π is secure in the CK^+ model if the following conditions hold:*
Correctness*: if two honest parties complete matching sessions, then they both compute the same session key except with negligible probability.*
Soundness*: for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{ck+}_{\Pi}(\mathcal{A})$ is negligible for any one of the cases listed in the following and Table 2. Note that in these cases except 5, when it is allowed, the ephemeral secret key or static secret key of the owner of sid^* or $\overline{\mathsf{sid}^*}$ is given to $\mathcal{A}$ directly once it is determined. For case 5, the leakage of static secret key happens after the test session ends.*

1. *the static secret key of the owner of sid^* is given to $\mathcal{A}$, if $\overline{\mathsf{sid}^*}$ does not exist.*
2. *the ephemeral secret key of owner of sid^* is given to $\mathcal{A}$, if $\overline{\mathsf{sid}^*}$ does not exist.*
3. *the static secret key of the owner of sid^* and the ephemeral secret key of $\overline{\mathsf{sid}^*}$ are given to $\mathcal{A}$, if $\overline{\mathsf{sid}^*}$ exists.*
4. *the ephemeral secret key of sid^* and the ephemeral secret key of $\overline{\mathsf{sid}^*}$ are given to $\mathcal{A}$, if $\overline{\mathsf{sid}^*}$ exists.*
5. *the static secret key of the owner of sid^* and the static secret key of the peer of sid^* are given to $\mathcal{A}$, if $\overline{\mathsf{sid}^*}$ exists.*
6. *the ephemeral secret key of sid^* and the static secret key of the peer of sid^* are given to $\mathcal{A}$, if $\overline{\mathsf{sid}^*}$ exists.*

2.4 The Quantum Random Oracle Model

Boneh et al. [5] introduced the quantum random oracle (QRO) model. Zhandary [46] proved that any $2q$-wise independent random function can be used to simulate the QRO allowing at most q queries. The one way-to-hiding (OW2H) lemma, initially proposed by Unruh [40], is a useful tool for security analysis in QROM. Recently, Ambainis et al. [2] introduced the semi-classical OW2H, which is very generic and flexible. The OW2H lemma is revisited in Theorem 3 of [2] (say as revisited OW2H lemma) and it is implied by semi-classical OW2H. Such revisited OW2H lemma is more suitable for this work. The difference is that [2] considers the query depth d, while we use the number of queries q.

Event	Case	sid^* owner	$\overline{\mathsf{sid}^*}$	ssk_A	esk_A	esk_B	ssk_B	Security
E_1	1	U_A	No	$\surd$	$\times$	-	$\times$	KCI
E_2	2	U_A	No	$\times$	$\surd$	-	$\times$	MEX
E_3	2	U_B	No	$\times$	-	$\surd$	$\times$	MEX
E_4	1	U_B	No	$\times$	-	$\times$	$\surd$	KCI
E_5	5	U_A or U_B	exists	$\surd$	$\times$	$\times$	$\surd$	wPFS
E_6	4	U_A or U_B	exists	$\times$	$\surd$	$\surd$	$\times$	MEX
$E_{7\text{-}1}$	3	U_A	exists	$\surd$	$\times$	$\surd$	$\times$	MEX
$E_{7\text{-}2}$	3	U_B	exists	$\times$	$\surd$	$\times$	$\surd$	MEX
$E_{8\text{-}1}$	6	U_A	exists	$\times$	$\surd$	$\times$	$\surd$	MEX
$E_{8\text{-}2}$	6	U_B	exists	$\surd$	$\times$	$\surd$	$\times$	MEX

Table 2. The cases of AKE adversary in Definition 2. Column $\overline{\mathsf{sid}^*}$ indicates the matching session of sid^* exists or not. ssk_A(resp. ssk_B) means the static secret key of U_A(resp. U_B). esk_A(resp. esk_B) is the ephemeral secret key of U_A(resp. U_B) in sid^* or $\overline{\mathsf{sid}^*}$. "$\surd$" means the secret key is revealed to adversary, while "$\times$" means not.

Lemma 1 (OW2H, Probabilities [2]). *Let $S \subseteq X$ be random. Let $\mathcal{O}_1, \mathcal{O}_2 : X \to Y$ be random functions satisfying $\forall x \notin S, \mathcal{O}_1(x) = \mathcal{O}_2(x)$. Let z be a random bitstring. $(S, \mathcal{O}_1, \mathcal{O}_2, z$ may have arbitrary joint distribution.) Let U_A be an oracle algorithm with query number q. Let $B^{\mathcal{O}_1}$ be an oracle algorithm that on input z does the following: pick $i \leftarrow 1, \cdots, q$, run $A^{\mathcal{O}_1}(z)$ until (just before) the i-th query, measure all query input registers in the computational basis, and output the set T of measurement outcomes. Let*

$$P_{\mathsf{left}} := \Pr[b = 1 : b \leftarrow A^{\mathcal{O}_1}(z)], P_{\mathsf{right}} := \Pr[b = 1 : b \leftarrow A^{\mathcal{O}_2}(z)],$$
$$P_{\mathsf{guess}} := \Pr[S \cap T \neq \emptyset : T \leftarrow B^{\mathcal{O}_1}(z)].$$

Then we have $|P_{\mathsf{left}} - P_{\mathsf{right}}| \leq 2q\sqrt{P_{\mathsf{guess}}}$ and $|\sqrt{P_{\mathsf{left}}} - \sqrt{P_{\mathsf{right}}}| \leq 2q\sqrt{P_{\mathsf{guess}}}$.

Lemma 2 ([38]). *Let $H : \{0,1\}^l \times \mathcal{X} \to \mathcal{Y}$ and $H' : \mathcal{X} \to \mathcal{Y}$ be two independent random oracles, where l is an integer. For any unbounded time quantum adversary $\mathcal{A}$ with at most q_H queries to H, we have*

$$\left|\Pr[\mathcal{A}^{H,H(s,\cdot)}() \to 1 | s \leftarrow \{0,1\}^l] - \Pr[\mathcal{A}^{H,H'}() \to 1]\right| \leq q_H \cdot 2^{\frac{-l+1}{2}}.$$

3 Authenticated Key Exchange from Isogeny in QROM

In this section, we propose a modified SIAKE from supersingular isogeny, which is secure in QROM.

Let $\mathsf{2PKE}_{\mathsf{sidh}} = (\mathsf{KGen1}, \mathsf{KGen2}, \mathsf{Enc}, \mathsf{Dec})$ be the 2-key PKE based on SI-DDH, as Fig. 4, with public key space $\mathcal{D}_{pk_1}$ and $\mathcal{D}_{pk_2}$, randomness space $\mathcal{R} = \{0,1\}^r$, message space $\mathcal{M} = \{0,1\}^n$ and ciphertext space $\mathcal{C}$. Let $\mathsf{PKE} = (\mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ be the PKE as Fig. 4, with randomness space $\mathcal{R}_1 = \{0,1\}^{r_1}$, message space $\{0,1\}^{n_1}$ and ciphertext space $\mathcal{C}_1$.

Let $\mathcal{D}_1$ (resp. $\mathcal{D}_2$) be a superset of $\mathcal{D}_{pk_1}$ (resp. $\mathcal{D}_{pk_2}$) such that its set membership problem is easy. Let $\mathcal{U}$ be the space of user's id. Define the following hash functions,

$$\begin{aligned}
&f : \{0,1\}^{2n} \to \{0,1\}^n, && f_1 : \{0,1\}^{2n} \to \{0,1\}^{n_1},\\
&G : \mathcal{D}_2 \times \{0,1\}^n \to \{0,1\}^r, && G_1 : \mathcal{D}_1 \times \{0,1\}^{n_1} \to \{0,1\}^{r_1}\\
&h : \mathcal{D}_1 \times \mathcal{D}_2 \times \{0,1\}^n \to \{0,1\}^n, && h_1 : \mathcal{D}_1 \times \{0,1\}^{n_1} \to \{0,1\}^n,\\
&h' : \mathcal{D}_1 \times \mathcal{D}_2 \times \{0,1\}^n \times \mathcal{C} \to \{0,1\}^n, && h'_1 : \mathcal{D}_1 \times \{0,1\}^n \times \mathcal{C}_1 \to \{0,1\}^n,\\
&H : \{0,1\}^{2n} \times \mathcal{C}_1 \times \mathcal{C} \times \mathcal{U}^2 \to \{0,1\}^n.
\end{aligned}$$

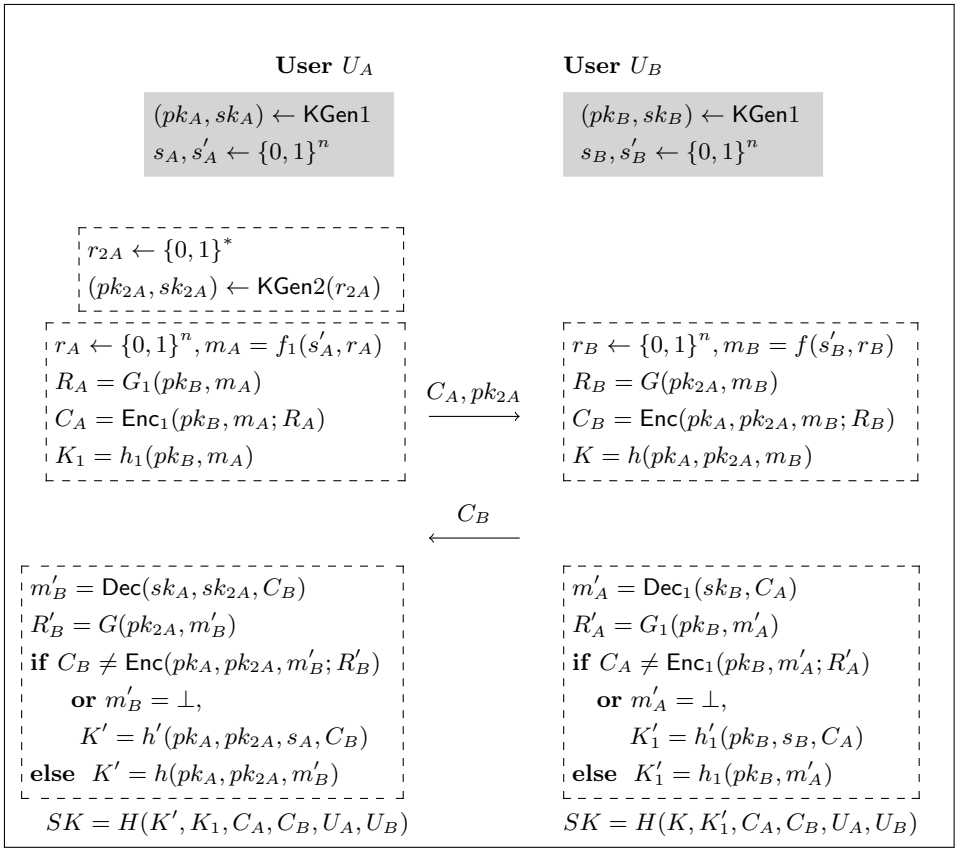

Fig. 5. Modified SIAKE in the QROM.

Setup: Each user's static public-secret key pair is generated by KGen1. Furthermore, let $s_P, s'_P \leftarrow \{0,1\}^n$ be static secret information for user U_P.

Modified SIAKE and Specifications: With the setup, the modified SIAKE between U_A and U_B is presented in Figure 5. The session state owned by U_A consists of r_{2A}, r_A. The session state owned by B consists of r_B. The main modifications are adding public keys into hash functions h and G.

