Strongly Secure Authenticated Key Exchange from Supersingular Isogenies*

Xiu Xu[1,2,4], Haiyang Xue[1,2,3] ✉, Kunpeng Wang[1,2,4], Man Ho Au[3], Bei Liang[5], Song Tian[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] Data Assurance and Communications Security Research Center, Beijing, China
[3] The Hong Kong Polytechnic University, Hong Kong
[4] School of Cyber Security, University of Chinese Academy of Sciences, China.
[5] Chalmers University of Technology, Gothenburg, Sweden
haiyangxc@gmail.com

Abstract. This paper aims to address the open problem, namely, to find new techniques to design and prove security of supersingular isogeny-based authenticated key exchange (AKE) protocols against the widest possible adversarial attacks, raised by Galbraith in 2018. Concretely, we present two AKEs based on a double-key PKE in the supersingular isogeny setting secure in the sense of CK^+, one of the strongest security models for AKE. Our contributions are summarised as follows. Firstly, we propose a strong $\mathsf{OW\text{-}CPA}$ secure PKE, $\mathsf{2PKE_{sidh}}$, based on SI-DDH assumption. By applying modified Fujisaki-Okamoto transformation, we obtain a $[\mathsf{OW\text{-}CCA}, \mathsf{OW\text{-}CPA}]$ secure KEM, $\mathsf{2KEM_{sidh}}$. Secondly, we propose a two-pass AKE, $\mathsf{SIAKE_2}$, based on SI-DDH assumption, using $\mathsf{2KEM_{sidh}}$ as a building block. Thirdly, we present a modified version of $\mathsf{2KEM_{sidh}}$ that is secure against leakage under the 1-Oracle SI-DH assumption. Using the modified $\mathsf{2KEM_{sidh}}$ as a building block, we then propose a three-pass AKE, $\mathsf{SIAKE_3}$, based on 1-Oracle SI-DH assumption. Finally, we prove that both $\mathsf{SIAKE_2}$ and $\mathsf{SIAKE_3}$ are CK^+ secure in the random oracle model and supports arbitrary registration. We also provide an implementation to illustrate the efficiency of our schemes. Our schemes compare favourably against existing isogeny-based AKEs. To the best of our knowledge, they are the first of its kind to offer security against arbitrary registration, wPFS, KCI and MEX simultaneously. Regarding efficiency, our schemes outperform existing schemes in terms of bandwidth as well as CPU cycle count.

Keywords: authenticated key exchange, key encapsulation mechanism, supersingular elliptic curve isogeny, post quantum

Version Notes

We have revised our paper, and some presentations in the current version are different from the previous versions.

- The 201808 Version: The original version.
- The 201810 Version: Add the FTTY work [FTTY18].
- The 201905 Version:

* This is the full version of a preliminary form as Strongly Secure Authenticated Key Exchange from Supersingular Isogenies in ASIACRYPT 2019

1. In the previous version, we utilize the modified chosen public key and chosen ciphertext security [Oka07] as the secure notion for our key encapsulation. In this version, we utilize the [OW-CCA, OW-CPA] security in [XLL+18] as the secure notion for our key encapsulation.
2. We utilize the crypto-friendly notions proposed by [FTTY18] to give a clearer and more convenient presentation.

– The 201909 Version: Fix some typos.

1 Introduction

Authenticated Key Exchange. Allowing two parties to agree on a common shared key over a public but possibly insecure channel, key exchange (KE) is a fundamental cryptographic primitive. Many studies have investigated how to achieve KE protocols that provide authentication [BR93,CK01,FSXY12,LLM07] and how to implement authenticated key exchange (AKE) with high efficiency [BCNP09,FSXY12,FSXY13,JKL04,LLM07,MQV95,MTI86] based on classical assumptions. Different of security models have been proposed, including BR model [BR93], CK model [CK01] and eCK model [LLM07]. Introduced in [Kra05] and reformulated by Fujioka et al. [FSXY12], currently, CK^+ security model is known as one of the 'strongest' and most 'desirable' security notions. The CK^+ model not only covers the security requirement in CK model, but also captures some advanced attacks such as the key compromise impersonation (KCI) attack, the maximal exposure (MEX) attack and the breaking of weak perfect forward secrecy (wPFS).

Supersingular Isogeny Diffie-Hellman Key Exchange (SIDH). Apart from lattice, code, hash and multivariate cryptography, supersingular elliptic curve isogeny is one of the most attractive candidates for post-quantum cryptography. The best-known protocol is Jao and De Feo's supersingular isogeny Diffie-Hellman key exchange (SIDH) [JD14] based on the hard problem of computing isogenies between supersingular elliptic curves. There are several interesting topics concerning SIDH in the literature. For example, computational efficiency [CLN16,FLO17,KAM18], key compression [CJL17], adaptive attacks on SIDH [GPST16], relationship of the underlying complexity problems [EHL18,GV17,UJ18], signature schemes [GPS16,JS14,STW12,YAJ17] and its standardization [JAC17,KLM15].

Recently, several work [FTTY18,Gal18,Lon18] have studied the important problem of designing AKE schemes from the basic SIDH primitive. As pointed out by Galbraith [Gal18], there are several challenges in adapting the security proof of existing well-designed AKE schemes (most of them are based on discrete logarithm assumption) to the SIDH case:

– Many AKE schemes based on discrete logarithm assumption, such as MQV [MQV95] and HMQV [Kra05], require a richer algebraic structure the supersingular isogeny does not possess.
– The protocols involving long-term/static secret keys are vulnerable to the adaptive attack [GPST16] aiming at the case where the static public key is used. More precisely, suppose that in a protocol Alice sets E_A as her static public key, and E_Y is an ephemeral public value sent by Bob. Galbraith et al. [GPST16] showed that adversary Bob can send (E_Y, R', S') with maliciously-crafted points R' and S' to gradually learn Alice's static secret key.
– The gap assumption that holds in the discrete logarithm setting is crucial for security proof. However, the gap assumption does not hold in the SIDH setting when polynomial queries are submitted to an *unlimited* decisional solver.

The State of the Art of SIDH AKE. Recently, there are many exciting results on the generic and non-generic constructions of AKE over supersingular curves [FTTY18,Gal18,Lon18]. Galbraith [Gal18] and Longa [Lon18] showed how to adapt the generic constructions of secure AKE from basic primitives like IND-CCA encryption/KEMs, MACs, PRFs etc, including the schemes proposed by Boyd, Cliff, Nieto and Paterson [BCNP09] (abbreviated as BCNP scheme), by Fujioka, Suzuki, Xagawa and Yoneyama [FSXY13] (abbreviated as FSXY scheme) and by Guilhem, Smart and Warinschi [GSW17] (abbreviated as GSW scheme), to the SIDH setting by inserting an IND-CCA secure KEM based on SIDH. Particularly, Longa [Lon18] showed how to use SIDH as basic building blocks to construct AKE schemes. However, these transformations lead either to more isogeny computations or increase in rounds of communication. The detailed analyses are examined and summarized in Table 1 of [Gal18]. Here we make a more concrete comparison among these AKE schemes in the SIDH setting in Table 1.

With respect to non-generic constructions, Galbraith proposed two SIDH-AKE protocols [Gal18], one of which is based on the Jeong-Katz-Lee [JKL04] scheme TS2 (we call it Gal 1) and another is an SIDH variant of NAXOS scheme (we call it Gal 2). Very recently, Fujioka et al. [FTTY18] gave two Diffie-Hellman like isogeny-based AKEs, which we denote as FTTY 1 where the session key is extracted from the combination of two Diffie-Hellman values, and FTTY 2 where the session key is extracted from four Diffie-Hellman values, respectively. Unfortunately, all of these schemes only provide security against adversaries with limited capabilities, such as wPFS security (details are given in section 1.3). Several known attacks are not taken into account, including arbitrary registrant for static public keys, the KCI attack, or the MEX attack. In an AKE system, the adversary-controlled parties may register arbitrary public keys and arbitrary registrant allows any party to register arbitrary public keys (even the same key with some other party) without any validity checks. In fact, neither Gal 1-2 nor FTTY 1-2 scheme allows the arbitrary registrant for the static public key. Otherwise, with malicious static public keys, a target secret key can be learned bit by bit, which implies that Gal 1-2 and FTTY 1-2 are not resistant to the adaptive attack. Moreover, Gal 1 is not resistant to the KCI attack and Gal 2 is not resistant to the MEX attack. Detailed analyses on those attacks against Gal 1-2 and FTTY 1-2 are given in the related works.

Thus, *"to find new techniques to design and prove security of AKE protocols in the SIDH setting, ... give a full analysis of AKE that includes the widest possible adversarial goals."*, a quote from Galbraith [Gal18], is the main problem to be addressed in the area of SIDH-based AKE. In this paper, we are motivated to address such an open problem.

1.1 Our Contributions

In this paper, we present two AKEs based on a double-key PKE in the SIDH setting and show that both of them allow arbitrary registrant and are CK^+ secure in the random oracle model. Our results are summarized as follows.

- We propose a strong $\mathsf{OW\text{-}CPA}$ secure PKE, $\mathsf{2PKE_{sidh}}$, based on SI-DDH assumption. The strong $\mathsf{OW\text{-}CPA}$ security is exactly the $[\mathsf{OW\text{-}CPA}, \cdot]$ security formalized in [XLL+18] which states that the PKE is still $\mathsf{OW\text{-}CPA}$ secure even if part of the public key is generated by the adversary. This construction may be of independent interest. Through the modified Fujisaki-Okamoto transformation [XLL+18], we obtain a $[\mathsf{OW\text{-}CCA}, \mathsf{OW\text{-}CPA}]$ secure KEM, $\mathsf{2KEM_{sidh}}$, to be used as the building block of our AKE.

– With $\mathsf{2KEM}_{\mathsf{sidh}}$ as the basic tool, we propose a two-pass AKE, SIAKE_2, based on SI-DDH assumption. SIAKE_2 is CK^+ secure in the random oracle model and supports arbitrary registration.
– We propose 1-Oracle SI-DH assumption, a strong version of the SI-DDH assumption. Contrary to its analogue, Oracle Diffie-Hellman problem [ABR01] in the discrete logarithm setting, the 1-Oracle SI-DH problem only allows one query to the oracle. We revisit $\mathsf{2PKE}_{\mathsf{sidh}}$ and provide a modified version of $\mathsf{2KEM}_{\mathsf{sidh}}$, and show that under the 1-Oracle SI-DH assumption both of them are still secure against leakage.
– Using the modified $\mathsf{2KEM}_{\mathsf{sidh}}$ as the basic tool, we propose a three-pass AKE, SIAKE_3, based on 1-Oracle SI-DH assumption. We prove that it supports arbitrary registration and is also CK^+ secure in the random oracle model.

From Table 1, we can observe that both SIAKE_2 and SIAKE_3 achieve the security against multiple possible adversaries, which to the best of our knowledge covers the most extensive adversarial goals, including arbitrary registrant, wPFS, KCI and MEX.