Theorem 1. *Assume 2PKE$_{sidh}$ is [OW-CPA, OW-CPA] secure and PKE is OW-PA secure, which both hold under the SI-DDH assumption. The modified SIAKE is CK$^+$ secure in the QROM. Concretely, assume N users are involved and there are at most l sessions between two users, for any quantum adversary $\mathcal{A}$ against modified SIAKE with at most q SessionStateReveal or SessionKeyReveal queries, and q_h (resp. q_G, q_{G_1}, q_f, q_{f_1}, q_{h_1}, $q_{h'}$, $q_{h'_1}$, q_H) quantum queries to RO h (resp. G, G_1 f, f_1, h_1, h', h'_1, H), there exist [OW-CPA, OW-CPA] solvers $\mathcal{D}$ or $\mathcal{C}$, or OW-CPA solver $\mathcal{B}$, such that,*

$$\begin{aligned}
\mathsf{Adv}^{ck+}_{\mathsf{SIAKE}}(\mathcal{A}) &\leq 4N^2l(q_G + q_h + 2q)\sqrt{Adv^{[\mathsf{OW\text{-}CPA},\cdot]}_{\mathsf{2PKE}_{sidh}}(\mathcal{D})} + 2N^2l/\ell_s^{e_s} \\
&\quad + 2N^2l \cdot (2q + q_H + q_f + 4)2^{\frac{-n+1}{2}} + 2N \cdot (q_{h'} + q_{h'_1} + l^2)2^{\frac{-n+1}{2}}, \\
\mathsf{Adv}^{ck+}_{\mathsf{SIAKE}}(\mathcal{A}) &\leq 4N^2l(q_G + q_h + 2q)\sqrt{Adv^{[\cdot,\mathsf{OW\text{-}CPA}]}_{\mathsf{2PKE}_{sidh}}(\mathcal{C})} + 2N^2l \cdot (q + q_H)2^{\frac{-n+1}{2}} \\
&\quad + 2N \cdot (2q + q_{h'} + q_f)2^{\frac{-n+1}{2}}, \\
\mathsf{Adv}^{ck+}_{\mathsf{SIAKE}}(\mathcal{A}) &\leq 4N^2l(q_{G_1} + q_{h_1} + 2q)\sqrt{Adv^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B})} \\
&\quad + 2N^2l \cdot (2q + q_H + q_{f_1} + 4)2^{\frac{-n+1}{2}} + 2N \cdot (q_{h'} + q_{h'_1})2^{\frac{-n+1}{2}}.
\end{aligned}$$

Proof of Theorem 1 (Sketch). Here, we give a sketch of the proof to illustrate the core idea. For details please refer to Section 4.

First of all, we assume the adversary does not make any SessionKeyReveal query. As in the definition, the adversary falls into one of the cases from E_1 to $E_{8\text{-}2}$ in table 2. Take event E_3 as an example, where the adversary sends pk^*_{2A} in the test-session and he/she knows r_B but does not know sk_A, s_A, s'_A and sk_B, s_B, s'_B used in this session. By Lemma 2 which says that quantum random oracle could be used as a pseudorandom function, $m^*_B = f(s'_B, r_B)$, i.e., the message, is randomly chosen. The OW2H lemma and $[\mathsf{OW\text{-}CPA}, \cdot]$ security would guarantee the randomness of K^* and the final session key. We can easily extend the argument for the case E_3 other cases.

Now we consider the case that the adversary makes the SessionKeyReveal queries as well. For E_5 the analysis is still the same. However, for other cases like E_3, the simulator does not hold the static secret key sk_A of U_A. If the adversary makes SessionKeyReveal queries that involve U_A, the simulator fails to compute the encapsulated key and the final session key. In the classical ROM, it is easy to overcome this obstacle by searching the hash lists and taking pk_{2A} as input of h, which is how SIAKE (and X3LH) handles.

Whereas to simulate SessionKeyReveal queries in QROM, we should embed the encryption under many public keys into h. Thus, both static and ephemeral public keys should be included as the inputs of h, which makes new issues arise, in particular, that public keys may not be honestly generated and the encryption may not be injective. Nevertheless, with solutions highlighted in our technique overview, we could build two lists of (honestly generated) static public keys and ephemeral public keys at the very beginning. Then, we could apply encryption-then-hash and decouple the static secret key of the test-session with the SessionKeyReveal oracle. Afterward, with a careful choice of S, the OW2H lemma can be applied to argue the randomness of the session key in the test-session.

Concretely, for E_3, the security is argued with a sequence of games as shown in Table 3. At first, we generate two lists L_1, L_2 of honest static keys for all (honest) users and their sessions. The session key for the invalid ciphertext (that involves U_A, the owner of the test-session) is computed with private oracles of ciphertext. Then, oracle query to h with an input (pk_1, pk_2, m) is conceptually replaced by the encryption-then-hash $h_q(pk_1, pk_2, \mathsf{Enc}(pk_1, pk_2, m; G(pk_2, m)))$ when $pk_1 \times pk_2 \in L_1 \times L_2$. We do the encryption-then-hash for $h_1(pk_1, m_1)$ with another private random oracle. Conceptually, the encapsulated key in the valid ciphertext is computed with the private oracles. By integrating the decapsulation for both the valid and invalid ciphertexts, we could avoid using the static secret key of U_A when answering SessionKeyReveal. After randomizing plaintext m^*_B in Game 5, we could replace $G \times h$ with a new oracle that differs from $G \times h$ on a set S. The set S should be carefully chosen such that the randomness and encapsulated key in the test-session are altered and the answer for SessionKeyReveal on any other session remains. Then, we can apply the OW2H lemma and argue the distinguisher with a square root of the advantage to solve the one-wayness game of the underlying 2-key PKE. Finally, since the quantum random oracle is a pseudorandom function, the session key in the test-session is pseudorandom as well.

The analyses could be extended to all the other cases.

4 Formal Security Proof

To prove Theorem 1, we should handle every case from Table 2. The reduction algorithm reduces the advantage of CK^+ adversary to that of attacking $[\mathsf{OW\text{-}CPA}, \cdot]$, $[\cdot, \mathsf{OW\text{-}CPA}]$ of $\mathsf{2PKE}_{\mathsf{sidh}}$ or OW-CPA of PKE depending on which case we cope with. For cases E_3, E_4, E_6, $E_{7\text{-}1}$ and $E_{8\text{-}2}$, the security relies on $[\mathsf{OW\text{-}CPA}, \cdot]$ of $\mathsf{2PKE}_{\mathsf{sidh}}$, and the sequences of games proceed similarly. For cases E_1, E_2, $E_{7\text{-}2}$ and $E_{8\text{-}1}$, the security relies on OW-CPA of PKE. And case E_5 (which is wPFS security) relies on $[\cdot, \mathsf{OW\text{-}CPA}]$ of $\mathsf{2PKE}_{\mathsf{sidh}}$.

Games		h (for encapsulated key) h_1 (for encapsulated key) G/G_1 (for randomness)	K of valid C **Justification** K of invalid C	R_B^* m_B^*/K^* SK^* (session key)
Games 0-1	I	$h(pk_1, pk_2, m)$	$h(pk_A, pk_{2,A}^j, m)/h_1(pk_A, m_1)$	$G(pk_{2A}^*, m_B^*)$
	II	$h_1(pk_1, m_1)$	– – – – –	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h'(pk_A, pk_{2,A}^j, s_A, C)/h'_1(pk_A, s_A, C)$	$H(K^*, \cdots)$
Game 2	I	$h(pk_1, pk_2, m)$	$h(pk_A, pk_{2,A}^j, m)/h_1(pk_A, m_1)$	$G(pk_{2A}^*, m_B^*)$
	II	$h_1(pk_1, m_1)$	**Lemma 3**	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h'_q(pk_A, pk_{2,A}^j, C)/h'_{1q}(pk_A, C)$	$H(K^*, \cdots)$
Game 3	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$G(pk_{2A}^*, m_B^*)$
	II	h_q^3 or h_q^4	**Conceptual**	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h'_q(pk_A, pk_{2,A}^j, C)/h'_{1q}(pk_A, C)$	$H(K^*, \cdots)$
Game 4	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$G(pk_{2A}^*, m_B^*)$
	II	h_q^3 or h_q^4	**Conceptual**	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$G(pk_2, m)/G_1(pk_1 m_1)$	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$H(K^*, \cdots)$
Game 5	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$G(pk_{2A}^*, m_B^*)$
	II	h_q^3 or h_q^4	**Lemma 4**	$f_r(r_B)$ $/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$H(K^*, \cdots)$
Game 6	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$R_B^* \leftarrow \mathcal{R}$
	II	h_q^3 or h_q^4	**Lemma 5/OW2H**	$f_r(r_B)/$ $K^* \leftarrow \{0,1\}^n$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$H(K^*, \cdots)$
Game 7	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$R_B^* \leftarrow \mathcal{R}$
	II	h_q^3 or h_q^4	**Lemma 2**	$f_r(r_B)/K^* \leftarrow \{0,1\}^n$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$SK^* \leftarrow \{0,1\}^n$

Table 3. Overview of games for the proof of Theorem 1 w.r.t case E_3. Some details are not shown in this table, such as building lists, the guess of test-session, the abort events, and some replacements of random oracles. Please refer to the Games for details. "valid C" and "invalid C" are those ciphertexts received by U_A, the owner of test-session. m_B^*, R_B^*, K^* indicate the message, randomness, encapsulated key corresponding to the ciphertext in test-session. SK^* is the session key of test-session.

Here, taking E_3 as example, we show the game sequence of proof in detail, which is illustrated in Table 3. For all the other cases, we will highlight the differences with taht for E_3.

In the following, we let Adv_i be $|\Pr[\mathcal{A} \Rightarrow 1|b = 1 \text{ in Game } i] - \Pr[\mathcal{A} \Rightarrow 1|b = 0 \text{ in Game } i]|$.

Game 0. This is the original CK$^+$ game as defined in Section 2.3.

In this game, $\mathcal{A}$ could query Send, $\mathsf{Corrupt}$, $\mathsf{SessionKeyReveal}$ and $\mathsf{SessionStateReveal}$ oracles whenever it wants. Note that $\mathcal{A}$ is also given access to quantum ROs for $f, f_1, G, G_1, h, h_1, h', h'_1$ and H. At some point, $\mathcal{A}$ chooses a test-session, and he may send messages or passively keep track of messages of test-session belonging to one of the cases in Table 2. As said before, we take E_3 as an example. Then $\mathcal{A}$ receives the test-session key SK^* or a totally random key depending on $b = 1$ or 0. After more queries to Send and other allowed oracles and quantum ROs, $\mathcal{A}$ finally outputs a bit b'. Let $\mathsf{Adv}^{ck+}_{\mathsf{SIAKE}}(\mathcal{A}) = \mathsf{Adv}_0$.

Game 1. In this game, we prepare honestly generated static keys and ephemeral keys for all users at the very beginning of the CK$^+$ game, in the state of on-the-fly in Game 0. As shown in Table 4, let $\{(pk_{1,i}, sk_{1,i})_{1\le i\le N}\}$ be the list of honest static public-secret keys, and L_1 be that of the public keys. Let $\{(pk_{2,i}^j, sk_{2,i}^j)_{1\le i\le N, 1\le j\le 2Nl}\}$ be the list of ephemeral public-secret keys, and L_2 be that of the public keys. Specially, $(pk_{1,i}, sk_{1,i})$ is the static

public-secret keys prepared for U_i and $pk^j_{2,i}$ is used by U_i as ephemeral public key in its j-th session when it's initiator.

We would like to highlight that this does not mean the prepared keys are always used in the real game For example, the adversary may arbitrarily register an *invalid* public key for U_i, then the pair $(pk_{1,i}, sk_{1,i})$ is not used. This also may happen for ephemeral keys, since the adversary may make Send queries. Fortunately, we do not need to answer the $\mathsf{SessionKeyReveal}$ query in these cases.