Scheme	Key Reg.	Assum.	Model	wPFS	KCI	MEX	Rd	Init isog	Resp isog	Mess Size
Gal 1 [Gal18]	Honest	SI-CDH	CK	✓	×	×	2	3	3	$108n$
Gal 2 [Gal18]	Honest	SI-CDH	BR	✓	✓	×	2	4	4	$108n$
FTTY 1 [FTTY18]	Honest	SI-DDH	CK	✓	×	×	1	3	3	$72n$
FTTY 2 [FTTY18]	Honest	di-SI-DDH	CK^+	✓	✓	✓	1	5	5	$72n$
GSW [GSW17]	Arbi.	SI-DDH	CK	✓	×	×	3	6	6	$186n$
BCNP [BCNP09,Lon18]	Arbi.	SI-DDH	CK	✓	✓	×	2	6	6	$148n$
FSXY [FSXY13,Lon18]	Arbi.	SI-DDH	CK^+	✓	✓	✓	2	6	6	$148n$
SIAKE_2	Arbi.	SI-DDH	CK^+	✓	✓	✓	2	6	5	$114n$
SIAKE_3	Arbi.	1-OSIDH	CK^+	✓	✓	✓	3	5	5	$80n$

Table 1. Comparison of existing AKE protocols on supersingular isogeny. **Key Reg.** represents registering the static public key. "Arbi" means arbitrary registrant is allowed while "Honest" means only honest registrants is allowed. **Assump.** is the abbreviation of assumptions. "1-OSIDH" is the abbreviation of 1-Oracle SI-DH assumption. **Rd** denotes the number of protocol's communication round. **Init isog** and **Resp isog** represent the number of isogeny computation that the initiator and responder have to perform respectively. **Mess Size** denotes the total message size. "✓" indicates that the scheme can resist this kind of attack while "×" indicates it cannot. n is the security parameter.

1.2 Technique Overview

Our core ideas and techniques are illustrated in Figure 1. Let E_0 be the starting curve, and $(P_1, Q_1), (P_2, Q_2)$ be the base points. E_{A_1}, E_{B_2}, E_X and E_Y are four intermediate curves which are part of static or ephemeral public keys. E_{A_1Y}, E_{XB_2} and E_{XY} are three final computing curves.

Let U_A, U_B be two parties in the AKEs. The SIDH works as follows: U_A chooses a secret, computes the isogeny $\phi_X : E_0 \to E_X$ with kernel G_X and publishes $X = (E_X, \phi_X(P_2), \phi_X(Q_2))$. U_B chooses a secret, computes the isogeny $\phi_Y : E_0 \to E_Y$ with kernel G_Y and publishes $Y = (E_Y, \phi_Y(P_1), \phi_Y(Q_1))$. They both can compute $E_{XY} \cong E_X/\phi_X(G_Y) \cong E_Y/\phi_Y(G_X)$.

The strategy to provide authentication with the static and ephemeral components is that every user registers a static public key such that U_A's static public key is $pk_{A_1} = (E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2))$ while U_B's static public key is $pk_{B_2} = (E_{B_2}, \phi_{B_2}(P_1), \phi_{B_2}(Q_1))$.

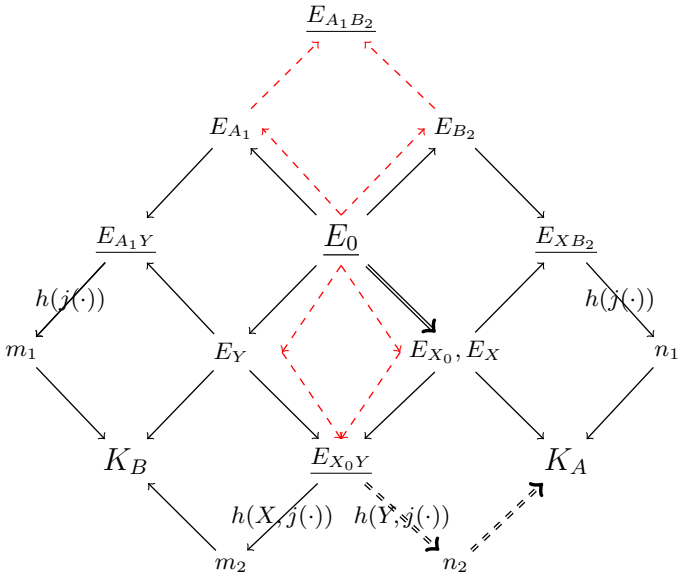

Fig. 1. Illustration of the core idea of SIAKE_2 and SIAKE_3. The red dashed lines illustrate the core ideas of Gal 1 scheme [Gal18]. In SIAKE_2, E_X and E_{X_0} are two independent curves. In SIAKE_3, $E_X = E_{X_0}$ and the dashed double arrow is included.

As shown in Figure 1, there is a natural way to extract a session key from four Diffie-Hellman values $E_{A_1B_2}$, E_{A_1Y}, E_{XB_2} and E_{XY} (Actually, this is what FTTY2 scheme does). However, it is risky to take $E_{A_1B_2}$ into account. Let us recall the adaptive attack from Galbraith, Petit, Shani and Ti [GPST16]. A malicious user U_B who registers his static public key E_{B_2} with specified points R', S', can learn one bit of the static secret key of U_A if he can also query the session key. As shown in Figure 1 with dashed lines, Galbraith [Gal18] involves $E_{A_1B_2}$ and E_{XY} for the session key. Under the adaptive attack [GPST16], adversary could gradually learn the static secret key by malicious registrations. Thus, $E_{A_1B_2}$ could not be included in the session key when arbitrary registrant is allowed.

Although now only E_{A_1Y}, E_{XB_2}, and E_{XY} are involved in the session key, the adaptive attack can still be launched if the CK$^+$ adversary (in case E_2 in Table 2) sends E_Y with specified points R', S' to U_A. With the ephemeral secret key for E_X and the session key, the adversary could still extract one bit of the static secret key. The problem can be tackled by a check of "validity" of $Y = (E_Y, R, S)$. Our solution is to employ the "re-encryption" technique used in Fujisaki-Okamoto (FO) transformation [FO99]. Precisely, $C = (Y, y_1, y_0)$ is the ciphertext under public key pk_{A_1} and X, where $Y = (E_0/\langle P_2 + [y]Q_2\rangle, \phi_Y(P_1), \phi_Y(Q_1))$, $y_1 = h(j(E_{A_1Y})) \oplus m_1$, $y_0 = h(j(E_{XY})) \oplus m_0$ and $y = G(m_1, m_0)$ for a hash function G, and the encapsulated key is $K_B = H(m_1, m_0, C)$. As a byproduct, we obtain the chosen ciphertext (CCA) secure KEM by the FO transformation and the "validity" of

$Y = (E_Y, R, S)$ can be checked by U_A using $y = G(m_1, m_0)$ so that the adaptive attack fails to work.

Now the CCA secure KEM with "re-encryption" avoids the adaptive attack, but it is still not sufficient for CK^+ security. The CK^+ adversary has the capability to adaptively *send* messages and adaptively query the *session state* and *session key* of non-test sessions. The capability of adaptively *sending* messages in the test session means that the adversary is allowed to choose one-part of the challenge public key X^* for (Y^*, y_1^*, y_0^*), while the capability of querying the *session state* and *session key* of non-test sessions implies that the adversary is also allowed to query the decapsulation oracle which decapsulates the ciphertext under several other public keys X'. This feature has been analyzed by [XLL+18] and formalized as $[\mathsf{OW\text{-}CCA}, \cdot]$ security. The modified Fujisaki-Okamoto [XLL+18] states that putting the public key in the hashing step when generating the encapsulated key would be sufficient. Precisely, K_B encapsulated in (Y, y_1, y_0) is $H(X, m_1, m_2, C)$.

The last challenge that we are facing is the relationship between X and Y, which leads to the difficulty in simulating the CK^+ game. In the test session, on the one hand X is part of the public key (pk_{A_1}, X) under which the ciphertext (Y, y_1, y_0) is computed. On the other hand X is part of the ciphertext (X, x_1, x_0) in which K_A is encapsulated under public key (pk_{B_2}, Y). Precisely, in the test session $X = ((E_X, R_2, S_2), x_1, x_0)$ is sent by AKE adversary $\mathcal{A}$, and the simulator $\mathcal{S}$ obtains challenge ciphertext (Y^*, y_1^*, y_0^*) from the $[\mathsf{OW\text{-}CCA}, \cdot]$ challenger (which means the secret y in Y^* is unknown). But to simulate the CK^+ game, especially to maintain the consistency of hash lists, $\mathcal{S}$ should learn $h(j(E_X/\langle R_2 + [y]S_2\rangle))$ to extract K_A encapsulated in (X, x_1, x_0).

We propose two solutions for this problem. One method is to add an extra X_0 such that X_0 is part of the public key (pk_{A_1}, X_0) under which the ciphertext (Y, y_1, y_0) is computed, while X is part of the ciphertext (X, x_1) under public key E_{B_2} (we omit Y). The other solution is to strengthen the underlying assumption as 1-Oracle SI-DH assumption such that $h(j(E_X/\langle R_2 + [y]S_2\rangle))$ could be leaked.

In consequence, the two solutions lead to two AKEs, namely, SIAKE_2 and SIAKE_3.

- Solution 1: We add an extra X_0 to take the position of X as part of the public key (pk_{A_1}, X_0) under which the ciphertext (Y, y_1, y_0) is computed, remove x_2 and set (X, x_1) as the ciphertext under public key E_{B_2} rather than (E_{B_2}, Y). Then the value of $h(j(E_X/\langle R_2 + [y]S_2\rangle))$ is not needed during the security proof. The drawback of this solution is that K'_A can not be included in the session state of U_B. Solution 1 leads to SIAKE_2.
- Solution 2: We strengthen the underlying SI-DDH assumption to the 1-Oracle SI-DH assumption to allow the leakage of $h(j(E_X/\langle R_2 + [y]S_2\rangle))$. The 1-Oracle SI-DH assumption can be considered as a hashed SI-DDH assumption where a one-time hashed SI-CDH oracle is allowed. Note that considering $\langle R_2 + [y]S_2\rangle = \langle [u]R_2 + [y][u]S_2\rangle$ for any integer $1 \leq u \leq \ell_2^{e_2}$ and coprime to ℓ_2, we employ a simple trick of tailoring the hash function as $h(Y, j(E_{XY}))$ in x_2 and $h(X, j(E_{XY}))$ in y_2. This solution results in SIAKE_3.

1.3 Related Works and Their Analysis

Galbraith [Gal18] proposed two SIDH variants of AKE, namely, Gal 1 from Jeong-Katz-Lee protocol [JKL04] and Gal 2 from NAXOS protocol [LLM07]. Considering the adaptive attack on static secret keys, Gal 1 protocol only allows honest registrants of static public

keys and it is also vulnerable to the KCI attack. So far, neither has there been any concrete MEX attack on Gal 1, nor any formal proofs to show Gal 1 is resistant to the MEX attack. Gal 2 protocol is provably secure in BR model, which only allows honest registrants of static public keys (if the adversary gets the ephemeral secret key, like x, the adaptive attack still works), and can not resist the MEX attack.

Very recently, Fujioka et al. [FTTY18] gave two Diffie-Hellman like isogeny-based AKEs, namely, FTTY 1 and FTTY 2. FTTY 1 protocol, which is quite similar to Gal 1 scheme, is CK secure in the quantum random oracle model, but it only allows honest registrants and cannot resist the KCI attack. FTTY 2 is secure in CK^+ model, but it also only allows honest registrants.

Below we illustrate in detail the (in)capability of Gal 1-2 and FTTY 1-2 on resisting the adaptive attacks (if the arbitrary registrant is allowed), the KCI attack, and the MEX attack.