Since this is only a conceptual change, we have $\mathsf{Adv}_0 = \mathsf{Adv}_1$.

For user	L_1 : Static Public Keys	L_2 : Ephemeral Public Keys
U_1	$pk_{1,1}$	$pk^1_{2,1}, \cdots, pk^{2Nl}_{2,1}$
:	:	:
U_i	$pk_{1,i}$	$pk^1_{2,i}, \cdots, pk^{2Nl}_{2,i}$
:	:	:
U_N	$pk_{1,N}$	$pk^1_{2,N}, \cdots, pk^{2Nl}_{2,N}$

Table 4. Prepared static keys and ephemeral keys.

Game 2. In this game, we guess the owner of the test-session, denote it by U_A, and change the way to compute encapsulated keys for invalid ciphertexts received by U_A. Assume that pk_A is the static public key, and (sk_A, s_A, s'_A) are the static secret keys of U_A. When U_A as an initiator uses $pk^j_{2,A}$ as an ephemeral public key and receives an invalid ciphertext C^j_A, the encapsulated key in C^j_A, i.e., $h'(pk_A, pk^j_{2,A}, s_A, C^j_A)$ is replaced by

$$h'_q(pk_A, pk^j_{2,A}, C^j_A); \tag{1}$$

and when U_A as a responder and receives an invalid ciphertext C^j_A, the encapsulated key, i.e., $h'_1(pk_A, s_A, C^j_A)$ is replaced by

$$h'_{1q}(pk_A, C^j_A), \tag{2}$$

where $h'_q : \mathcal{D}_1 \times \mathcal{D}_2 \times \mathcal{C} \to \{0,1\}^n$ and $h'_{1q} : \mathcal{D}_1 \times \mathcal{C}_1 \to \{0,1\}^n$ are internal hash functions.

Lemma 3. $\mathsf{Adv}_1 \leq N \cdot \mathsf{Adv}_2 + 2N(2Nl + q_{h'} + q_{h'_1})2^{\frac{-n+1}{2}}$.

Game 3. We change the way to answer queries to h (resp. h_1), and also change the way to compute K (resp. K_1) from the valid ciphertext received by U_A. This game is to make preparation for getting rid of the usage of sk_A during $\mathsf{SessionKeyReveal}$ queries that involve U_A.

Firstly, h (resp. h_1) is answered using two internal random oracles according to the domain separation:

$$h(pk_1, pk_2, m) = \begin{cases} h^1_q(pk_1, pk_2, \mathsf{Enc}(pk_1, pk_2, m; \tilde{G}(pk_2, m))) & \text{if } \ pk_1 \times pk_2 \in L_1 \times L_2 \\ h^2_q(pk_1, pk_2, m) & \text{o.w.} \end{cases}$$

$$h_1(pk_1, m_1) = \begin{cases} h^3_q(pk_1, \mathsf{Enc}_1(pk_1, m_1; \tilde{G}_1(m_1))) & \text{if } \ pk_1 \in L_1 \\ h^4_q(pk_1, m_1) & \text{o.w.} \end{cases}$$

where $h^1_q : \mathcal{D}_1 \times \mathcal{D}_2 \times \mathcal{C} \to \{0,1\}^n$, $h^3_q : \mathcal{D}_1 \times \mathcal{C}_1 \to \{0,1\}^n$, $h^2_q : \mathcal{D}_1 \times \mathcal{D}_2 \times \{0,1\}^n \to \{0,1\}^n$, and $h^4_q : \mathcal{D}_1 \times \{0,1\}^{n_1} \to \{0,1\}^n$ are internal oracles. Because of the perfect correctness of $\mathsf{2PKE_{sidh}}$ and PKE, both the derandomized Enc in h^1_q and Enc_1 in h^3_q are injective functions on messages. Thus h and h_1 are still uniformly random. This is only a conceptual change.

Secondly, when U_A, as an initiator, uses $pk^j_{2,A}$ as ephemeral public key and receives a valid ciphertext C^j_A, then $K = h(pk_A, pk^j_{2,A}, \mathsf{Dec}(sk_A, sk^j_{2,A}, C^j_A))$ is replaced by

$$h^1_q(pk_A, pk^j_{2,A}, C^j_A). \tag{3}$$

When U_A is a responder and receives a valid ciphertext C^j_A, $h_1(pk_A, \mathsf{Dec}_1(sk_A, C^j_A))$ is replaced by

$$h^3_q(pk_A, C^j_A). \tag{4}$$

This is also a conceptual replacement. By checking the cases one by one, the replacements for encapsulated keys of valid ciphertexts are consistent with the replacements of h and h_1. The view of $\mathcal{A}$ in Game 2 and Game 3 is identical even for unbounded (quantum) adversary. Thus, $\mathsf{Adv}_2 = \mathsf{Adv}_3$.

Game 4. Now we are ready to get rid of using the secret key sk_A to answer SessionKeyReveal queries. We incorporate the ways to decapsulate K for both valid and invalid ciphertexts received by U_A. For an invalid ciphertext sent to U_A, the encapsulated key K is computed the same as for a valid ciphertext. Concretely, Equ.(1) of Game 2 is replaced by Equ.(3), and Equ.(2) is replaced by Equ.(4).

Since h^1_q and h^3_q are internal oracles, the adversary can only access to h^1_q and h^3_q by querying h. The ciphertexts on which $\mathcal{A}$ could query to h^1_q and h^3_q are all valid. However, the ciphertexts on which $\mathcal{A}$ queries to h'_q, h'_{1q} are all invalid. That is, the domain consisting of all the ciphertexts on which $\mathcal{A}$ could query to h^1_q (resp. h^3_q) is disjoint with that of h'_q (resp. h'_{1q}). Switching the internal oracles when receiving invalid ciphertexts does not change the view of an unbounded (quantum) adversary. Thus, $\mathsf{Adv}_3 = \mathsf{Adv}_4$.

Note that in Game 4 and the subsequent games, the secret key sk_A is not used anymore. We are ready to decouple K^* from the challenge ciphertext in the test-session.

Game 5. Let $L_{\mathsf{2after}} \subset L_2$ be $\{pk^j_{2,A}\}_{Nl+1 \le j \le 2Nl}$. After the test-session, all the ephemeral public keys used by U_A will be chosen from L_{2after}. If any public key in L_{2after} is equal to pk^*_{2A}, abort. Secondly, we guess the responder of test-session and denote it as U_B, and also guess which session between U_A and U_B is the test-session at the very beginning. If the guess fails, just abort. Thirdly, we change the generation of m_B as $m_B := f_r(r_B)$ with an internal random oracle f_r, which is identical to $m_B \leftarrow \mathcal{M}$. Particularly, in the test-session $m^*_B \leftarrow \mathcal{M}$. Finally, if there exists a message used by $\mathcal{A}$ (to interacts with U_A) before the test-session, which is equal to m^*_B, the game also aborts.

Lemma 4. $\mathsf{Adv}_4 - Nl \cdot \mathsf{Adv}_5 \le 2Nl \cdot (q + q_f)2^{\frac{-n+1}{2}} + 2Nl/\ell^{e_s}_s + Nl^2 2^{-n+1}$.

Now, random oracles G and h can be regarded as a single oracle $G \times h$. As shown in [25], if G and h have the same domain, we can use $G \times h$ to simulate them. Here, we could easily construct a G' that can simulate G with the same domain as h,i.e., $G'(pk_A, pk_2, m) = G(pk_2, m)$. Then, a query (pk_2, m) to G can be directly converted to a query (pk_A, pk_2, m) to G' for fixed pk_A. Therefore, we can use $G' \times h$ to simulate both G and h. For simplicity, in the context, we stick to using $G \times h$ instead of $G' \times h$. We take $\{pk_A\} \times \mathcal{D}_2 \times \mathcal{M}$ as the domain of G. Looking ahead, the same holds for $\ddot{G}$ and $\ddot{h}$ defined in Game 6.

Game 6. Define set $S := \{pk_A\} \times \mathcal{D}_2 \setminus L_{\mathsf{2after}} \times \{m^*_B\}$. Let $\ddot{h}$ (resp. $\ddot{G}$) be a function such that the function values on S (resp. $\mathcal{D}_2 \setminus L_{\mathsf{2after}} \times \{m^*_B\}$) are totally random, and $\ddot{h} = h$ (resp. $\ddot{G} = G$) everywhere else. In this game, $G \times h$ is replaced by $\ddot{G} \times \ddot{h}$.

A equivalent description of this game is that: $G \times h$ is still the same, but now their values on S that we provide to $\mathcal{A}$ in the CK$^+$ games are totally random. Specially, the randomess $R^*_B = G(pk^*_{2A}, m^*_B)$ for C^*_B and encapsulated key $K^* = h(pk_A, pk^*_{2A}, m^*_B)$ are replaced by random strings.

Lemma 5. $\mathsf{Adv}_5 - \mathsf{Adv}_6 \le 2(q_G + q_h + 2q)\sqrt{Adv^{[\mathsf{OW\text{-}CPA},\cdot]}_{\mathsf{2PKE}_{sidh}}(\mathcal{D})}$.

Game 7. We change the session key SK^* to a totally random key. With a similar argument on the third modification of Game 5, by Lemma 2, $\mathsf{Adv}_6 - \mathsf{Adv}_7 \leq 2(q+q_H)2^{\frac{-n+1}{2}}$, since K^* is totally random from the view of $\mathcal{A}$. Now, SK^* is always random and $\mathsf{Adv}_7 = 0$.

To sum up, we give the upper bound of AKE adversary for the case E_3 as the first in-equation in Theorem 1.

For case E_4, the proof of the game sequences is almost the same as for E_3, except that in Game 5 the AKE adversary $\mathcal{A}$ does not know r_B for E_4, while he does not know s'_B for E_3 instead. For case E_2, the difference with proof of E_3 lies in that the role of U_A and U_B is exchanged, and the challenge ciphertext is under 1-key public key encryption in E_2, while in case E_3 it is under 2-key PKE instead. For case E_1, the proof of its game sequences is almost the same as for E_2, except that in E_1 the AKE adversary $\mathcal{A}$ does not know r_A, while he does not know s'_A in E_2 instead.

For cases E_6, E_{7-1}, E_{7-2}, E_{8-1}, E_{8-2}, the proof is almost the same as for E_3, E_1, E_4, E_2, E_3 respectively.

For case E_5, the proof is much simpler since we only need to deal with the weak perfect forward security, which means no decapsulation oracle is needed, and the injective mapping with encryption under many public keys technique can be left out. □

4.1 Proof of Lemmas

4.1.1 Proof of Lemma 3: $\mathsf{Adv}_1 \leq N \cdot \mathsf{Adv}_2 + 2N(2Nl + q_{h'} + q_{h'_1})2^{\frac{-n+1}{2}}$.

Let Game 1-1 be an internal game in which we guess whom is the owner of test-session, i.e., U_A. If the guess is wrong, abort. Obviously, $\mathsf{Adv}_1 = N \cdot \mathsf{Adv}_{1-1}$.

To argue the difference between Adv_{1-1} and Adv_2, there are two cases that should be handled, namely, either U_A is an initiator or a responder. Here, we prove the case when U_A is an initiator. Note that s_A is totally random for $\mathcal{A}$. By Lemma 2, we construct an algorithm $\mathcal{T}$ to distinguish which oracle it is given access to, $h'_q(\cdot,\cdot,\cdot)$ or $h'(\cdot,\cdot,s_A,\cdot)$. To this end, $\mathcal{T}$ simulates the AKE game. It first guesses the owner of test-session, generates the static public-secret keys for every user except U_A. For user U_A, $\mathcal{T}$ only honestly generates (pk_A, sk_A) but without knowing s_A. For an invalid ciphertext C_A^j sent to initiator U_A who uses $pk_{2,A}^j$ as ephemeral public key, $\mathcal{T}$ makes query to the challenge random oracle $h'(\cdot,\cdot,s_A,\cdot)$ or $h'_q(\cdot,\cdot,\cdot)$ with tuple $pk_A, pk_{2,A}^j$, C_A^j depending on $\sigma = 1$ or 0, to set the encapsulated key K. After receiving b' from $\mathcal{A}$, $\mathcal{T}$ returns b' as the guess of σ.