Adaptive attacks if *arbitrary* registrant is allowed. Suppose that in a protocol Alice sets $E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2)$ as her static public key. The goal of a malicious adversary is to compute Alice's static secret key. As illustrated in Figure 1, the session key of Gal 1 is extracted from E_{XY} and $E_{A_1B_2}$. By applying the adaptive attacks [GPST16], a malicious adversary can register (E_{B_2}, R', S') with specified points R' and S', rather than $\phi_{B_2}(P_1)$ and $\phi_{B_2}(Q_1)$, as the static public key for Bob. By checking whether the session key computed by Alice (which can be obtained from SessionKeyReveal query) is equal to that computed by Bob, one bit of Alice's static secret key is determined. The adversary gradually learns Alice's static secret key by registering several valid static public keys according to adaptive attacks. Such an attack can be applied to FTTY 1 directly and it also works for FTTY 2 if the adversary also has the ephemeral secret key x of Alice (which can be obtained by querying SessionStateReveal), which means that FTTY 2 also does not allow arbitrary registrant. Gal 2 does not allow arbitrary registrant either, since if the adversary has the ephemeral secret key x of Alice (which can be obtained from SessionStateReveal query), by honestly registering static public key for Bob, then sending (E_Y, R', S') with specified points R' and S', and checking whether the session key computed by Alice is equal to that computed by Bob, the adversary is able to learn one bit of Alice's static secret key.

KCI Attacks. KCI attacks state that if a static secret key is revealed, an adversary can try to impersonate any other honest parties in order to fool the owners of the exposed secret keys. Neither Gal 1 nor FTTY 1 are resistant to the KCI attack since each session key is extracted from E_{XY} and $E_{A_1B_2}$, and by generating $E_Y, \phi_Y(P_1), \phi_Y(Q_1)$ and sending it to Alice on behalf of Bob, with Alice's static secret key the adversary could compute the session key even if Bob's static secret key is unknown.

MEX Attacks. In MEX, an adversary aims to distinguish the session key from a random value under the disclosure of the ephemeral secret key of (at least) one party of the test session. Gal 2 is not resistant to the MEX attack since its session key is extracted from E_{XY}, E_{XB_2}, and E_{A_1Y}, thus it is easy for an adversary to compute those curves with the ephemeral secret key corresponding to E_X and E_Y.

2 Preliminaries

2.1 SIDH and Crypto-friendly Description

We recall briefly the SIDH protocol using the same notation as [JD14,JAC17]. Let p be a large prime of the 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 $E_0[\ell_2^{e_2}]$. The public parameters are $(E_0; P_1, Q_1; P_2, Q_2; \ell_1, \ell_2, e_1, e_2)$.

$$\begin{array}{ccc} E_0 & \xrightarrow{\phi_A} & E_A = E_0/\langle R_A\rangle \\ \downarrow{\scriptstyle \phi_B} & & \downarrow{\scriptstyle \phi_{AB}} \\ E_B = E_0/\langle R_B\rangle & \xrightarrow{\phi_{BA}} & E_{AB} = E_0/\langle R_A, R_B\rangle \end{array}$$

Fig. 2. SIDH

The SIDH, as depicted in Figure 2, works as follows. Alice chooses her secret key k_a from $\mathbb{Z}_{\ell_1^{e_1}}$ and computes the isogeny $\phi_A : E_0 \to E_A$ whose kernel is the subgroup $\langle R_A\rangle = \langle P_1 + [k_a]Q_1\rangle$. She then sends to Bob her public key which is E_A together with the two points $\phi_A(P_2), \phi_A(Q_2)$. Similarly, Bob chooses his secret key k_b from $\mathbb{Z}_{\ell_2^{e_2}}$ and computes the isogeny $\phi_B : E_0 \to E_B$ with kernel subgroup $\langle R_B\rangle = \langle P_2 + [k_b]Q_2\rangle$. He sends to Alice his public key which is E_B together with the two points $\phi_B(P_1), \phi_B(Q_1)$. To get the shared secret, Alice computes the isogeny $\phi_{BA} : E_B \to E_{BA}$ with kernel subgroup generated by $\phi_B(P_1) + [k_a]\phi_B(Q_1)$. Similarly, Bob computes the isogeny $\phi_{AB} : E_A \to E_{AB}$ with kernel subgroup generated by $\phi_A(P_2) + [k_b]\phi_A(Q_2)$. Since the composed isogeny $\phi_{AB} \circ \phi_A$ has the same kernel $\langle R_A, R_B\rangle$ as $\phi_{BA} \circ \phi_B$, Alice and Bob can share the same j-invariant $j(E_{AB}) = j(E_{BA})$.

It will be helpful to have a crypto-friendly description of SIDH for the presentation of our AKEs. We follow the treatment of Fujioka et al. [FTTY18]. In what follows 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 $\mathfrak{a} = k_a$ and $\mathfrak{b} = k_b$, then we define,

$$\begin{aligned}
\mathfrak{g}^{\mathfrak{a}} &= (E_A; \phi_A(P_t), \phi_A(Q_t)) \in \mathrm{SSEC}_A, \\
&\qquad \text{where } R_A = P_s + [k_a]Q_s, \phi_A : E_0 \to E_A = E_0/\langle R_A \rangle; \\
\mathfrak{g}^{\mathfrak{b}} &= (E_B; \phi_B(P_s), \phi_B(Q_s)) \in \mathrm{SSEC}_B, \\
&\qquad \text{where } R_B = P_t + [k_b]Q_t, \phi_B : E_0 \to E_B = E_0/\langle R_B \rangle; \\
(\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}} &= j(E_{BA}), \text{where } R_{BA} = \phi_B(P_s) + [k_a]\phi_B(Q_s), \\
&\qquad \phi_{BA} : E_B \to E_{BA} = E_B/\langle R_{BA} \rangle; \\
(\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}} &= j(E_{AB}), \text{ where } R_{AB} = \phi_A(P_t) + [k_b]\phi_A(Q_t), \\
&\qquad \phi_{AB} : E_A \to E_{AB} = E_A/\langle R_{AB} \rangle.
\end{aligned}$$

Using this notation, the SIDH looks almost exactly like the classical Diffie-Hellman. That is, the public parameters are $\mathfrak{g}$ and $\mathfrak{e}$. Alice chooses a secret key $\mathfrak{a}$ and sends $\mathfrak{g}^{\mathfrak{a}}$ to Bob, while Bob chooses a secret key $\mathfrak{b}$ and sends $\mathfrak{g}^{\mathfrak{b}}$ to Alice. The shared key is, as we expect, $j = (\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}} = (\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}}$.

2.2 Standard SIDH Assumptions

We describe two standard assumptions about supersingular isogeny based on the crypto-friendly notation. Let $s \neq t$ and $s, t \in \{1, 2\}$.

Definition 1 (SI-CDH Assumption [JD14,FTTY18]). *The SI-CDH problem is that, given public parameters $\mathfrak{g}$ and $\mathfrak{e}$, and $\mathfrak{g}^{\mathfrak{a}}$, $\mathfrak{g}^{\mathfrak{b}}$ where $\mathfrak{a} \leftarrow \mathbb{Z}_{\ell_s^{e_s}}$, $\mathfrak{b} \leftarrow \mathbb{Z}_{\ell_t^{e_t}}$, compute the j-invariant $(\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}} = (\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}}$. For any PPT algorithm $\mathcal{A}$, we define the advantage of solving SI-CDH problem as*

$$Adv_{\mathcal{A}}^{sicdh} = Pr[j' = (\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}} | j' \leftarrow \mathcal{A}(\mathfrak{g}, \mathfrak{e}, \mathfrak{g}^{\mathfrak{a}}, \mathfrak{g}^{\mathfrak{b}})].$$

The SI-CDH assumption states: for any PPT algorithm $\mathcal{A}$, the advantage of solving SI-CDH problem is negligible.

Definition 2 (SI-DDH Assumption [JD14,FTTY18]). *Let $\mathfrak{g}$ and $\mathfrak{e}$ be that defined in SI-CDH assumption. Let D_0 and D_1 be two distributions defined as:*

$$\begin{aligned}
D_1 :=& \{\mathfrak{e}, \mathfrak{g}, \mathfrak{g}^{\mathfrak{a}}, \mathfrak{g}^{\mathfrak{b}}, (\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}} | \mathfrak{a} \leftarrow \mathbb{Z}_{\ell_s^{e_s}}, \mathfrak{b} \leftarrow \mathbb{Z}_{\ell_t^{e_t}}\} \\
D_0 :=& \{\mathfrak{e}, \mathfrak{g}, \mathfrak{g}^{\mathfrak{a}}, \mathfrak{g}^{\mathfrak{b}}, (\mathfrak{g}^{\mathfrak{s}})^{\mathfrak{t}} | \mathfrak{a}, \mathfrak{s} \leftarrow \mathbb{Z}_{\ell_s^{e_s}}, \mathfrak{b}, \mathfrak{t} \leftarrow \mathbb{Z}_{\ell_t^{e_t}}\}
\end{aligned}$$

The SI-DDH problem is that given a random sample from D_b depending on $b \leftarrow \{0, 1\}$, guess b. The advantage of solving SI-DDH problem for any PPT algorithm $\mathcal{A}$ is

$$Adv_{\mathcal{A}}^{siddh} = Pr[b' = b | b' \leftarrow \mathcal{A}(\mathfrak{d}_b \leftarrow D_b), b \leftarrow \{0, 1\}] - 1/2.$$

The SI-DDH assumption states: for any PPT algorithm $\mathcal{A}$, the advantage of solving SI-DDH problem is negligible.

2.3 CK$^+$ Security Model

We recall the CK$^+$ model introduced by [Kra05] and later refined by [FSXY12], which is a CK model [CK01] integrated with the weak PFS, resistance to KCI and MEX properties. We focus on 3-pass and 2-pass protocols in this paper. For simplicity, we only show the model specified to 2-pass protocols. As for 3-pass protocol, we can extend it by adding an extra message in the matching session identifier and Send definitions.

In an AKE protocol, U_i denotes a party indexed by i, who is modeled as a probabilistic polynomial time (PPT) interactive Turing machine. We assume that each party U_i owns a static pair of secret-public key (sk_i, pk_i), where the static public key is related to U_i's identity by a certification authority (CA). No other actions by the CA are required or assumed. In particular, we make no assumption on whether the CA requires a proof-of possession of the private key from a registrant of a public key, and we do not assume any specific checks on the value of a public key.

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*. 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 is called the session *responder*.

According to the specification of AKE, the party creates 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. Note that Canetti-Krawczyk [CK01] defines session state as session-specific secret information, but leaves it up to a protocol to specify which information is included in a session state. LaMacchia et al. [LLM07] explicitly set all random coins used by a party in a session as session-specific secret information and call it *ephemeral secret key*. 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, then 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 say 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 follows to capture real attacks in open networks, including the control of communication and the access to some of the secret information.

- Send(message): $\mathcal{A}$ sends messages in one of the 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.
- SessionKeyReveal(sid): if the session sid is completed, $\mathcal{A}$ obtains the session key SK for sid.
- SessionStateReveal(sid): $\mathcal{A}$ obtains the session state of the owner of sid if the session is not completed. The session state includes all ephemeral secret keys and intermediate

computation results except for immediately erased information, but does not include the static secret key.
- $\mathsf{Corrupt}(U_i)$: By this query, $\mathcal{A}$ learns all information of U_A (including the static secret, session states and session keys stored at U_A). In addition, from the moment that U_A is corrupted, all its actions may be controlled by $\mathcal{A}$.

Freshness. Let $\mathsf{sid}^* = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ or $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ be a completed session between honest users 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) $\mathsf{SessionStateReveal}(\mathsf{sid}^*)$, $\mathsf{SessionKeyReveal}(\mathsf{sid}^*)$, and $\mathsf{SessionStateReveal}(\overline{\mathsf{sid}}^*)$, $\mathsf{SessionKeyReveal}(\overline{\mathsf{sid}}^*)$ if $\overline{\mathsf{sid}}^*$ exists; 2) $\mathsf{SessionStateReveal}(\mathsf{sid}^*)$ and $\mathsf{SessionKeyReveal}(\mathsf{sid}^*)$ if $\overline{\mathsf{sid}}^*$ does not exist.
Security Experiment. The adversary $\mathcal{A}$ could make a sequence of the queries described above. During the experiment, $\mathcal{A}$ makes the query of $\mathsf{Test}(\mathsf{sid}^*)$, where sid^* must be a fresh session. $\mathsf{Test}(\mathsf{sid}^*)$ select random bit $b \in \{0, 1\}$, and return the session key held by sid^* if $b = 0$; and return a random key if $b = 1$. The experiment continues until $\mathcal{A}$ returns b' as a guess of b. The adversary $\mathcal{A}$ wins the game if the test session sid^* is still fresh and $b' = b$. The advantage of the adversary $\mathcal{A}$ is defined as $\mathsf{Adv}_{\Pi}^{ck+}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}$.

Definition 3. *We say 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 $\mathcal{A}$, $\mathsf{Adv}_{\Pi}^{\mathrm{CK}+}(\mathcal{A})$ is negligible for the test session sid^*,*

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 the 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.*

As indicated in Table 2, the CK$^+$ model captures all non-trivial patterns of exposure of static and ephemeral secret keys listed in Definition 3, and these ten cases cover wPFS, resistance to KCI, and MEX attacks.

2.4 2-Key PKE and KEM

In this section, we provide the definitions of 2-key PKE and 2-key KEM, as well as the modified Fujisaki-Okamoto transformation proposed by Xue et al. [XLL+18].

A 2-key PKE with a plaintext space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$ consists of a quadruple of PPT algorithms $\mathsf{2PKE}$=(KeyG1, KeyG0, Enc, Dec) described as follows:

- $\mathsf{KeyG1}(n, pp)$: on input a security parameter n and public parameter pp, output a pair of public and secret keys (pk_1, sk_1).

Event	Case	sid^*	$\overline{\mathsf{sid}}^*$	sk_A	ek_A	ek_B	sk_B	Security
E_1	1	A	No	$\surd$	$\times$	-	$\times$	KCI
E_2	2	A	No	$\times$	$\surd$	-	$\times$	MEX
E_3	2	B	No	$\times$	-	$\surd$	$\times$	MEX
E_4	1	B	No	$\times$	-	$\times$	$\surd$	KCI
E_5	4	A or B	Yes	$\surd$	$\times$	$\times$	$\surd$	wPFS
E_6	5	A or B	Yes	$\times$	$\surd$	$\surd$	$\times$	MEX
$E_{7\text{-}1}$	3	A	Yes	$\surd$	$\times$	$\surd$	$\times$	KCI
$E_{7\text{-}2}$	3	B	Yes	$\times$	$\surd$	$\times$	$\surd$	KCI
$E_{8\text{-}1}$	6	A	Yes	$\times$	$\surd$	$\times$	$\surd$	KCI
$E_{8\text{-}2}$	6	B	Yes	$\surd$	$\times$	$\surd$	$\times$	KCI

Table 2. The behavior of AKE adversary in CK$^+$ model. $\overline{\mathsf{sid}}^*$ is the matching session of sid^*, if it exists. "Yes" means that there exists $\overline{\mathsf{sid}}^*$ and "No" means not. sk_A (resp. sk_B) means the static secret key of A (resp. B). ek_A (resp. ek_B) is the ephemeral secret key of A (resp. B) in sid^* or $\overline{\mathsf{sid}}^*$ if there exists. "$\surd$" means the secret key may be revealed to adversary, "$\times$" means the secret key is not revealed. "-" means the secret key does not exist.

- $\mathsf{KeyG0}(n, pp)$: on input a security parameter n and public parameter pp, output a pair of public and secret keys (pk_0, sk_0).
- $\mathsf{Enc}(pk_1, pk_0, m; r)$: on input public keys pk_1, pk_0 and a plaintext $m \in \mathcal{M}$, output a ciphertext $C \in \mathcal{C}$.
- $\mathsf{Dec}(sk_1, sk_0, C)$: on input secret keys sk_1, sk_0 and a cipheretext $C \in \mathcal{C}$, output a plaintext m.

Correctness. For $(pk_1, sk_1) \leftarrow \mathsf{KeyG1}(n, pp)$, $(pk_0, sk_0) \leftarrow \mathsf{KeyG0}(n, pp)$ and $C \leftarrow \mathsf{Enc}(pk_1, pk_0, m; r)$, then we have $\mathsf{Dec}(sk_1, sk_0, C) = m$.

Game $[\mathsf{OW\text{-}CPA}, \cdot]$ **on** pk_1	**Game** $[\cdot, \mathsf{OW\text{-}CPA}]$ **on** pk_0
01 $(pk_1, sk_1) \leftarrow \mathsf{KeyG1}(n, pp)$;	07 $(pk_0, sk_0) \leftarrow \mathsf{KeyG0}(n, pp)$;
02 $(state, pk_0^*) \leftarrow \mathcal{A}_1(pk_1)$;	08 $(state, pk_1^*) \leftarrow \mathcal{B}_1(pk_0)$;
03 $m \leftarrow \mathcal{M}$;	09 $m \leftarrow \mathcal{M}$;
04 $c^* \leftarrow \mathsf{Enc}(pk_1, pk_0^*, m)$;	10 $c^* \leftarrow \mathsf{Enc}(pk_1^*, pk_0, m)$;
05 $m' \leftarrow \mathcal{A}_2(state, c^*)$;	11 $m' \leftarrow \mathcal{B}_2(state, c^*)$;
06 return $m' \stackrel{?}{=} m$	12 return $m' \stackrel{?}{=} m$

Fig. 3. The $[\mathsf{OW\text{-}CPA}, \cdot]$ (resp. $[\cdot, \mathsf{OW\text{-}CPA}]$) game of 2PKE for adversaries $\mathcal{A}$ (resp. $\mathcal{B}$).

The security games of 2PKE are formalized in Figure 3. We define the advantage of $\mathcal{A}$ winning in the game $[\mathsf{OW\text{-}CPA}, \cdot]$ as $\mathrm{Adv}^{[\mathsf{OW\text{-}CPA}, \cdot]}_{\mathsf{2PKE}}(\mathcal{A}) = \Pr[[\mathsf{OW\text{-}CPA}, \cdot]^{\mathcal{A}} \Rightarrow 1]$, and the advantage of $\mathcal{B}$ in the game $[\cdot, \mathsf{OW\text{-}CPA}]$ as $\mathrm{Adv}^{[\cdot, \mathsf{OW\text{-}CPA}]}_{\mathsf{2PKE}}(\mathcal{B}) = \Pr[[\cdot, \mathsf{OW\text{-}CPA}]^{\mathcal{B}} \Rightarrow 1]$, respectively.

The 2-key key encapsulation (2-key KEM) 2KEM is defined similarly.

- $\mathsf{KeyGen1}(n, pp)$: on input a security parameter n and public parameter pp, output a pair of public-secret keys (pk_1, sk_1). In order to show the randomness that is used, we denote key generation algorithm as $\mathsf{KeyGen1}(n, r)$.

– $\mathsf{KeyGen0}(n, pp)$: on input a security parameter n and public parameter pp, output a pair of public and secret keys (pk_0, sk_0).
– $\mathsf{Encaps}(pk_1, pk_0)$: on input public keys pk_1, pk_0, output a ciphertext c and encapsulated key k in key space $\mathcal{K}$. Sometimes, we explicitly add the randomness r and denote it as $\mathsf{Encaps}(pk_1, pk_0; r)$.
– $\mathsf{Decaps}(sk_1, sk_0, c)$: on input secret keys sk_1, sk_0 and a ciphertext c, output a key k.

CORRECTNESS. For $(pk_1, sk_1) \leftarrow \mathsf{KeyGen1}(n, pp)$, $(pk_0, sk_0) \leftarrow \mathsf{KeyGen0}(n, pp)$ and $(c, k) \leftarrow \mathsf{Encaps}(pk_1, pk_0)$, it holds that $\mathsf{Decaps}(sk_1, sk_0, c) = k$.

Game [OW-CCA, $\cdot$] on pk_1	**Game [$\cdot$, OW-CPA] on pk_0**
01 $(pk_1, sk_1) \leftarrow \mathsf{KeyGen1}(n, pp)$;	07 $(pk_0, sk_0) \leftarrow \mathsf{KeyGen0}(n, pp)$;
02 $L_0 = \{(-, -, -)\}$;	08 $(state, pk_1^*) \leftarrow \mathcal{B}_1(pk_0)$;
03 $(state, pk_0^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{cca}}, \mathcal{O}_{\mathsf{leak}_0}}(pk_1)$;	09 $(c^*, k^*) \leftarrow \mathsf{Encaps}(pk_1^*, pk_0)$;
04 $(c^*, k^*) \leftarrow \mathsf{Encaps}(pk_1, pk_0^*)$;	10 $k' \leftarrow \mathcal{B}_2(state, c^*)$;
05 $k' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{cca}}, \mathcal{O}_{\mathsf{leak}_0}}(state, c^*)$;	11 return $k' \stackrel{?}{=} k^*$
06 return $k' \stackrel{?}{=} k^*$	

Fig. 4. The $[\mathsf{OW\text{-}CCA}, \cdot]$ (resp. $[\cdot, \mathsf{OW\text{-}CPA}]$) game of 2KEM for adversaries $\mathcal{A}$ (resp. $\mathcal{B}$). The oracles $\mathcal{O}_{\mathsf{leak}_0}$ and $\mathcal{O}_{\mathsf{cca}}$ are defined in the following.

The security games of 2KEM are formalized in Figure 4. On the i-th query of $\mathcal{O}_{\mathsf{leak}_0}$, the challenger generates $(pk_0^i, sk_0^i) \leftarrow \mathsf{KeyGen0}(r_0^i)$, sets $L_0 = L_0 \cup \{(pk_0^i, sk_0^i)\}$ and returns (pk_0^i, sk_0^i) to adversary $\mathcal{A}$. $\mathcal{O}_{\mathsf{cca}}(pk_0', c')$ works as follows: If $pk_0' \in [L_0]_1$ and $(c', pk_0') \neq (c^*, pk_0^*)$, compute and return the corresponding $k' \leftarrow \mathsf{Decaps}(sk_1, sk_0', c')$, otherwise return $\perp$.