If the challenge oracle $\mathcal{T}$ queried is $h'(\cdot,\cdot,s_A,\cdot)$, it is Game 1-1. If the oracle $\mathcal{T}$ queried is $h'_q(\cdot,\cdot,\cdot)$, it is Game 2. The number of $\mathcal{T}$'s queries to h'_q is less than $Nl + q_{h'}$. By Lemma 2, for both $b = 1$ and 0, $\Pr[b' = 1|\sigma = 1] - \Pr[b' = 1|\sigma = 0] \leq (Nl + q_{h'})2^{\frac{-n+1}{2}}$. To further consider the similar replacement of h'_1 when U_A is a responder, we have $\mathsf{Adv}_{1-1} - \mathsf{Adv}_2 \leq 2(Nl+q_{h'})2^{\frac{-n+1}{2}} + 2(Nl+q_{h'_1})2^{\frac{-n+1}{2}}$. Thus, $\mathsf{Adv}_1 \leq N \cdot \mathsf{Adv}_2 + 2N(2Nl+q_{h'}+q_{h'_1})2^{\frac{-n+1}{2}}$. □

4.1.2 Proof of Lemma 4:

$$\mathsf{Adv}_4 - Nl \cdot \mathsf{Adv}_5 \leq 2Nl \cdot (q+q_f)2^{\frac{-n+1}{2}} + 2Nl/\ell_s^{e_s} + Nl^2 2^{-n+1}.$$

Define an intermediate Game 4-1, in which it aborts when there exists a public key in $L_{2\mathsf{after}}$ equal to the ephemeral public key used by adversary in test-session. Obviously $\mathsf{Adv}_4 - \mathsf{Adv}_{4-1} \leq Nl/\ell_s^{e_s}$.

Define an intermediate Game 4-2, in which the simulator guesses who is the responder (here assume it is U_B) of test-session and which session is the test-session. If the guess succeeds, go on as Game 4-1, otherwise abort. The probability of successful guess is exactly $1/Nl$. Thus, $\mathsf{Adv}_{4-1} = Nl \cdot \mathsf{Adv}_{4-2}$.

Define an intermediate Game 4-3, in which the message used by U_B is randomly chosen. Note that s'_B is totally random for $\mathcal{A}$. By Lemma 2, we construct an algorithm $\mathcal{T}$ to distinguish oracle f_r from oracle $f(s'_B,\cdot)$. Given quantum accessible random oracle f, $\mathcal{T}$ is

given access to $f(s'_B, \cdot)$ if $\sigma = 1$; otherwise $\mathcal{T}$ is given access to f_r. $\mathcal{T}$ finally outputs a guess σ' for σ. To this end, $\mathcal{T}$ simulates the AKE game. $\mathcal{T}$ honestly sets the static public/secret keys of every user except U_B. For user U_B, $\mathcal{T}$ honestly sets the static secret key but leaves s'_B as empty. For any session that involves U_B, $\mathcal{T}$ queries oracle $f(s'_B, \cdot)$ or $f_r(\cdot)$ with r_B. After receiving b' from $\mathcal{A}$ as the guess of b, $\mathcal{T}$ returns b' as the guess of σ. If $\mathcal{T}$ queried $f(s'_B, \cdot)$, the AKE game is Game 6-2. If $\mathcal{T}$ queried $f_r(\cdot)$, the AKE game is Game 4-3. By Lemma 2, for both $b = 1$ and 0, $\Pr[b' = 1|\sigma = 1] - \Pr[b' = 1|\sigma = 0] \leq (q + q_f)2^{\frac{-n+1}{2}}$. Thus $\mathsf{Adv}_{4-2} - \mathsf{Adv}_{4-3} \leq 2(q + q_f)2^{\frac{-n+1}{2}}$.

Furthermore, since now m^*_B is randomly chosen, the probability that it is equal to any message used by $\mathcal{A}$ (to interacts with U_A) before the test-session, is less than $l \times 2^{-n}$.

To sum up, $\mathsf{Adv}_4 - Nl \cdot \mathsf{Adv}_5 \leq 2Nl(q + q_f)2^{\frac{-n+1}{2}} + Nl/\ell_s^{e_s} + Nl^2 \times 2^{-n+1}$. □

4.1.3 Proof of Lemma 5: $\mathsf{Adv}_5 - \mathsf{Adv}_6 \leq 2(q_G + q_h + 2q)\sqrt{\mathrm{Adv}^{[\mathsf{OW\text{-}CPA},\cdot]}_{\mathsf{2PKE}_{\mathsf{sidh}}}(\mathcal{D})}$.

Let $\tilde{h}^1_q$ be the function, which aborts on set $S = \{pk_A\} \times \mathcal{D}_2 \setminus L_{\mathsf{2after}} \times \{m^*_B\}$, and is equal to h^1_q everywhere else. Since any $\mathsf{SessionKeyReveal}$ query on non-test-session does not need h^1_q on S, $\tilde{h}^1_q$ and h^3_q could be used to answer the $\mathsf{SessionKeyReveal}$ query that involves U_A in both Game 5 and Game 6.

Define $\mathcal{S}^{G\times h}$ as the following: on input

$$z = (L_1, L_2, pk_A, \mathsf{Enc}(pk_A, \cdot, m^*_B; G(\cdot, m^*_B)), h(pk_A, \cdot, m^*_B), \tilde{h}^1_q, h^3_q),$$

where $\mathsf{Enc}(pk_A, \cdot, m^*_B; G(\cdot, m^*_B)), h(pk_A, \cdot, m^*_B)$ can be seen as two one-time oracles, generate the static public-secret key pairs for U_P as in Game 5. Set pk_A as the static public key of U_A. For any $\mathsf{SessionKeyReveal}$ query that involves U_A, utilize $\tilde{h}^1_q$ and h^3_q to compute the encapsulated key and session key. When $\mathcal{A}$ sends pk^*_{2A} in test-session, $\mathcal{S}$ computes $C^*_B := \mathsf{Enc}(pk_A, pk^*_{2A}, m^*_B; G(pk^*_{2A}, m^*_B))$, and $K^* := h(pk_A, pk^*_{2A}, m^*_B)$ with pk^*_{2A} and z. $\mathcal{S}$ chooses $b \leftarrow \{0, 1\}$. If $b = 0$, set $SK^* = H(K^*, K_1, C_A, C^*_B, U_A, U_B)$, else $SK^* = H(K, K_1, C_A, C^*_B, U_A, U_B)$, where K_1 is extracted from C_A using sk_B and $K \leftarrow \{0, 1\}^n$. $\mathcal{S}$ then returns what $\mathcal{A}$ outputs.

Thus, on input z, $\mathcal{S}^{G\times h}$ could simulate Game 5 perfectly. By replacing $G \times h$ with $\ddot{G} \times \ddot{h}$, on the same input, $\mathcal{S}^{\ddot{G}\times\ddot{h}}$ could also simulate Game 6 perfectly. Thus,

$$\mathsf{Adv}_7 = |\Pr[\mathcal{S}^{G\times h} \Rightarrow 1|b = 1] - \Pr[\mathcal{S}^{G\times h} \Rightarrow 1|b = 0]|;$$
$$\mathsf{Adv}_8 = |\Pr[\mathcal{S}^{\ddot{G}\times\ddot{h}} \Rightarrow 1|b = 1] - \Pr[\mathcal{S}^{\ddot{G}\times\ddot{h}} \Rightarrow 1|b = 0]|.$$

Let $B^{\ddot{G}\times\ddot{h}}$ be an oracle algorithm that: with the input z does as following: randomly choose $k \leftarrow \{1, \cdots, q_G + q_h\}$, run $\mathcal{S}^{\ddot{G}\times\ddot{h}}(z)$ until (exactly before the starting of) the k-th query, measure all queried input registers in the computational basis, and output the measurement as outcomes.

For $b = 0$ and 1, applying the OW2H (Lemma 1) by setting $X = \mathcal{D}_1 \times \mathcal{D}_2 \times \mathcal{M}$, $Y = \{0, 1\}^{r+n}$, $S = \{pk_A\} \times \mathcal{D}_2 \setminus L_{\mathsf{2after}} \times \{m^*_B\}$, $\mathcal{O}_1 = \ddot{G} \times \ddot{h}$, $\mathcal{O}_2 = G \times h$, and $z = (L_1, L_2, pk_A, \mathsf{Enc}(pk_A, \cdot, m^*_B; G(\cdot, m^*_B)), h(pk_A, \cdot, m^*_B), \tilde{h}^1_q, h^3_q)$, we have

$$|\Pr[\mathcal{S}^{\ddot{G}\times\ddot{h}}(z) \Rightarrow 1] - \Pr[\mathcal{S}^{G\times h}(z) \Rightarrow 1]| \leq 2(q_G + q_h + 2q)\sqrt{\Pr[s \in S|s \leftarrow B^{\ddot{G}\times\ddot{h}}(z)]}.$$

To bound the probability $\Pr[s \in S|s \leftarrow B^{\ddot{G}\times\ddot{h}}(z)]$, we construct an adversary $\mathcal{D}$ against the $[\mathsf{OW\text{-}CPA}, \cdot]$-security.

- Select $k \leftarrow \{1, \cdots, q_G + q_h\}$.
- Given the pk_A from the $[\mathsf{OW\text{-}CPA}, \cdot]$ challenger, generate a list of static keys and include an additional pair $(pk_A, -)$ to get L_1, and generate a list of ephemeral keys, L_2.

– $\mathcal{D}$ randomly guesses which users are the initiator U_A and responder U_B in the test-session, and which session is the test-session.
– Pick $2q_G$ ($2q_{G_1}$, $2q_h$, $2q_{h_1}$, $2q_{h'}$, $2q_{h'_1}$, $2q_f$, $2q_{f_1}$,$2q_H$, $2q_{h_q^1}$, $2q_{h_q^2}$, $2q_{h_q^3}$, $2q_{h_q^4}$)-wise independent function uniformly to simulate the random oracle $\ddot{G}$ (G_1, $\ddot{h}$, h_1, h', h'_1, f, f_1, H, h_q^1, h_q^2, h_q^3, h_q^4).
– For any $\mathsf{SessionKeyReveal}$ query that does not involve U_A, the challenger uses static secret key to extract encapsulated and session key. For any $\mathsf{SessionKeyReveal}$ query that involves U_A, it answers using h_q^i, for $1 \le i \le 4$.
– On receiving pk_{2A}^* and C_A in test-session, forward pk_{2A}^* to $[\mathsf{OW\text{-}CPA}, \cdot]$ game. On receiving challenge ciphertext C^* under pk_A and pk_{2A}^*, choose $K \leftarrow \mathcal{K}$ and return $SK^* = H(K, K_1 = h_1(pk_B, \mathsf{Dec}_1(sk_B, C_A)), C_A, C^*, U_A, U_B)$.
– Measure the argument of the k-th query to $\ddot{G} \times \ddot{h}$ and output m.

Since $\mathcal{D}$ simulates perfectly, $\Pr[s \in \mathcal{S} | s \leftarrow B^{\ddot{G} \times \ddot{h}}(z)] = \mathrm{Adv}^{[\mathsf{OW\text{-}CPA}, \cdot]}_{\mathsf{2PKE}_{\mathsf{sidh}}}(\mathcal{D})$.