We define the advantage of $\mathcal{A}$ winning in the game $[\mathsf{OW\text{-}CCA}, \cdot]$ as

$$\mathrm{Adv}_{\mathsf{2KEM}}^{[\mathsf{OW\text{-}CCA}, \cdot]}(\mathcal{A}) = \Pr[[\mathsf{OW\text{-}CCA}, \cdot]^{\mathcal{A}} \Rightarrow 1],$$

and the advantage of $\mathcal{B}$ winning in the game $[\cdot, \mathsf{OW\text{-}CPA}]$ as:

$$\mathrm{Adv}_{\mathsf{2KEM}}^{[\cdot, \mathsf{OW\text{-}CPA}]}(\mathcal{B}) = \Pr[[\cdot, \mathsf{OW\text{-}CPA}]^{\mathcal{B}} \Rightarrow 1].$$

According to [XLL+18], the modified Fujisaki-Okamoto transformation in Fig. 5 builds a $[\mathsf{OW\text{-}CCA}, \mathsf{OW\text{-}CPA}]$ secure 2-Key KEM from any $[\mathsf{OW\text{-}CPA}, \mathsf{OW\text{-}CPA}]$ secure 2-key PKE. Note that in [XLL+18] they consider the decryption failure, but we do not take the decryption failure into account here since the encryption scheme based on SI-DDH is perfectly correct.

Lemma 1 (Theorem 7 [XLL+18]). *For any $[\mathsf{OW\text{-}CCA}, \cdot]$ adversary $\mathcal{C}$, or $[\cdot, \mathsf{OW\text{-}CPA}]$ adversary $\mathcal{D}$ against 2KEM with at most q_H queries to random oracle H, there are $[\mathsf{OW\text{-}CPA}, \cdot]$ adversary $\mathcal{A}$, or $[\cdot, \mathsf{OW\text{-}CPA}]$ adversary $\mathcal{B}$ against 2PKE, that make at most q_H (resp. q_G) queries to random oracle H (resp. G) s.t.*

$$Adv_{\mathsf{2KEM}}^{[\mathsf{OW\text{-}CCA}, \cdot]}(\mathcal{C}) \leq \frac{q_H}{2^n} + q_H \cdot Adv_{\mathsf{2PKE}}^{[\mathsf{OW\text{-}CPA}, \cdot]}(\mathcal{A}),$$

$$Adv_{\mathsf{2KEM}}^{[\cdot, \mathsf{OW\text{-}CPA}]}(\mathcal{D}) \leq Adv_{\mathsf{2PKE}}^{[\cdot, \mathsf{OW\text{-}CPA}]}(\mathcal{B}).$$

$\underline{\mathsf{KeyGen1}(n)}$	$\underline{\mathsf{KeyGen0}(n)}$
$(pk_1', sk_1') \leftarrow \mathsf{KeyG1}$	$(pk_0', sk_0') \leftarrow \mathsf{KeyG0}$
$s_1 \leftarrow \{0,1\}^n$	$sk_0 = sk_0'$
$sk_1 = (sk_1', s_1)$	$pk_0 = pk_0'$
$pk_1 = pk_1'$	return (K, c)
$\underline{\mathsf{Encaps}(pk_1, pk_0)}$;	$\underline{\mathsf{Decaps}(sk_1, sk_0, c)}$
$m \leftarrow \mathcal{M}$	$m' = \mathsf{Dec}(sk_1', sk_0', c)$
$c \leftarrow \mathsf{Enc}(pk_1, pk_0, m; G(m))$	$c' = \mathsf{Enc}(pk_1, pk_0, m'; G(m'))$
$K = H(pk_0, m, c)$	if $c \neq c'$, let $m' = s_1$
return (K, c)	return $K = H(pk_0, m', c)$

Fig. 5. The modified Fujisaki-Okamoto from [OW-CPA, OW-CPA] secure 2-key PKE to [OW-CCA, OW-CPA] secure 2-key KEM 2KEM.

3 [OW-CCA, OW-CPA] Secure KEM from SIDH

We now propose a [OW-CCA, OW-CPA] secure 2-key KEM from supersingular isogeny. It is the core building block for our AKEs. At first, we propose a [OW-CPA, OW-CPA] 2-key PKE from supersingular isogeny, and then apply the modified Fujisaki-Okamoto transformation to obtain a 2-key KEM.

Choose $p = \ell_1^{e_1}\ell_2^{e_2} \cdot f \pm 1, E_0, \{P_1, Q_1\}, \{P_2, Q_2\}$ as above. Let $h : \{0,1\}^* \rightarrow \{0,1\}^n$ be a random hash function from pair-wise independent hash function families $\mathcal{H}$. Let $\mathfrak{g} = (E_0; P_1, Q_1, P_2, Q_2)$ and $\mathfrak{e} = (\ell_1, \ell_2, e_1, e_2)$ be public parameters. Let $\{s, t\} = \{1, 2\}$. The [OW-CPA, OW-CPA] 2-key PKE $\mathsf{2PKE}_{sidh}$ is built as follows.

- $\mathsf{KeyG1}(n,\mathsf{pp})$: on input security parameter and public parameter, randomly choose a secret $\mathfrak{a}_1 \leftarrow \mathbb{Z}_{\ell_s^{e_s}}$ and compute $\mathfrak{g}^{\mathfrak{a}_1}$. Then output
$$sk_1 := \mathfrak{a}_1, pk_1 := \mathfrak{g}^{\mathfrak{a}_1}.$$
- $\mathsf{KeyG0}(n,\mathsf{pp})$: on input security parameter and public parameter, randomly choose a secret $\mathfrak{a}_0 \leftarrow \mathbb{Z}_{\ell_s^{e_s}}$ and compute $\mathfrak{g}^{\mathfrak{a}_0}$. Then output
$$sk_0 := \mathfrak{a}_0, pk_0 := \mathfrak{g}^{\mathfrak{a}_0}.$$
- $\mathsf{Enc}(pk_1, pk_0, m)$: on input public keys and a message $m = m_1||m_0 \in \{0,1\}^{2n}$, randomly choose $\mathfrak{b} \leftarrow \mathbb{Z}_{\ell_t^{e_t}}$ and compute $\mathfrak{g}^{\mathfrak{b}}$, $h((\mathfrak{g}^{\mathfrak{a}_1})^{\mathfrak{b}}) \oplus m_1$ and $h((\mathfrak{g}^{\mathfrak{a}_0})^{\mathfrak{b}}) \oplus m_0$. The ciphertext is
$$c := \left(\mathfrak{g}^{\mathfrak{b}}, h\left((\mathfrak{g}^{\mathfrak{a}_1})^{\mathfrak{b}}\right) \oplus m_1, h\left((\mathfrak{g}^{\mathfrak{a}_0})^{\mathfrak{b}}\right) \oplus m_0\right).$$
- $\mathsf{Dec}(sk_1, sk_0, c)$: on input secret keys $sk_1 = \mathfrak{a}_1$, $sk_0 = \mathfrak{a}_0$ and ciphertext $c = (\mathfrak{c}_1, c_2, c_3)$, compute $m_1 := c_2 \oplus h\left(\mathfrak{c}_1^{\mathfrak{a}_1}\right)$ and $m_0 := c_3 \oplus h\left(\mathfrak{c}_1^{\mathfrak{a}_0}\right)$. The plaintext is $m = m_1||m_0$.

The correctness of $\mathsf{2PKE}_{sidh}$ is straightforward due to the correctness of SIDH.

Lemma 2. *Under the SI-DDH assumption, $\mathsf{2PKE}_{sidh}$ is [OW-CPA, OW-CPA] secure. Precisely, for any PPT [OW-CPA, ·] (resp. [·, OW-CPA]) adversary $\mathcal{A}$ (resp. $\mathcal{C}$), there exists algorithm $\mathcal{B}$ (resp. $\mathcal{D}$) such that*
$$Adv^{[\mathsf{OW\text{-}CPA},\cdot]}_{\mathsf{2PKE_{sidh}}}(\mathcal{A}) \leq 2Adv^{siddh}_{\mathcal{B}} + 1/2^n + negl,$$
$$(resp.\quad Adv^{[\cdot,\mathsf{OW\text{-}CPA}]}_{\mathsf{2PKE_{sidh}}}(\mathcal{C}) \leq 2Adv^{siddh}_{\mathcal{D}} + 1/2^n + negl).$$

Proof. We reduce the $[\mathsf{OW\text{-}CPA}, \cdot]$ security to the underlying SI-DDH assumption. It is analogous for the $[\cdot, \mathsf{OW\text{-}CPA}]$ security. We prove the $[\mathsf{OW\text{-}CPA}, \cdot]$ security via a sequence of games.

Game 0: This is the original $[\mathsf{OW\text{-}CPA}, \cdot]$ challenge game in Fig. 3. We denote the event that the adversary wins the games as Succ_0.

Game 1: In this game we modify $[\mathsf{OW\text{-}CPA}, \cdot]$ challenge game by requiring that the adversary wins the game if $m'_1 = m_1$. We denote this event as Succ_1 (In Game i ($i \geq 1$), we denote this event as Succ_i). Note that in Game 0, the adversary wins only if both $m'_1 = m_1$ and $m'_0 = m_0$. Thus, we have $\Pr[\mathsf{Succ}_0] \leq \Pr[\mathsf{Succ}_1]$.

Game 2: In this game, we modify the computation of challenge ciphertext. Specifically, $(\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}_1}$ is replaced by a random j-invariant j^*. We construct an algorithm $\mathcal{B}$ to solve the SI-DDH problem given an instance $(\mathfrak{g}, \mathfrak{g}_1, \mathfrak{g}_2, j)$, if there exists an algorithm $\mathcal{A}$ to distinguish Game 1 and Game 2.

$\underline{\mathcal{B}(\mathfrak{e}, \mathfrak{g}, \mathfrak{g}_1, \mathfrak{g}_2, j)}$
01 $pk_1 \leftarrow \mathfrak{g}_1$
02 $pk_0^*, \text{state} \leftarrow \mathcal{A}(pk_1)$
03 $m_1 \leftarrow \{0,1\}^n$
04 $\mathfrak{c}_1^* = \mathfrak{g}_2$, $c_2^* = h(j) \oplus m_1$, $c_3^* \leftarrow \{0,1\}^n$
05 $c^* = (\mathfrak{c}_1^*, c_2^*, c_3^*)$
06 $m'_1 || m'_0 \leftarrow \mathcal{A}(\text{state}, c^*)$
07 If $m'_1 = m_1$, $b' = 1$, else $b' \leftarrow \{0,1\}$.

If $(\mathfrak{g}, \mathfrak{g}_1, \mathfrak{g}_2, j)$ is an SI-DDH tuple, $\mathcal{B}$ perfectly simulates Game 1, else $\mathcal{B}$ perfectly simulates Game 2. In the SI-DDH challenge, we have

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

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

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

Remark 1: By setting pk_0 and sk_0 to be empty and the ciphertext to be $\mathfrak{c}_1, c_2$, the scheme is exactly the ElGamal scheme and is $\mathsf{OW\text{-}CPA}$ secure under the SI-DDH assumption.

Applying the modified Fujisaki-Okamoto in Fig. 5, we get a $[\mathsf{OW\text{-}CCA}, \mathsf{OW\text{-}CPA}]$ secure 2-key KEM $\mathsf{2KEM}_{sidh}$ in Fig. 6. Let $G : \{0,1\}^{2n} \to \{0,1\}^*$ and $H : \{0,1\}^* \to \{0,1\}^{2n}$ be hash functions. Note that there is a subtle difference between the Fig. 6 and the modified Fujisaki-Okamoto in Fig. 5 that the "re-encryption" only needs to check the correctness of $\mathfrak{c}_1$.

Theorem 1. *Under the SI-DDH assumption,* $\mathsf{2KEM}_{sidh}$ *is* $[\mathsf{OW\text{-}CCA}, \mathsf{OW\text{-}CPA}]$ *secure in the random oracle model. Precisely, for any PPT* $[\mathsf{OW\text{-}CCA}, \cdot]$ *(resp.* $[\cdot, \mathsf{OW\text{-}CPA}]$*) adversary*

KeyGen1	**KeyGen0**
$\mathfrak{a}_1 \leftarrow \mathbb{Z}_{\ell_s^{e_s}}$, $s_1 \leftarrow \{0,1\}^{2n}$	$\mathfrak{a}_0 \leftarrow \mathbb{Z}_{\ell_s^{e_s}}$
$pk_1 := \mathfrak{g}^{\mathfrak{a}_1}$, $sk_1 := (\mathfrak{a}_1, s_1)$	$pk_0 := \mathfrak{g}^{\mathfrak{a}_0}$, $sk_0 := \mathfrak{a}_0$
Encaps(pk_1, pk_0)	**Decaps**(sk_1, sk_0)
$m_1, m_0 \leftarrow \{0,1\}^n$, $\mathfrak{b} := G(m_1, m_0)$	$m_1' := c_2 \oplus h(\mathfrak{c}_1^{\mathfrak{a}_1})$
$\mathfrak{c}_1 = \mathfrak{g}^{\mathfrak{b}}$, $c_2 = h((\mathfrak{g}^{\mathfrak{a}_1})^b) \oplus m_1$	$m_0' := c_3 \oplus h(\mathfrak{c}_1^{\mathfrak{a}_0})$
$c_3 = h((\mathfrak{g}^{\mathfrak{a}_0})^b) \oplus m_0$	$\mathfrak{b} := G(m_1', m_0')$
$c := (\mathfrak{c}_1, c_2, c_3)$	If $\mathfrak{c}_1 \neq \mathfrak{g}^{\mathfrak{b}}$, $m_1 \| m_0 = s_1$
$K := H(pk_0, m_1 \| m_0, c)$	$K := H(pk_0, m_1 \| m_0, c)$

Fig. 6. The [OW-CCA, OW-CPA] secure $\mathsf{2KEM}_{sidh}$.

$\mathcal{A}$ (resp. $\mathcal{C}$) with at most q_H queries to H oracle, there exists algorithm $\mathcal{B}$ (resp. $\mathcal{D}$) solving SI-DDH problem such that

$$Adv^{[\mathsf{OW\text{-}CCA},\cdot]}_{\mathsf{2KEM}_{sidh}}(\mathcal{A}) \leq \frac{q_H}{2^{2n}} + q_H \cdot \left(2Adv^{siddh}_{\mathcal{B}} + 1/2^n + negl\right),$$

$$(resp.\ Adv^{[\cdot,\mathsf{OW\text{-}CPA}]}_{\mathsf{2KEM}_{sidh}}(\mathcal{C}) \leq 2Adv^{siddh}_{\mathcal{D}} + 1/2^n + negl).$$

Proof. According to Lemma 1, the [OW-CCA, OW-CPA] security of $\mathsf{2KEM}_{sidh}$ is guaranteed by the [OW-CPA, OW-CPA] security of $\mathsf{2PKE}_{sidh}$. By applying Lemma 2, the [OW-CCA, OW-CPA] security is finally reduced to the underlying SI-DDH assumption. □

Remark 2: By setting pk_0 and sk_0 to be empty, the message space to be $\{0,1\}^n$, the input of G to be $(m_1, -)$ and the ciphertext to be $\mathfrak{c}_1, c_2$, the scheme is exactly the FO transformed ElGamal scheme and is OW-CCA secure under the SI-DDH assumption.

4 Two-pass SIAKE

In this section, we propose a two-pass AKE based on SI-DDH assumption. The two-pass AKE SIAKE_2 is shown in Fig. 7.

Public Parameters: Let $\mathfrak{e} = (\ell_1, \ell_2, e_1, e_2)$ and $\mathfrak{g} = (E_0; P_1, Q_1, P_2, Q_2)$. Let $g : \{0,1\}^* \to \{0,1\}^{2n}$, $h : \{0,1\}^n \to \{0,1\}^n$, $G : \{0,1\}^{2n} \to \{0,1\}^*$, $H : \{0,1\}^* \to \{0,1\}^{2n}$, $\tilde{H} : \{0,1\}^* \to \{0,1\}^n$ be hash functions.

Register: Any user registers two sets of public-secret keys. One set of keys is assigned by the user as initiator, and another set is assigned as responder. For user U_A, it first chooses $sk_{A_1} := (\mathfrak{a}_1 \in \mathbb{Z}_{\ell_1^{e_1}}, s_{A1} \leftarrow \{0,1\}^{2n})$ and computes $pk_{A_1} := \mathfrak{g}^{\mathfrak{a}_1}$, then chooses $sk_{A_2} := (\mathfrak{a}_2 \in \mathbb{Z}_{\ell_2^{e_2}}, s_{A2} \leftarrow \{0,1\}^{2n})$ and computes $pk_{A_2} := \mathfrak{g}^{\mathfrak{a}_2}$.

Phase 1: User U_A randomly chooses $r_A, \mathfrak{x}_0 \leftarrow \mathbb{Z}_{\ell_1^{e_1}}$ as two ephemeral randomness. Let $n_1 \leftarrow g(r_A, \mathfrak{a}_1)$, $\mathfrak{x} := G(g(r_A, \mathfrak{a}_1))$. Then U_A computes $X_0 := \mathfrak{g}^{\mathfrak{x}_0}$, $X := \mathfrak{g}^{\mathfrak{x}}$, $x_1 := h((\mathfrak{g}^{\mathfrak{b}_2})^{\mathfrak{x}}) \oplus n_1$, and sends X_0, X, x_1 to U_B. U_A computes $K_A := H(n_1, X, x_1)$.

Phase 2: User U_B randomly chooses $r_B \leftarrow \mathbb{Z}_{\ell_2^{e_2}}$ as the ephemeral randomness and computes $m_1 \| m_0 \leftarrow g(r_B, s_b)$, $\mathfrak{y} \leftarrow G(m_1, m_0)$, and $Y := \mathfrak{g}^{\mathfrak{y}}$. On receiving (X_0, X, x_1) from U_A, U_B computes $y_1 := h((\mathfrak{g}^{\mathfrak{a}_1})^{\mathfrak{y}}) \oplus m_1$, $y_0 := h((\mathfrak{g}^{\mathfrak{x}_0})^{\mathfrak{y}}) \oplus m_0$, $K_B := H(X, m_1, m_0, Y, y_1, y_0)$, and sends (Y, y_1, y_0) to U_A. U_B decrypts X, x_1 to extract n_1' and $\mathfrak{x}' \leftarrow G(n_1')$. If $X \neq \mathfrak{g}^{\mathfrak{x}}$, set

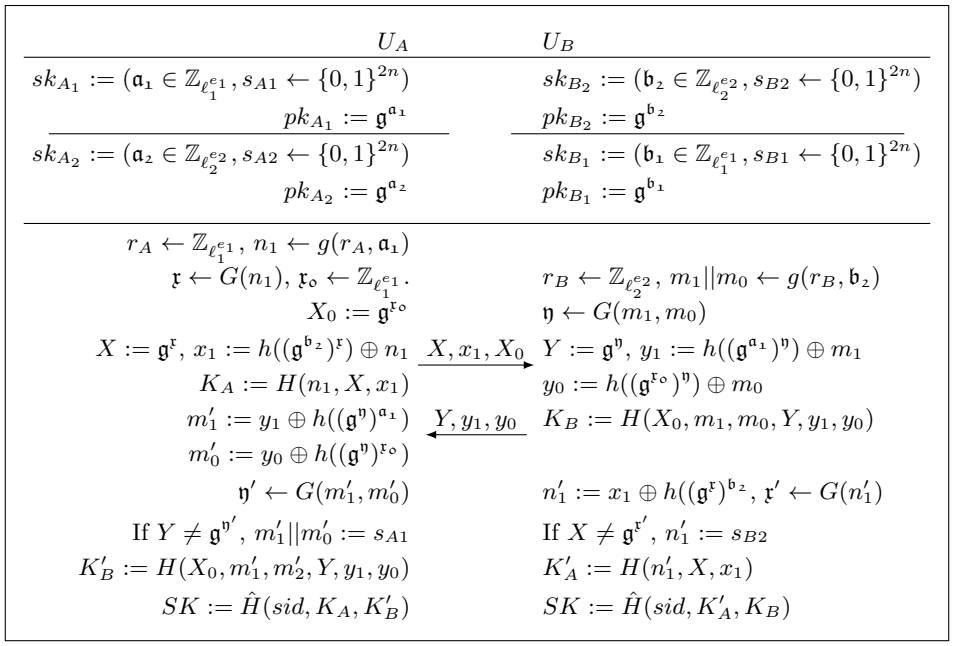

Fig. 7. A Compact 2-pass AKE SIAKE_2 Based on SI-DDH. Here sid is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, X_0, Y, y_1, y_0\big)$.

$n_1' := s_{B2}$. Let $K_A' := H(n_1', X, x_1)$. The session key is $SK := \hat{H}(sid, K_A', K_B)$ where sid is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, X_0, Y, y_1, y_0\big)$.

Phase 3: On receiving (Y, y_1, y_0) from U_B, U_A computes $m_1' := y_1 \oplus h((\mathfrak{g}^{\mathfrak{y}})^{\mathfrak{a}_1})$, $m_0' := y_0 \oplus h((\mathfrak{g}^{\mathfrak{y}})^{\mathfrak{x}})$ to extract $\mathfrak{y}' \leftarrow G(m_1', m_0')$. If $Y \neq \mathfrak{g}^{\mathfrak{y}}$, set $m_1'||m_0' := s_{A1}$. Let $K_B' := H(X_0, m_1', m_0', Y, y_1, y_0)$. The session key is $SK := \hat{H}(sid, K_A, K_B')$ where sid is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, X_0, Y, y_1, y_0\big)$.

The session state owned by U_A consists of the ephemeral secret key $r_A, \mathfrak{x}_\circ$, the decapsulated key K_B' and the encapsulated key K_A. The session state owned by U_B consists of the ephemeral secrete key r_B and the encapsulated key K_B, but not the decapsulated key K_A'.