By summing the equations, $\mathsf{Adv}_5 - \mathsf{Adv}_6 \le 4(q_G + q_h + 2q)\sqrt{\mathrm{Adv}^{[\mathsf{OW\text{-}CPA}, \cdot]}_{\mathsf{2PKE}_{\mathsf{sidh}}}(\mathcal{D})}$. □

5 New Construction and Extension

We first give a new instantiation based on commutative supersingular isogeny, then extend the proof for modified SIAKE to that for modified $\mathsf{X3LH}$.

5.1 CSIAKE from Commutative Supersingular Isogenies

Castryck et al. [15] proposed a commutative supersingular isogeny Diffie-Hellman (CSIDH) key exchange. Although concrete parameters for CSIDH is heavily debated [9,37,4,11], it is considered as a candidate of quantum-resistant primitive.

Let $p = 4 \times l_1 \cdots l_n - 1$ be a large prime, where each l_i is a small distinct odd prime. p and the supersingular elliptic curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_p$ with endomorphism ring $\mathcal{O} = \mathbb{Z}[\pi]$ are public parameters. The CSIDH key exchange works as following: Alice randomly chooses $(e_{A1}, \cdots, e_{An})$ from a range $[-m, m]$. These integers represent the ideal class $[\mathfrak{a}] = [\mathfrak{l}_1^{e_{A1}} \cdots \mathfrak{l}_n^{e_{An}}] \in \mathrm{cl}(\mathcal{O})$. Alice computes $[\mathfrak{a}]E_0$. Bob chooses his own secret $[\mathfrak{b}]$ and computes $[\mathfrak{b}]E_0$. They both could compute the common curve $[\mathfrak{a}][\mathfrak{b}]E_0 = [\mathfrak{b}][\mathfrak{a}]E_0$ in the form $y^2 = x^3 + sx^2 + x$. The share secret is the Montgomery coefficient of common curve, i.e., s.

We propose a 2-key PKE $\mathsf{2PKE}_{\mathsf{csidh}}$ and an one-key $\mathsf{PKE}_{\mathsf{csidh}}$ from CSIDH in Fig.6.

$\mathsf{KGen1}$	$\mathsf{Enc}_1(pk_1, m_1)$	$\mathsf{Dec}_1(sk_1, C)$
$(e_{11}, \cdots, e_{1n}) \leftarrow [-m, m]^n$ $sk_1 = [\mathfrak{a}_1] = [\mathfrak{l}_1^{e_{11}} \cdots \mathfrak{l}_n^{e_{1n}}]$ $pk_1 = [\mathfrak{a}_1]E_0$	$(f_1, \cdots, f_n) \leftarrow [-m, m]^n$ $c_1 = [\mathfrak{b}]E_0 = [\mathfrak{l}_1^{f_1} \cdots \mathfrak{l}_n^{f_n}]E_0$ $c_2 = \mathsf{H}(\mathsf{Coef}([\mathfrak{b}]pk_1)) \oplus m_1$	$(c_1, c_2) \leftarrow C$ $m_1 = c_2 \oplus \mathsf{H}(\mathsf{Coef}([\mathfrak{a}_1]c_1))$
$\mathsf{KGen2}$	$\mathsf{Enc}(pk_1, pk_2, m_1 \| m_2)$	$\mathsf{Dec}(sk_1, sk_2, C)$
$(e_{21}, \cdots, e_{2n}) \leftarrow [-m, m]^n$ $sk_2 = [\mathfrak{a}_2] = [\mathfrak{l}_1^{e_{21}} \cdots \mathfrak{l}_n^{e_{2n}}]$ $pk_2 = [\mathfrak{a}_2]E_0$	$(f_1, \cdots, f_n) \leftarrow [-m, m]^n$ $c_1 = [\mathfrak{b}]E_0 = [\mathfrak{l}_1^{f_1} \cdots \mathfrak{l}_n^{f_n}]E_0$ $c_2 = \mathsf{H}(\mathsf{Coef}([\mathfrak{b}]pk_1)) \oplus m_1$ $c_3 = \mathsf{H}(\mathsf{Coef}([\mathfrak{b}]pk_2)) \oplus m_2$	$(c_1, c_2, c_3) \leftarrow C$ $m_1 = c_2 \oplus \mathsf{H}(\mathsf{Coef}([\mathfrak{a}_1]c_1))$ $m_2 = c_3 \oplus \mathsf{H}(\mathsf{Coef}([\mathfrak{a}_2]c_1))$

Fig. 6. $\mathsf{2PKE}_{\mathsf{csidh}} = (\mathsf{KGen1}, \mathsf{KGen2}, \mathsf{Enc}, \mathsf{Dec})$ and $\mathsf{PKE}_{\mathsf{csidh}} = (\mathsf{KGen1}, \mathsf{Enc}_1, \mathsf{Dec}_1)$ based on CSIDH. $\mathsf{Coef}(\cdot)$ is Montgomery coefficient of the input curve. $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ is a random pair-wise independent hash function.

Definition 3 (CSI-DDH Assumption[5]). *Let* $E_A = [\mathfrak{a}]E_0$, $E_B = [\mathfrak{b}]E_0$, $E_C = [\mathfrak{c}]E_0$ *be randomly chosen.* $b \leftarrow \{0,1\}$. *If* $b = 0$, $E' = E_C$, *otherwise* $E' = [\mathfrak{a}][\mathfrak{b}]E_0$. *Given* (E_0, E_A, E_B, E'), *problem solver* $\mathcal{A}$ *outputs* b' *as the guess of* b. *Define* $Adv_{\mathcal{A}}^{csiddh}(E_0, E_A, E_B, E') = Pr[b = b'] - 1/2$. *The CSI-DDH assumption states that for any PPT adversary* $\mathcal{A}$, $Adv_{\mathcal{A}}^{csiddh}$ *is negligible.*

Lemma 6. *Based on CSI-DDH assumption,* $\mathsf{2PKE_{csidh}}$ *is* [*OW-CPA*, *OW-CPA*] *secure and* $\mathsf{PKE_{csidh}}$ *is OW-CPA secure.*

Please refer to Appendix C for a formal proof of Lemma 6.

Replacing $\mathsf{2PKE_{sidh}}$ and PKE of SIAKE with $\mathsf{2PKE_{csidh}}$ and $\mathsf{PKE_{csidh}}$ respectively, we would get a secure CSIDH-based AKE, CSIAKE, in the QROM. Using CSIDH-5280 recommended by [11], the total communication of CSIAKE is 2028 Bytes, while HKSU [23] needs 2688 Bytes. The initiator and responder in HKSU need to compute 6 and 6 isogenies, while in CSIAKE they compute 6 and 5 isogenies respectively.

5.2 Extension to A Modified X3LH

We remark that, in the proof of Theorem 1, we do not use any other specific properties of $\mathsf{2PKE_{sidh}}$ except [OW-CPA, OW-CPA] security and perfect correctness. Thus, we could extend the scheme in Fig. 5 to build AKE from general 2-key PKE and one-key PKE, if the decryption failure is taken into account.

Let $\mathsf{2PKE} = (\mathsf{KGen1}, \mathsf{KGen2}, \mathsf{Enc}, \mathsf{Dec})$ be an [OW-CPA, OW-CPA] secure 2-key PKE with decryption error δ_2. Let $\mathsf{PKE} = (\mathsf{KGen1}, \mathsf{Enc}_1, \mathsf{Dec}_1)$ be an OW-CPA secure PKE with decryption error δ_1. We could abuse the same notions and scheme of Fig. 5 to get a modified secure X3LH in QROM. Compared with the original X3LH, the modifications are mainly putting public keys in two hash functions for randomness and encapsulated key.

Theorem 2. *Assume the public key entropy of 2PKE is* ε_{pk2}, N *users are involved and there are at most* l *sessions between two users. For any quantum adversary* $\mathcal{A}$ *against the modified X3LH with at most* q *SessionStateReveal or SessionKeyReveal queries, and* q_h *(resp.* q_G, q_{G_1}, q_f, q_{f_1}, q_{h_1}, $q_{h'}$, $q_{h'_1}$, q_H*) quantum queries to RO* h *(resp.* G, G_1 f, f_1, h_1, h', h'_1, H*), there exist* [*OW-CPA*, *OW-CPA*] *solvers* $\mathcal{D}$ *or* $\mathcal{C}$, *or OW-CPA solver* $\mathcal{B}$, *such that,*

$$
\begin{aligned}
\mathsf{Adv}^{ck+}_{\mathsf{X3LH}}(\mathcal{A}) \leq\ & 4N^2 l(q_G + q_h + 2q)\sqrt{Adv^{[\mathsf{OW\text{-}CPA},\cdot]}_{\mathsf{2PKE}}(\mathcal{D})} \\
& + 2N^2 l \cdot (2q + q_H + q_f + 4)2^{\frac{-n+1}{2}} + 2N \cdot (q_{h'} + q_{h'_1} + l^2)2^{\frac{-n+1}{2}} \\
& + 16N(q_G + 2q)^2\delta_2 + 16N(q_{G_1} + 2q)^2\delta_1 + 2N^2 l\varepsilon_{\mathsf{pk2}}, \\
\mathsf{Adv}^{ck+}_{\mathsf{X3LH}}(\mathcal{A}) \leq\ & 4N^2 l(q_G + q_h + 2q)\sqrt{Adv^{[\cdot,\mathsf{OW\text{-}CPA}]}_{\mathsf{2PKE}}(\mathcal{C})} + 2N^2 l \cdot (q + q_H)2^{\frac{-n+1}{2}} \\
& + 2N \cdot (2q + q_{h'} + q_f)2^{\frac{-n+1}{2}}, \\
\mathsf{Adv}^{ck+}_{\mathsf{X3LH}}(\mathcal{A}) \leq\ & 4N^2 l(q_{G_1} + q_{h_1} + 2q)\sqrt{Adv^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B})} \\
& + 2N^2 l \cdot (2q + q_H + q_{f_1} + 4)2^{\frac{-n+1}{2}} + 2N \cdot (q_{h'} + q_{h'_1})2^{\frac{-n+1}{2}} \\
& + 16N(q_G + 2q)^2\delta_2 + 16N(q_{G_1} + 2q)^2\delta_1.
\end{aligned}
$$

We present a sketch of the proof. Please refer Appendix D for details. We also only handle case E_3 here. All the other cases could be similarly extended.

The decryption failure would bring obstacle to the argument of Game 3 in Table 3. Specifically, since 2PKE and PKE are not perfect correct, the derandomized Enc in h_q^1 and Enc_1 in h_q^3 may not be injective. We borrow the technique used by [23], which rules out

[5] It is defined as a generalized form by using cryptographic invariant maps in [7]

the "bad randomness" conducing to decryption failure. Informally, we add a Game 1.5 after Game 1, in which G and G_1 are replaced by random oracles that only sample good randomness. After Game 4, we switch them back to G and G_1 in Game 4.5. This would guarantee the injective property required by Game 3.

The advantage of adversary introduced by the replacement of G and G_1 could be bounded by the Generic Distinguishing Problem given in Appendix D. Please refer Appendix D for a detailed analysis. We would like to highlight that: since the set of good randomness should be identified, simulating replaced random oracles of G and G_1 requires unbounded power, which implies that the simulator is an unbounded algorithm. It still makes sense because the differences between these games (i.e., Game 1.5, Game 2, Game 3, Game 4, and Game 4.5) are analyzed from the information-theoretical perspective.

The QROM security of FSXY. As pointed out by [44] and illustrated in Fig. 7, 2-key PKE of FSXY can be regarded as parallel execution of two PKEs. Integrating such a result to X3LH, we re-answer the open problem on (modified) FSXY's security in QROM after [23]. In our answer, the security requirement of underlying PKE is the same with FSXY, says OW-CPA, while [23] requires IND-CPA. We also note that, when applying to FSXY, our framework needs one more re-encryption than HKSU.