Theorem 2. *Under the SI-DDH assumption,* SIAKE_2 *is* CK^+ *secure in the random oracle model. Precisely, if the number of users is* N *and there are at most* l *sessions between any two users, for any PPT adversary* $\mathcal{A}$ *against* SIAKE_2 *with* q *times of hash oracle queries, there exists* $\mathcal{B}$ *s.t.*

$$Adv^{\mathrm{CK}^+}_{\mathsf{SIAKE}_2}(\mathcal{A}) \leq 1/2 + N^2 \cdot l \cdot q \cdot \left(4Adv^{\mathsf{siddh}}_{\mathcal{B}} + \frac{q+1}{2^n} + negl\right).$$

Proof sketch: Obviously, U_A sends X_0 and a $\mathsf{OW\text{-}CCA}$ secure ciphertext X, x_1 under public key pk_{B_2} to U_B. U_B responds with a $[\mathsf{OW\text{-}CCA}, \mathsf{OW\text{-}CPA}]$ secure ciphertext Y, y_1, y_0 under public keys pk_{A_1} and X_0 to U_A. We first assume that the AKE adversary only has the capability to Send message and does not query $\mathsf{SessionKeyReveal}$ and $\mathsf{SessionStateReveal}$ on non-test sessions. Then under the assumption of $[\mathsf{OW\text{-}CPA}, \mathsf{OW\text{-}CPA}]$ security, SIAKE_2 is

secure. Take the event E_3 (one of the behaviors presented in Appendix A Table 6) as an example, where the adversary may send X_0 in the test session and he/she knows $\mathfrak{b}_2$ but not $\mathfrak{a}_1$ or r_B. Since the adversary does not know $\mathfrak{a}_1$ and $\mathfrak{y}$, the [OW-CPA, OW-CPA] security guarantees that K_B encapsulated in (Y, y_1, y_0) is secure (thus SK is random assuming $\hat{H}$ is a random oracle) even the adversary chooses part of the public key X_0. Note that to simulate the CK$^+$ game and reduce the advantage of the AKE adversary to the advantage of solving underlying [OW-CPA, $\cdot$] game, the simulator does not hold the static secret key $\mathfrak{a}_1$ of U_A. It is safe if the adversary does not make SessionKeyReveal and SessionStateReveal queries. However if the adversary makes SessionKeyReveal queries that involves U_A, the simulator fails to compute the encapsulated key and session key. Nevertheless, when the underlying KEM is [OW-CCA, OW-CPA] secure, the simulator could query the strong decapsulation oracle to get the encapsulated key and session key, so the reduction works. In other events, the proof proceeds similarly.

Proof. We give representative security proof in two cases E_5 and E_3 in Table 2, where one is wPFS and the other is the MEX attack. The other cases can be easily extended or modified from the proof of E_3, so they are omitted here. Table 3 presents the outline of reduction.

Assumption	2-Key PKE	2-Key KEM	Cases in Table 2
SI-DDH	[$\cdot$, OW-CPA], $pk_0 = \mathfrak{g}^{\mathfrak{r}_0}$	[$\cdot$, OW-CPA], $pk_0 = \mathfrak{g}^{\mathfrak{r}_0}$	E_5
SI-DDH	[OW-CPA, $\cdot$], $pk_1 = \mathfrak{g}^{\mathfrak{a}_1}$	[OW-CCA, $\cdot$], $pk_1 = \mathfrak{g}^{\mathfrak{a}_1}$	E_3, E_4, E_6, $E_{7\text{-}2}$, $E_{8\text{-}1}$
SI-DDH	OW-CPA	OW-CCA, $pk_1 = \mathfrak{g}^{\mathfrak{b}_2}$	E_1, E_2, $E_{7\text{-}1}$, $E_{8\text{-}2}$

Table 3. The outline of security reduction for SIAKE$_2$.

wPFS E_5. The proof of this case proceeds via a sequence of games, i.e. Game 0 to 2. In this case, the test session sid^* (with owner as responder or initiator) has a matching session $\overline{\mathsf{sid}}^*$. Both static secret keys of the initiator and the responder are leaked to $\mathcal{A}$. We denote the event that the AKE adversary $\mathcal{A}$ outputs b' such that $b = b'$ as Succ_i in Game i.

Game 0: This is the original CK$^+$ game in case E_5. In the test session, the adversary owns all the static secret keys, i.e. $\mathfrak{a}_1, \mathfrak{a}_2, \mathfrak{b}_1, \mathfrak{b}_2$ asssuming the test session is between U_A and U_B.

Game 1: In this game, we change the way to generate $m_1||m_0$ in the test session by replacing $m_1||m_0 \leftarrow g(r_B, \mathfrak{b}_2)$ with $m_1||m_0 \leftarrow \{0,1\}^{2n}$. Since g is a random oracle, $\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1] \leq N^2 \cdot l \cdot \frac{q}{2^n}$.

Game 2: In this game, we change the session key in the test session by replacing $\hat{H}(sid, K'_A, K_B)$ with a random bit-string in $\{0,1\}^n$. Obviously, $\Pr[\mathsf{Succ}_2] = 1/2$.

We construct an algorithm $\mathcal{B}$ to solve the [$\cdot$, OW-CPA] security of 2KEM$_{sidh}$, if there exists an algorithm $\mathcal{A}$ to distinguish Game 1 and Game 2.

On receiving the public key pk_0 from the [$\cdot$, OW-CPA] challenger, to simulate the CK$^+$ game, $\mathcal{B}$ randomly chooses two parties U_A, U_B and the i-th session as a guess of the test session with success probability $1/N^2 l$. $\mathcal{B}$ computes and sets all the static secrets and public key pairs by himself for all N users U_P as both responder and initiator. Particularly, $\mathcal{B}$ sets the static secret and public key pair (pk_{B_2}, sk_{B_2}) for U_B as responder, and sets pk_{A_1} for U_A as initiator. $\mathcal{B}$ sends pk_{A_1} to [$\cdot$, OW-CPA] challenger and receives the challenge ciphertext C^*. Then $\mathcal{B}$ simulates all the communications and SessionStateReveal and SessionKeyReveal as those in Game 1 except the test session. In the test session, $\mathcal{B}$ sets $X_0 = pk_0$ and responds $(Y, y_1, y_0) = C^*$.

Finally, $\mathcal{B}$ checks the hash list queried by $\mathcal{A}$. If there exists some $(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, X_0, C^*, K_A, K_B)$ in the list such that K_A is the key encapsulated in (X, x_1) (since (X, x_1) is honestly generated by $\mathcal{B}$, it can compute K_A), $\mathcal{B}$ chooses a random one and outputs K_B, otherwise $\perp$. Denote flag as the event that $\mathcal{A}$ explicitly queries $(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, X_0, C^*, K_A, K_B)$ to the oracle $\hat{H}$ such that K_A is the key encapsulated in (X, x_1) and K_B is the key encapsulated in C^*. If flag does not happen, $\mathcal{B}$ perfectly simulates both Game 1 and Game 2. Thus,

$$\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2] \leq \Pr[\mathsf{flag}] \leq N^2 \cdot l \cdot \mathrm{Adv}^{[\cdot,\mathsf{OW\text{-}CPA}]}_{\mathsf{2PKE}_{sidh}}(\mathcal{C}).$$

By Lemma 2, $\Pr[\mathsf{Succ}_0] \leq 1/2 + N^2 \cdot l \cdot \left(\frac{q}{2^n} + 2\mathrm{Adv}^{\mathsf{siddh}}_{\mathcal{B}} + 1/2^n + negl\right)$.

MEX E_3. In this case, the test session sid^* with its owner as responder does not have a matching session which means that X, x_1, X_0 is sent by adversary. And the randomness r_B are leaked to $\mathcal{A}$. It is more complicated than E_5. At first, (X, x_1, X_0) in the test session is generated by $\mathcal{A}$ rather than $\mathcal{B}$. However, (X, x_1) is the ciphertext under public key pk_{B_2}, and the encapsulated key K_A can be extracted with sk_{B_2}. Furthermore, the challenge public key that the security relies upon is the static public key, and this will affect the simulation of answering $\mathsf{SessionStateReveal}$ and $\mathsf{SessionKeyReveal}$ queries. Fortunately, $\mathsf{2PKE}_{sidh}$ provides a strong decapsulation oracle to answer those queries.

The proof also proceeds via a sequence of games, i.e., Game 0 to 2. We denote the event that $\mathcal{A}$ outputs b' such that $b = b'$ as Succ_i in Game i.

Game 0: This is the original CK$^+$ game in case E_3. In the test session, r_B is leaked to the adversary assuming the test session is between U_A and U_B.

Game 1: In this game, we change the way to generate $m_1||m_0$ in the test session by replacing $m_1||m_0 \leftarrow g(r_B, \mathfrak{b}_2)$ with $m_1||m_0 \leftarrow \{0,1\}^{2n}$. Although r_B is leaked to $\mathcal{A}$, since g is a random oracle, $\mathcal{A}$ will not find this change without querying g with $r_B, \mathfrak{b}_2$. We denote Askg as the event $\mathcal{A}$ queries g with $r_B, \mathfrak{b}_2$. If event Askg happens, we can extract $\mathfrak{b}_2$ and utilize it to solve the underlying SI-DDH problem. Precisely, given $(\mathfrak{g}, \mathfrak{g}_1, \mathfrak{g}_2, j)$, $\mathcal{B}$ randomly chooses U_B as a guess of the responder in the test session with success probability $\frac{1}{2N}$. $\mathcal{B}$ sets $pk_{B_2} := \mathfrak{g}_2$. When event Askg happens, $\mathcal{B}$ uses $\mathfrak{b}_2$ to output $j \stackrel{?}{=} \mathfrak{g}_1^{\mathfrak{b}_2}$.

$$\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1] \leq 2N \cdot \mathrm{Adv}^{\mathsf{siddh}}_{\mathcal{B}}.$$

Game 2: In this game, we change the session key in the test session by replacing $\hat{H}(sid, K'_A, K_B)$ with a random bit-string in $\{0,1\}^n$. Obviously, $\Pr[\mathsf{Succ}_2] = 1/2$.

We construct an algorithm $\mathcal{B}$ to solve the $[\mathsf{OW\text{-}CCA}, \cdot]$ security of $\mathsf{2KEM_{sidh}}$, if there exists an algorithm $\mathcal{A}$ to distinguish Game 1 and Game 2.

On receiving the public key pk_1 from the $[\mathsf{OW\text{-}CPA}, \cdot]$ challenger, to simulate the CK$^+$ game, $\mathcal{B}$ randomly chooses two parties U_A, U_B and the i-th session as a guess of the test session with success probability $1/N^2 l$. $\mathcal{B}$ computes and sets all the static secret and public key pairs on his own for all N users U_P as both responder and initiator except the static public key for U_A as initiator.

- Specifically, $\mathcal{B}$ sets the static secret and public key pair (pk_{A_2}, sk_{A_2}) that invloves U_A as responder, and sets pk_1 (receiving from the $[\mathsf{OW\text{-}CPA}, \cdot]$ challenger) for U_A as initiator.
- In the test session, on receiving (X, x_1, X_0) from $\mathcal{A}$, $\mathcal{B}$ sends $pk_0^* = X_0$ to the $[\mathsf{OW\text{-}CCA}, \cdot]$ challenger and receives the challenge ciphertext C^*. Then $\mathcal{B}$ returns C^* to $\mathcal{A}$ as response.
- $\mathcal{B}$ simulates all the communications and $\mathsf{SessionStateReveal}$ and $\mathsf{SessionKeyReveal}$ queries as those in Game 1 except that involves U_A as initiator (since $\mathcal{B}$ does not know sk_{A_1}).

– For those SessionStateReveal and SessionKeyReveal queries involves U_A as initiator (for example, U_A honestly sends out X', x'_1, X'_0 and receives (Y', y'_1, y'_0)), $\mathcal{B}$ queries the $\mathcal{O}$ oracle with $(X'_0; Y', y'_1, y'_0)$ provided by the $[\mathsf{OW\text{-}CCA}, \cdot]$ challenger to extract the encapsulated key and maintains the consistency of the $\hat{H}$ list with SessionStateReveal and SessionKeyReveal queries.