$\mathsf{KGen1}$	$\mathsf{KGen2}$	$\mathsf{Enc}(pk_1, pk_2, m_1 \Vert m_2)$	$\mathsf{Dec}(sk_2, sk_1, c_1 \Vert c_2)$
$(pk_1, sk_1) \leftarrow \mathsf{KGen}_1$;	$(pk_2, sk_2) \leftarrow \mathsf{KGen}_1$	$c_1 = \mathsf{Enc}_1(pk_1, m_1; r_1)$	$m_1 = \mathsf{Dec}_1(sk_1, c_1)$
		$c_2 = \mathsf{Enc}_1(pk_2, m_2; r_2)$	$m_2 = \mathsf{Dec}_1(sk_2, c_2)$

Fig. 7. 2-key PKE $\mathsf{2PKE} = (\mathsf{KGen1}, \mathsf{KGen2}, \mathsf{Enc}, \mathsf{Dec})$ from 1-key PKE $\mathsf{PKE} := (\mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$.

6 Conclusion

In this work, we show that, with a slight modification, SIAKE is secure in the quantum random oracle model. Our work also proves the QROM security of modified X3LH, which introduces new construction from commutative supersingular isogenies.

References

1. Ambainis, A., Rosmanis, A., Unruh, D.: Quantum attacks on classical proof systems. In FOCS 2014, pp. 474-483.
2. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semiclassical oracles. Cryptology ePrint Archive, Report 2018/904
3. Bernstein D. J.: Multi-user Schnorr security, revisited. https://eprint.iacr.org/2015/996
4. Bernstein, D. J. Re: [pqc-forum] https://groups.google.com/a/list.nist.gov/forum/#!original/pqc-forum/svm1kDy6c54/0gFOLitbAgAJ
5. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In ASIACRYPT 2011, pp. 41-69.
6. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Stehlé D.: CRYSTALS - Kyber: a CCA-secure Module-lattice-based KEM. In 2018 S&P, pp. 353-367.
7. Boneh, D., Glass, D., Krashen, D., Lauter, K., Sharif, S., Silverberg, A., Tibouchi, M., Zhandry, M.: Multiparty Non-Interactive Key Exchange and More From Isogenies on Elliptic Curves. J. Math. Cryptol. 14(1): 5-14 (2020)
8. Bindel, N., Hamburg, M., Hülsing, A., Persichetti, E.: Tighter proofs of CCA security in the quantum random oracle model, In TCC 2019, pp. 61-90.

9. Bernstein, D. J., Lange, T., Martindale, C., Panny, L.: Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. In EUROCRYPT 2019, pp. 409-441.
10. Bernstein, D. J., Hamburg M., Re: [pqc-forum] NIST pqc-forum mailing list, 2018, https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/SrFO_vK3xbI/m/utjUZ9hJDwAJ
11. Bonnetain, X., Schrottenloher, A.: Quantum Security Analysis of CSIDH. In EUROCRYPT 2020, pp. 493-522.
12. Cohn-Gordon, K., Cremers, C., Gjøsteen, K., Jacobsen, H., Jager, T.: Highly Efficient Key Exchange Protocols with Optimal Tightness. In, CRYPTO 2019, pp. 767-797
13. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In EUROCRYPT 2001, pp. 453-474.
14. Cremers, C.J.F.: Formally and Practically Relating the CK, CK-HMQV, and eCK Security Models for Authenticated Key Exchange. Cryptology ePrint Archive, Report 2009/253
15. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In ASIACRYPT 2018, pp. 395-427.
16. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology, 8(3), 209-247 (2014).
17. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly Secure Authenticated Key Exchange from Factoring Codes and Lattices. In PKC 2012, pp. 467-484.
18. Fujioka A., Suzuki K., Xagawa K., Yoneyama K.: Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In AsiaCCS 2013, pp. 83-94.
19. Fujioka, A., Takashima, K., Terada, S., Yoneyama, K.: Supersingular Isogeny Diffie-Hellman Authenticated Key Exchange. In ICISC 2018, pp. 177-195.
20. Galbraith, S. D.: Authenticated key exchange for SIDH. Cryptology ePrint Archive, Report 2018/266
21. Galbraith, S. D., Petit, C., Shani, B., Ti, Y. B.: On the security of supersingular isogeny cryptosystems. In ASIACRYPT 2016, pp. 63-91.
22. Hofheinz, D., Hövelmanns, K., and Kiltz, E.: A Modular Analysis of the Fujisaki-Okamoto Transformation. In TCC 2017, pp. 341-371.
23. Hövelmanns, K., Kiltz, E., Schäge, S. and Unruh, D.: Generic Authenticated Key Exchange in the Quantum Random Oracle Model. In PKC 2020, pp. 389-422.
24. Jao, D., Azarderakhsh, R., Campagna, M., et al.: SIKE candidate for NIST. https://sike.org/
25. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: IND-CCA-Secure Key Encapsulation Mechanism in the Quantum Random Oracle Model, Revisited. In CRYPTO 2018, pp. 96-125.
26. Jiang, H., Zhang, Z., Ma, Z.: On the non-tightness of measurement-based reductions for key encapsulation mechanism in the quantum random oracle model. Cryptology ePrint Archive, Report 2019/494
27. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In CRYPTO 2005, pp. 546-566.
28. de Kock, B., Gjøsteen, K., Veroni, M.: Practical isogeny-based key-exchange with optimal tightness. In: SAC 2020 (2020), to appear.
29. Kirkwood, D., Lackey, B.C., McVey, J., Motley, M., Solinas, J.A., Tuller, D.: Failure is not an option: Standardization issues for post-quantum key agreement, 2015.
30. Kuchta, V., Sakzad, A., Stehle, D, Steinfeld, R., Sun, S.: Measure-Rewind-Measure: Tighter Quantum Random Oracle Model Proofs for One-Way to Hiding and CCA Security. In EUROCRYPT 2020, pp. 703-728.
31. Kawashima, T., Takashima, K., Aikawa, Y., Takagi, T.: An efficient authenticated key exchange from random self-reducibility on CSIDH. eprint 2020/1178
32. LeGrow, J., Jao, D., Azarderakhsh, J.: Modeling Quantum-Safe Authenticated Key Establishment, and an Isogeny-Based Protocol. Cryptology ePrint Archive, Report 2018/282
33. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In ProvSec 2007, pp. 1-16.
34. Longa, P.: A Note on Post-Quantum Authenticated Key Exchange from Supersingular Isogenies. Cryptology ePrint Archive, Report 2018/267.
35. NIST Post-Quantum Cryptography Standardization, https://csrc.nist.gov/Projects/Post-Quantum-Cryptography
36. Peikert, C.: Lattice Cryptography for the Internet. In PQCrypto 2014, pp. 197-219.
37. Peikert, C.: He gives C-sieves on the CSIDH, In EUROCRYPT 2020, pp. 463-492.

38. Saito, T., Xagawa, K., Yamakawa, T.: Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model. In EUROCRYPT (3) 2018, pp. 520-551
39. Targhi, E.E., Unruh, D.: Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In TCC 2016-B, pp. 192-216
40. Unruh, D.: Revocable quantum timed-release encryption. Journal of the ACM, 62(6):No.49, 2015. A preliminary version appeared in EUROCRYPT 2014.
41. Urbanik, D., Jao, D.: SoK: The Problem Landscape of SIDH. APKC@AsiaCCS 2018, pp. 53-60
42. Xiao, Y., Zhang, R., Ma, H.: Modular Design of Role-Symmetric Authenticated Key Exchange Protocols. ASIACRYPT (4) 2021: 742-772
43. Xue, H., Au, M., Yang, R., Liang, B., Jiang, H.: Understanding and Constructing AKE via Double-Key Key Encapsulation Mechanism. Cryptology ePrint Archive, Report 2018/904, version 20210527:111835
44. Xue, H., Li, B., Lu, X., Liang, B., He, J.: Understanding and Constructing AKE via Double-Key Key Encapsulation Mechanism. In ASIACRYPT 2018, pp. 158-189.
45. Xu, X., Xue, H., Wang, K., Au, M., Tian, S.: Strongly Secure Authenticated Key Exchange from Supersingular Isogenies. In ASIACRYPT 2019, pp. 278-308.
46. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In CRYPTO 2012, pp. 758-775
47. Zhandry, M.: How to record quantum queries, and applications to quantum indifferentiability. In CRYPTO 2019, pp. 239-268.

Appendix A FO transformation in QROM

Properties and requirements of existing transformations from probabilistic PKE to CCA secure KEM are summarized in Table 5.

In QROM (and also classical ROM), the main challenges include how to simulate the decapsulation oracle without a secret key, and how to argue the randomness of the encapsulated key in the challenge ciphertext.

To simulate the decapsulation oracle without a secret key in QROM, the additional hash [1,22] has high overhead and reduction loss, while injective map and private RO have low overhead and tight reduction. Boneh et al. [5] firstly introduced the technique of modeling $h(m)$ with $h_q \circ f(m)$ to provide decapsulation oracle, where h_q is a private RO and f is an *injective* function. Inspired by [5], Saito et al. [38] and Jiang et al. [25] independently assumed $\mathsf{Enc}(pk, \cdot; G(\cdot))$ is injective and modeled $h(m)$ as $h_q \circ \mathsf{Enc}(pk, m; G(m))$. Thus, $K = h(m)$ encapsulated in C can be computed as $h_q(C)$. We call this technique as injective mapping with encryption under fixed public key and illustrate it in Fig. 2.

With this decapsulation oracle using injective mapping with encryption under a fixed public key, several techniques are used to argue the randomness of the encapsulated key in challenge ciphertext. Saito et al. [38] introduced the "puncture" technique which relies on the IND-CPA security. Jiang et al. [25] applied their extended OW2H lemma to $G \times h$ as a unified oracle, thereby reducing the security to a weaker assumption, OW-CPA secure PKE. Recently, Hövelmanns et al. [23] extended the "puncture" analysis of [38] by considering decryption error.

Schemes	Inj. map.	prob. PKE	Additional Hash	Security Bound	DecError
[39,22]	-	IND/OW-CPA	len.-pre	$q^{3/2}\sqrt[4]{\varepsilon}$	✓
[38]	✓	IND-CPA	×	$q\sqrt{\varepsilon}$	×
[23]	✓	IND-CPA	×	$q\sqrt{\varepsilon}$	✓
[25]	✓	OW-CPA	×	$q\sqrt{\varepsilon}$	✓
[2]	✓	IND-CPA	×	$\sqrt{q\varepsilon}$	✓
[8]	✓	IND-CPA	×	$\sqrt{q\varepsilon}$	✓
[30]	✓	IND-CPA	×	$q^2\varepsilon$	✓

Table 5. Comparison of proof for FO type probabilistic PKE-to-KEM transform in the QROM. Inj. map. indicates the injective mapping with encryption under fixed public key. len.-pre means additional hash should be length preserving. q is the number of random oracle queries. ε is the advantage against OW/IND-CPA security of PKE.

Ambainis et al. [2] proposed an improved OW2H lemma, namely, the semi-classical OW2H, which implies the extended OW2H in [25] and gives better security bounds in several PKEs. Bindel, et al. [8] proposed a double-sided OW2H lemma and reduced the q factor from reduction loss. Recently, Kuchta et al. [30] bypass the square-root advantage loss using an updated double-sided OW2H lemma with the rewinding technique.

After the proposal of the OW2H lemma, as mentioned above, several variants are proposed. We summarize them in Table 6. S is the set that two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$ differ. The 'Must know' column shows the oracles available to the one-wayness attacker. 1_S refers to the indicator function of S. The one-wayness attacker outputs an element in S with probability ϵ. The lemma shows an upper bound of the difference between $\mathcal{A}^{\mathcal{O}_1}$ and $\mathcal{A}^{\mathcal{O}_1}$ as a function of ϵ or q, the number of queries to oracle.

OW2H Variants	$\lvert S\rvert$	Must know	Bound
Original [40,2]	Arbitrary	$\mathcal{O}_1$ or $\mathcal{O}_2$	$q\sqrt{\epsilon}$
Semi-classical [2]	Arbitrary	($\mathcal{O}_1$ or $\mathcal{O}_2$) and 1_S	$\sqrt{q\epsilon}$
Double-side [8]	1	$\mathcal{O}_1$ and $\mathcal{O}_2$	$\sqrt{\epsilon}$
Double-side-Revisited [30]	Arbitrary	$\mathcal{O}_1$ and $\mathcal{O}_2$	$q\epsilon$

Table 6. Comparison of OW2H lemmas.

Appendix B Discussions on Security Models: CK, CK$_{HMQV}$, eCK and CK$^+$

To achieve a stronger security after the CK model [13] was introduced, CK$_{HMQV}$ [27], eCK [33], and CK$^+$ [17] models are being proposed. Due to subtle but crucial differences between them, these models are incomparable. Cremers [14] formally analyzed the relation of CK [13], eCK [33], and CK$_{HMQV}$ [27]. We give a brief discussion here. For formal details, please refer to [14] and [42].

Matching session: A session s could be defined with s_A (the owner of s), s_B (intended peer), s_R (the role performed by s_A), s_{send}/s_{recv} (the message send/received by s_A), and/or s_{sid} (the session identifier, only used in CK model). In CK model, two sessions s and s' match if $s_A = s'_B$, $s_B = s'_A$ and $s_{sid} = s'_{sid}$. In CK$_{HMQV}$, if $s_A = s'_B$, $s_B = s'_A$, $s_{send} = s'_{recv}$, and $s_{recv} = s'_{send}$. s and s' match. For eCK, s and s' are matching sessions if they match in CK$_{HMQV}$ and $s_R \neq s'_R$. CK$^+$ claims to reformulate CK$_{HMQV}$, but uses the definition of the matching session in eCK. The subtle differences are crucial since the role-symmetric AKE that is secure in one model may be insecure in another model [14].

Session state reveal vs ephemeral key reveal: The CK model [13] allows the session state reveal by an adversary, but leaves the AKE protocol to specify the contents of the session state. Depending on the content of the session state reveal, a weaker AKE may be proved secure [33]. Thus, eCK replaces session state reveal with the ephemeral key reveal, which is equivalent to specifying the content of the session state as the ephemeral secret key. However, if more session state is allowed to be revealed, the eCK secure scheme may be insecure [14].

How and when the ephemeral/static secret key related to test session is given to the adversary. In the CK model, the compromise of the static secret key of the test session's owner is not allowed before the test session expires, thus not detecting key compromise impersonation (KCI) attack. To capture KCI, CK$_{HMQV}$, eCK and CK$^+$ allow the compromise of the static secret key of the owner before the test session ends. CK$_{HMQV}$, eCK and CK$^+$ also consider the weak version of perfect forward security (wPFS), i.e., the corruption of executor or peer is allowed after the end of the test session only if the matching session exists. The exposure of ephemeral/static secret key related to test session in eCK is modeled by adaptive queries to long-term key reveal/ephemeral key reveal. In CK$^+$ (and CK$_{HMQV}$), the exposure is not modeled by such queries. The ephemeral secret key is given to the adversary directly after it is generated (thus they actually allow adaptive queries to ephemeral key reveal); the exposure of the static secret key is modeled as given to the adversary directly when it is allowed.

Our Choice: We use the CK$^+$ model here with a more strict definition. We require the session state reveal at least include ephemeral secret key. The freshness still forbid the session state to reveal on the test session and its matching session, while the exposure of static or ephemeral secret key related to test session is allowed to capture KCI, wPFS and maximal exposure attack (MEX). To capture KCI, the static secret key of the owner of the

test session is given to the adversary directly after it is determined; for wPFS, the static secret keys of the owner and its peer in the test session are given to the adversary at the end of the test session; for maximal exposure attack (MEX), the ephemeral secret key is given to the adversary after it is determined.

Appendix C The proof for Lemma 6

We only prove the $[\mathsf{OW\text{-}CPA}, \cdot]$ security of $\mathsf{2PKE}_{csidh}$ under CSI-DDH assumption. It is analogous for its $[\cdot, \mathsf{OW\text{-}CPA}]$ security and the $\mathsf{OW\text{-}CPA}$ security of 1-key PKE.

The proves proceeds with a sequence of games. In Game i ($i \geq 1$), we denote the event of adversary's success as Succ_i

Game 0: This is the original $[\mathsf{OW\text{-}CPA}, \cdot]$ challenge game in Fig. 3. Thus, we have $\Pr[\mathsf{Succ}_1] = \mathrm{Adv}^{[\mathsf{OW\text{-}CPA},\cdot]}_{\mathsf{2PKE}_{\mathsf{csidh}}}(\mathcal{A})$.

Game 1: In this game we modify $[\mathsf{OW\text{-}CPA}, \cdot]$ challenge game, and state that the adversary wins only when $m'_1 = m_1$. We denote this event as Succ_1. Note that in Game 0, the adversary wins only if both $m'_1 = m_1$ and $m'_2 = m_2$. Thus, we have $\Pr[\mathsf{Succ}_0] \leq \Pr[\mathsf{Succ}_1]$.

Game 2: In this game, we modify the generation of challenge ciphertext. Specifically, $[\mathfrak{b}]pk_1$ is replaced by a random $[\mathfrak{c}]E_0$. We construct an algorithm $\mathcal{B}$ to solve the CSI-DDH problem given an instance $(E_0, [\mathfrak{a}]E_0, [\mathfrak{a}]E_0, E')$, if there exists an algorithm $\mathcal{A}$ to distinguish Game 1 and Game 2.

$\mathcal{B}(E_0, E_A, E_B, E')$
1 $pk_1 \leftarrow E_A$
2 $pk_2^*, \mathrm{state} \leftarrow \mathcal{A}(pk_1)$
3 $m_1 \leftarrow \{0,1\}^n$
4 $c_1^* = E_B$, $c_2^* = h(\mathsf{Coef}(E')) \oplus m_1$, $c_3^* \leftarrow \{0,1\}^n$
5 $m'_1 || m'_2 \leftarrow \mathcal{A}(\mathrm{state}, (c_1^*, c_2^*, c_3^*))$
6 If $m'_1 = m_1$, $b' = 1$, else $b' \leftarrow \{0,1\}$.

It is easy to check that if (E_0, E_A, E_B, E') is a CSI-DDH tuple, $\mathcal{B}$ perfectly simulates Game 1, else $\mathcal{B}$ perfectly simulates Game 2. Thus, we have

$$\begin{aligned}
\mathrm{Adv}^{csiddh}_{\mathcal{B}} &= \Pr[b = b'] - 1/2 \\
&= 1/2(\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]) \\
&= 1/2(\Pr[b' = 1 | \text{Game 1}] - \Pr[b' = 1 | \text{Game 2}]) \\
&= 1/2(\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]).
\end{aligned}$$

Game 3: In this game, we modify the computation of the challenge ciphertext. Specifically, $\mathsf{H}(\mathsf{Coef}([\mathfrak{c}]E_0))$ is replaced by a random string H^*. Now c_2^* is a completely random string in $\{0,1\}^n$. Thus, the advantage to compute m_1 is $\Pr[\mathsf{Succ}_3] = 1/2^n$. Note that, since H is a pair-wise independent hash function, by the leftover hash lemma, $\epsilon = |\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$ is negligible.

To sum up, we have that $\Pr[\mathsf{Succ}_0] \leq 2\mathrm{Adv}^{csiddh}_{\mathcal{B}} + 1/2^n + \epsilon$.

Appendix D Security proof for Modified X3LH

The proof for Theorem 2 proceeds almost the same as that for Theorem 1. The deference is that we should take decryption failure into account, which will need the following lemma.

Lemma 7 (Generic Distinguishing Problem, [23]). *Let X be a finite set, and $F : X \to \{0,1\}$ be a quantum accessible oracle. Let B_{λ_x} be a Bernoulli distribution that depends on*

$x \in X$, that is, for each x, $\Pr[F(x) = 1] = \lambda_x$. Let λ be the upper bound of λ_x for every $x \in X$. For any unbounded quantum algorithm $\mathcal{A}$ issuing at most q quantum queries,

$$\left|\Pr[\mathcal{A}^F() \to 1 | F(x) \leftarrow B_{\lambda_x}] - \Pr[\mathcal{A}^F() \to 1 | F(x) = 0]\right| \leq 8(q+1)^2\lambda.$$

Games		h (for encapsulated key) h_1 (for encapsulated key) G/G_1 (for randomness)	$K$of valid C **Justification** K of invalid C	R_B^* m_B^*/K^* SK^* (session key)
Games 0-1	I	$h(pk_1, pk_2, m)$	$h(pk_A, pk_{2,A}^j, m)/h_1(pk_A, m_1)$	$G(pk_{2A}^*, m_B^*)$
	II	$h_1(pk_1, m_1)$	– – – – –	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h'(pk_A, pk_{2,A}^j, s_A, C)/h'_1(pk_A, s_A, C)$	$H(K^*, \cdots)$
Game 1.5	I	$h(pk_1, pk_2, m)$	$h(pk_A, pk_{2,A}^j, m)/h_1(pk_A, m_1)$	$G(pk_{2A}^*, m_B^*)$
	II	$h_1(pk_1, m_1)$	**Lemma 8**	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$\tilde{G}(pk_2, m)/\tilde{G}_1(pk_1, m_1)$	$h'(pk_A, pk_{2,A}^j, s_A, C)/h'_1(pk_A, s_A, C)$	$H(K^*, \cdots)$
Game 2	I	$h(pk_1, pk_2, m)$	$h(pk_A, pk_{2,A}^j, m)/h_1(pk_A, m_1)$	$G(pk_{2A}^*, m_B^*)$
	II	$h_1(pk_1, m_1)$	**Lemma 3**	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$\tilde{G}(pk_2, m)/\tilde{G}_1(pk_1, m_1)$	$h'_q(pk_A, pk_{2,A}^j, C)/h'_{1q}(pk_A, C)$	$H(K^*, \cdots)$
Game 3	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$G(pk_{2A}^*, m_B^*)$
	II	h_q^3 or h_q^4	**Conceptual**	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$\tilde{G}(pk_2, m)/\tilde{G}_1(pk_1, m_1)$	$h'_q(pk_A, pk_{2,A}^j, C)/h'_{1q}(pk_A, C)$	$H(K^*, \cdots)$
Game 4	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$G(pk_{2A}^*, m_B^*)$
	II	h_q^3 or h_q^4	**Conceptual**	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$\tilde{G}(pk_2, m)/\tilde{G}_1(pk_1 m_1)$	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$H(K^*, \cdots)$
Game 4.5	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$G(pk_{2A}^*, m_B^*)$
	II	h_q^3 or h_q^4	**Lemma 8**	$f(s'_B, r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$H(K^*, \cdots)$
Game 5	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$G(pk_{2A}^*, m_B^*)$
	II	h_q^3 or h_q^4	**Lemma 4**	$f_r(r_B)/h(pk_A, pk_{2A}^*, m_B^*)$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$H(K^*, \cdots)$
Game 6	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$R_B^* \leftarrow \mathcal{R}$
	II	h_q^3 or h_q^4	**Lemma 5/OW2H**	$f_r(r_B)/K^* \leftarrow \{0,1\}^n$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$H(K^*, \cdots)$
Game 7	I	h_q^1 or h_q^2	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$R_B^* \leftarrow \mathcal{R}$
	II	h_q^3 or h_q^4	**Lemma 2**	$f_r(r_B)/K^* \leftarrow \{0,1\}^n$
	III	$G(pk_2, m)/G_1(pk_1, m_1)$	$h_q^1(pk_A, pk_{2,A}^j, C)/h_q^3(pk_A, C)$	$SK^* \leftarrow \{0,1\}^n$

Table 7. Overview of games for the proof of Theorem 2 w.r.t case E_3. Some details are not shown in this table, such as building lists, the guess of test session, the abort events, and some replacements of random oracles. Please refer to the Games for details. "valid C" and "invalid C" are those ciphertexts received by U_A, the owner of test session. m_B^*, R_B^*, K^* indicate the message, randomness, encapsulated key corresponding to the ciphertext in test session. SK^* is the session key of test session.

We also only present case E_3. All the other cases could be similarly extended. The game sequence is illustrated in Table 7, which is essentially adding Game 1.5 between Game 1 and 2 of Table 3 and adding Game 4.5 between Game 4 and 5 Table 3. Game 1.5 and Game 4.5 are used to eliminate and switch back "bad randomness" (which would introduce decryption failure), respectively. All the other games run as the same as those in Table 3. Thus, we only present Games 1.5 and 4.5.

Game 1.5. In this game, we impose a requirement that no decryption failure for Enc (resp. Enc_1) will occur with respect to public key pairs from $L_1 \times L_2$ (resp. L_1). The random oracle G (resp. G_1) is replaced with $\tilde{G}$ (resp. $\tilde{G}_1$) that only samples good randomness (which will be defined later) for all public keys in $L_1 \times L_2$ (resp. L_1). We say that two key-pairs belongs to $L_1 \times L_2$ only when they share the same user index.

For any fixed public key pairs $\left[(pk_{1,i}, sk_{1,i}), (pk^j_{2,i}, sk^j_{2,i})\right] \in L_1 \times L_2$, $pk_2 \in \mathcal{D}_2$, and $m \in \{0,1\}^n$, define $\mathcal{R}_{\mathsf{bad}}(i, j; pk_2, m)$ as

$$\begin{cases} \{r \in \mathcal{R} | \mathsf{Dec}(sk_{1,i}, sk^j_{2,i}, \mathsf{Enc}(pk_{1,i}, pk^j_{2,i}, m; r)) \neq m\} & \text{if } pk_2 = pk^j_{2,i} \\ \emptyset & \text{o.w.} \end{cases}$$

For fixed L_1 and L_2, let $\mathcal{R}_{\mathsf{bad}}(L_1, L_2; pk_2, m) := \cup_{i \in [1,N], j \in [1,2Nl]} \mathcal{R}_{\mathsf{bad}}(i, j; pk_2, m)$ be the set of bad randomness for the encryption Enc, and let the set of good randomness be $\mathcal{R}_{\mathsf{good}}(L_1, L_2; pk_2, m) := \mathcal{R} \setminus \mathcal{R}_{\mathsf{bad}}(L_1, L_2; pk_2, m)$.

For a fixed public key pk_1, and $m_1 \in \{0,1\}^{n_1}$, define $\mathcal{R}_{1\mathsf{bad}}(pk_1, m_1)$ as

$$\begin{cases} \{r_1 \in \mathcal{R}_1 | \mathsf{Dec}_1(sk_{1,i}, \mathsf{Enc}_1(pk_{1,i}, m_1; r_1)) \neq m_1\} & \text{if } \exists i \text{ s.t. } pk_1 = pk_{1,i} \\ \emptyset & \text{o.w.} \end{cases}$$

For a fixed L_1 and m_1, denote $\mathcal{R}_{1\mathsf{bad}}(pk_1, m_1)$ as the set of bad randomness for Enc_1 and $\mathcal{R}_{1\mathsf{good}}(pk_1, m_1) = \mathcal{R}_1 \setminus \mathcal{R}_{1\mathsf{bad}}(pk_1, m_1)$ as the set of good randomness.

Concretely, we choose internal $2(q_G + q_{G_1} + 2q)$-wise independent random functions g_q and g_{1q}. On receiving $pk_2 \times m \in \mathcal{D}_2 \times \{0,1\}^n$, $\tilde{G}$ samples and outputs an element from set $\mathcal{R}_{\mathsf{good}}(L_1, L_2; pk_2, m)$ using randomness $g_q(pk_2, m)$. On input $pk_1 \times m_1 \in \mathcal{D}_1 \times \{0,1\}^{n_1}$, $\tilde{G}_1$ samples and outputs an element from set $\mathcal{R}_{1\mathsf{good}}(pk_1, m_1)$ using randomness $g_{1q}(pk_1, m_1)$.

Note that from game 1.5 to 4.5, since the set of good randomness should be identified, simulating $\tilde{G}$ and $\tilde{G}_1$ requires unbounded power, which implies that the simulator is an unbounded algorithm. It still makes sense because the differences between these games are analyzed from the information-theoretical perspective.

Lemma 8. $\mathit{Adv}_1 - \mathit{Adv}_{1.5} \leq 16(q_G + 2q)^2\delta_2 + 16(q_{G_1} + 2q)^2\delta_1$.

Game 4.5. We switch back to G (resp. G_1) from $\tilde{G}$ (resp. $\tilde{G}_1$). The argument is similar to that in Game 1.5. And we have $\mathsf{Adv}_4 - \mathsf{Adv}_{4.5} \leq 16(q_G + 2q)^2\delta_2 + 16(q_{G_1} + 2q)^2\delta_1$.

Proof of Lemma 8: We first define an internal Game 1-1 in which only G is replaced.

By the definition of $\mathcal{R}_{\mathsf{bad}}(i, j; pk_2, m)$ in Game 1.5, when $pk_2 \in L_2$, there exists i^*, j^* such that $\mathcal{R}_{\mathsf{bad}}(i, j; pk_2, m)$ is non-empty only when $i = i^*$ and $j = j^*$; when $pk_2 \notin L_2$, $\mathcal{R}_{\mathsf{bad}}(i, j; pk_2, m)$ is always empty.

Define $\delta(i, j; pk_2, m) = \frac{|\mathcal{R}_{\mathsf{bad}}(i,j;pk_2,m)|}{|\mathcal{R}|}$ and $\delta(i, j) = \max_{m \in \{0,1\}^n} \delta(i, j; pk_2, m)$. By the definition of correctness, $\mathrm{E}(\delta(i, j)) = \delta_2$ or 0, depending on $pk_2 \in L_2$ or not, where the expectation is taken over $(pk_{1,i}, sk_{1,i}) \leftarrow \mathsf{KGen1}, (pk^j_{2,i}, sk^j_{2,i}) \leftarrow \mathsf{KGen2}$. Thus, $\exists\, i^*, j^*$, such that

$$\begin{aligned} \delta(L_1, L_2, pk_2, m) :=& \frac{|\mathcal{R}_{\mathsf{bad}}(L_1, L_2; pk_2, m)|}{|\mathcal{R}|} \leq \sum_{i \in [1,N], j \in [1,2Nl]} \frac{|\mathcal{R}_{\mathsf{bad}}(i, j; pk_2, m)|}{|\mathcal{R}|} \\ &= \frac{|\mathcal{R}_{\mathsf{bad}}(i^*, j^*; pk_2, m)|}{|\mathcal{R}|} = \delta(i^*, j^*; pk_2, m). \end{aligned}$$

Let $\delta(L_1, L_2) := \max^{pk_2 \in L_2}_{m \in \{0,1\}^n} \delta(L_1, L_2; pk_2, m)$. By taking the expectation on $\delta(L_1, L_2)$ over the generation of L_1 and L_2, $\exists j^*$, such that

$$\mathrm{E}(\delta(L_1, L_2)) = \mathrm{E}\left(\max^{pk_2 \in L_2}_{m \in \{0,1\}^n} (\delta(L_1, L_2; pk_2, m))\right) \qquad \leq \mathrm{E}(\delta(i^*, j^*)) = \delta_2,$$

where the last expectation is taken over $(pk_{1,i^*}, sk_{1,i^*}) \leftarrow \mathsf{KGen1}$ and $(pk_{2,i^*}^{j^*}, sk_{2,i^*}^{j^*}) \leftarrow \mathsf{KGen2}$.

To give the upper bound of $\mathsf{Adv}_1 - \mathsf{Adv}_{1-1}$, we utilize the distinguisher between Game 1 and Game 1.5 for $b = 1$ and $b = 0$ together with Lemma 7 to construct an unbounded quantum adversary $\mathcal{B}^{|F\rangle}$ to solve the generic distinguishing problem.

$\mathcal{B}$, on input randomly chosen L_1, L_2, simulates the game as in Game 1. Let $\lambda(pk_2, m) = \delta(L_1, L_2; pk_2, m)$. Let $F(pk_2, m)$ be bernoulli-distributed $B_{\lambda(pk_2,m)}$ or always 0 with respect to the generic distinguishing problem. Define G as

$$G(pk_2, m) = \begin{cases} \mathsf{Sample}(\mathcal{R}_{\mathsf{good}}(L_1, L_2; pk_2, m); g_q(m)) & \text{if } \ F(pk_2, m) = 0 \\ \mathsf{Sample}(\mathcal{R}_{\mathsf{bad}}(L_1, L_2; pk_2, m); g_q(m)) & \text{o.w.} \end{cases}$$

where $\mathsf{Sample}(S; r)$ outputs an element from a set S with randomness r.

When $F(pk_2, m)$ is bernoulli-distributed according to $B_{\lambda(pk_2,m)}$, G is as in Game 1; when it is always 0, G is the same as in Game 1-1. At last, $\mathcal{B}$ just returns what $\mathcal{A}$ guesses.

For both $b = 1$ and 0, $\mathcal{B}^{|F\rangle}(L_1, L_2)$ perfectly simulates Game 1-1 or Game 1 for $\mathcal{A}$ corresponding to F is always 0 or bernoulli-distributed. By further applying Lemma 7 with $\lambda = \delta(L_1, L_2)$, we have

$$\begin{aligned} &|\Pr[b' = 1 | b = 1 \text{ (resp.0) in Game 1}] - \Pr[b' = 1 | b = 1 \text{ (resp.0) in Game 1-1}]| \\ &\quad = \Pr[\mathcal{B}^{|F\rangle}(L_1, L_2) \to 1 | F \leftarrow B_{\lambda(pk_2,m)}] - \Pr[\mathcal{B}^{|F\rangle}(L_1, L_2) \to 1 | F \equiv 0] \\ &\quad \leq 8 \cdot (q_G + 2q)^2 \delta(L_1, L_2). \end{aligned}$$

By taking the expectation over L_1 and L_2, we have $\mathsf{Adv}_1 - \mathsf{Adv}_{1-1} \leq 16 \cdot (q_G + 2q)^2 \delta_2$.

Now, we consider the replacement of G_1. Define $\delta(pk_1, m_1) = \frac{|\mathcal{R}_{1\mathsf{bad}}(pk_1, m_1)|}{|\mathcal{R}_1|}$. Then, $\delta_1 = \mathrm{E}(\max_{pk_1,m}(\delta(pk_1, m_1)))$.

By constructing a similar unbounded quantum adversary $\mathcal{B}^{|F\rangle}$, where $F(pk_1, m_1)$ is bernoulli-distrubution $B_{\delta(pk_1,m_1)}$ or always 0, with the similar analysis for the switch of G_1, we have $\mathsf{Adv}_{1-1} - \mathsf{Adv}_{1.5} \leq 16(q_{G_1} + 2q)^2 \delta_1$.