Finally, $\mathcal{B}$ checks the hash list queried by $\mathcal{A}$. If there exists some $(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, X_0, C^*, K_A, K_B)$ in the list such that K_A is the key encapsulated in (X, x_1) (since (X, x_1) is honestly generated by $\mathcal{B}$, it can compute K_A), $\mathcal{B}$ chooses a random one and outputs K_B, otherwise $\perp$. Denote flag as the event that $\mathcal{A}$ explicitly queries $(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, X_0, C^*, K_A, K_B)$ to the oracle $\hat{H}$ such that K_A is the key encapsulated in (X, x_1) and K_B is the key encapsulated in C^*. If flag does not happen, $\mathcal{B}$ perfectly simulates both Game 1 and Game 2. Thus,

$$\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2] \leq \Pr[\mathsf{flag}] \leq N^2 \cdot l \cdot \mathrm{Adv}^{[\mathsf{OW\text{-}CCA},\cdot]}_{\mathsf{2KEM}_{sidh}}(\mathcal{C}).$$

By Theorem 1, to sum up,

$$\Pr[\mathsf{Succ}_0] \leq 1/2 + N^2 \cdot l \cdot q \cdot \left(4\mathrm{Adv}^{\mathsf{siddh}}_{\mathcal{B}} + 1/2^n + negl\right).$$

□

5 Three-pass SIAKE

We first enhance the SI-DDH assumption to 1-Oracle SI-DH assumption, and analyze its reliability. Based on this assumption, we propose the three-pass SIAKE_3.

5.1 1-Oracle SI-DH and Implied 2-key KEM

The 1-Oracle SI-DH assumption is inspired by the Oracle Diffie-Hellman assumption over classical group given by Abdalla, Bellare and Rogaway [ABR01] for analyzing DHIES. Let $G :=< g >$ and $|G| = p$ be a prime. The Oracle Diffie-Hellman assumption states that, given (g, g^a, g^b, h), it is difficult to decide whether $h = H(g^{ab})$ or not (where H is a hash function), even the solver could make polynomial queries to an oracle $H_B(\cdot)$ which will return $H(v^b)$ with $v \in G$ satisfying $v \neq g^a$. Under the Oracle Diffie-Hellman assumption, the DHIES scheme is chosen ciphertext secure [ABR01].

However, the Oracle Diffie-Hellman assumption can not be directly extended in the supersingular isogeny setting. As we have presented several times before, the adaptive attack [GPST16] would allow extraction of every bit of b with polynomial queries to $H_B(\cdot)$ with specified points, implying the analogue of Oracle Diffie-Hellman problem in the supersingular isogeny setting could be solved. Moreover, in the classical group, if $v \neq g^a$, then $v^b \neq (g^a)^b$. However, in the supersingular isogeny setting, even if $v \neq g^a \in \mathrm{SSEC}_A$, it is possible that v^b is equal to $(g^a)^b$.

Fortunately, only one query to $H_B(\cdot)$ with $v \neq g^a$ is needed for our purpose and the adaptive attack does not work. Furthermore, when $H_B(v) = H(v, v^b)$, even if $v \neq g^a$, the case $H(v, v^b) = H(g^a, (g^a)^b)$ occurs with negligible probability.

Definition 4 (1-Oracle SI-DH Assumption). *Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function. Let $\mathfrak{e}$ and $\mathfrak{g}$ be public parameters as defined in SI-DDH assumption. Let D_0 and D_1 be two distributions:*

$$D_1 := \{\mathfrak{e}, \mathfrak{g}, \mathfrak{g}^{\mathfrak{a}}, \mathfrak{g}^{\mathfrak{b}}, H(\mathfrak{g}^{\mathfrak{a}}, (\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}}) | \mathfrak{a} \leftarrow \mathbb{Z}_{\ell_s^{e_s}}, \mathfrak{b} \leftarrow \mathbb{Z}_{\ell_t^{e_t}}\}$$
$$D_0 := \{\mathfrak{e}, \mathfrak{g}, \mathfrak{g}^{\mathfrak{a}}, \mathfrak{g}^{\mathfrak{b}}, h | \mathfrak{a} \leftarrow \mathbb{Z}_{\ell_s^{e_s}}, \mathfrak{b} \leftarrow \mathbb{Z}_{\ell_t^{e_t}}, h \leftarrow \{0,1\}^n\}$$

The 1-Oracle SI-DH problem is, given a random sample from D_b depending on $b \leftarrow \{0,1\}$, and a one-time oracle H_B, guessing b. The one-time oracle H_B can be queried only one time with $\mathfrak{y} \in SSEC_A$ and $\mathfrak{y} \neq \mathfrak{g}^{\mathfrak{a}}$, and it will return $H(\mathfrak{y}, \mathfrak{y}^{\mathfrak{b}})$. The advantage of $\mathcal{A}$ to solve the 1-Oracle SI-DH problem is

$$Adv_{\mathcal{A}}^{1osidh} = Pr[b' = b | \mathcal{A}^{H_B(\cdot)}(\mathfrak{d}_b \leftarrow D_b) = b', b \leftarrow \{0,1\}] - 1/2.$$

The 1-Oracle SI-DH assumption states that for any PPT algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}^{1osidh}$ is negligible.

We emphasize that the adversary is allowed to query the hashed SIDH oracle H_B only once with $\mathfrak{y} \neq \mathfrak{g}^{\mathfrak{a}}$. If there are polynomial queries, the 1-Oracle SI-DH problem can be solved by the adaptive attack in [GPST16]. Please also notice that the hash function involves $\mathfrak{g}^{\mathfrak{a}}$ or $\mathfrak{y}$ as input besides the j-invariant. Otherwise the 1-Oracle SI-DH problem is easy. Let $\mathfrak{g}^{\mathfrak{a}} = (E_A, \phi_A(P_t), \phi_A(Q_t))$. Since $\langle \phi_A(P_s) + [y]\phi_A(Q_s) \rangle = \langle [u]\phi_A(P_s) + [y][u]\phi_A(Q_s) \rangle$ for any integer $1 \leq u \leq \ell_s^{e_s}$ and coprime to ℓ_s, the attacker sets $E_Y = E_A$, $R = [u]\phi_A(P_s)$, $S = [u]\phi_A(Q_s)$ and $\mathfrak{y} = (Y, R, S)$. Then $(\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}}$ and $\mathfrak{y}^{\mathfrak{b}}$ will produce the same j-invariant. However, when taking $\mathfrak{g}^{\mathfrak{a}}$ or $\mathfrak{y}$ as input of H, any query with $\mathfrak{y} \neq \mathfrak{g}^{\mathfrak{a}}$ to H_B will get a completely different value.

1-Gap SI-DH problem is similar to the SI-CDH problem but the adversary is given access to a highly restricted SI-DDH oracle.

Definition 5 (1-Gap SI-DH Assmption). *Let $\mathfrak{e}$ and $\mathfrak{g}$ be public parameters. The 1-Gap SI-DH problem is that, given $\mathfrak{g}^{\mathfrak{a}}$, $\mathfrak{g}^{\mathfrak{b}}$ (where $\mathfrak{a} \leftarrow \mathbb{Z}_{\ell_s^{e_s}}$, $\mathfrak{b} \leftarrow \mathbb{Z}_{\ell_t^{e_t}}$), and an oracle $\mathcal{O}_{siddh}(\mathfrak{y}, \cdot)$, compute the j-invariant $(\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}} = (\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}}$. Here, $\mathfrak{y} \in SSEC_A$ is chosen by the adversary $\mathcal{A}$ at any time before its first queries to $\mathcal{O}_{siddh}(\mathfrak{y}, \cdot)$. $\mathcal{O}_{siddh}(\mathfrak{y}, j)$ will return 1 if $j = \mathfrak{y}^{\mathfrak{b}}$, and 0 otherwise. For any PPT algorithm $\mathcal{A}$, we define the advantage of solving 1-Gap SI-DH problem as*

$$Adv_{\mathcal{A}}^{1gsidh} = Pr[j' = (\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}} | \mathcal{A}^{\mathcal{O}_{siddh}(\mathfrak{y}, \cdot)}(\mathfrak{g}, \mathfrak{e}, \mathfrak{g}^{\mathfrak{a}}, \mathfrak{g}^{\mathfrak{b}}) \to (\mathfrak{y}, j')].$$

The 1-Gap SI-DH assumption states: for any PPT algorithm $\mathcal{A}$, the advantage of solving 1-Gap SI-DH problem is negligible.

We emphasize that if the adversary is allowed to query $\mathcal{O}_{\mathsf{siddh}}(\cdot, \cdot)$ with unlimited numbers of $\mathfrak{y}$, 1-Gap SI-DH problem can be solved using the adaptive attack in [GPST16]. However, here $\mathcal{O}_{\mathsf{siddh}}(\cdot, \cdot)$ oracle only allows to be queried once with $\mathfrak{y}$ of adversary's choice.

Discussion. These two assumptions are "non-standard" for supersingular isogeny. The adaptive attack [GPST16] and its extension can not easily break these new assumptions. We encourage more works on the analysis of the hardness of these two problems.

The following theorem shows that the 1-Gap SI-DH assumption implies the 1-Oracle SI-DH assumption when the hash function H is modeled as a random oracle.

Theorem 3. *For any algorithm $\mathcal{A}$ against the 1-Oracle SI-DH problem there exists an algorithm $\mathcal{B}$ against the 1-Gap SI-DH problem such that*

$$Adv_{\mathcal{A},H}^{1osidh}(n) \leq q_H \cdot Adv_{\mathcal{B}}^{1gsidh}(n),$$

where q_H is the number of times to query $\mathcal{O}_{siddh}(\mathfrak{y}, \cdot)$.

Proof. Let $\mathcal{A}$ be any algorithm solving the 1-Oracle SI-DH problem. We construct an algorithms $\mathcal{B}$ to solve the 1-Gap SI-DH problem using $\mathcal{A}$ as a sub-routine in Figure 8. The challenge is how to maintain the hash list so as to keep the consistency with the one-time Oracle H_B. Actually, the limited oracle $\mathcal{O}_{\mathsf{siddh}}(\cdot,\cdot)$ would help $\mathcal{B}$ to fix it.

Algorithm $\mathcal{B}^{\mathcal{O}_{\mathsf{siddh}}(\cdot,\cdot)}(\mathfrak{e},\mathfrak{g},\mathfrak{g}^{\mathfrak{a}},\mathfrak{g}^{\mathfrak{b}})$	
01 $h_0, h_1 \leftarrow \{0,1\}^n$	One time $H_B(\mathfrak{y})$
02 $b \leftarrow \{0,1\}$	01 Choose $\mathfrak{y}$ as the base of $\mathcal{O}_{\mathsf{siddh}}$
03 Run $\mathcal{A}^{H_B(\cdot),H}(\mathfrak{g},\mathfrak{g}^{\mathfrak{a}},\mathfrak{g}^{\mathfrak{b}},h_b)$	02 if $\exists(\mathfrak{y},j',h') \in L_H \wedge \mathcal{O}_{\mathsf{siddh}}(\mathfrak{y},j')=1$
04 a. For one-time query H_B	03 return h'
05 do as one-time H_B	04 else $h'' \leftarrow \{0,1\}^n$, $L_H = L_H \cup \{\mathfrak{y},j',h''\}$
06 b. For the H-query	05 return h''
07 do as $H(\mathfrak{x},j')$	$H(\mathfrak{x},j')$
08 c. Let b' be the output of $\mathcal{A}$	01 if $\exists(\mathfrak{x},j',h') \in L_H$ return h'
09 return $(\cdot,j,\cdot) \leftarrow L_H$	02 otherwise $h \leftarrow \{0,1\}^n$, $L_H = L_H \cup \{(\mathfrak{x},j',h)\}$
10 return j	03 return h

Fig. 8. Algorithm $\mathcal{B}$ for attacking the 1-Gap SI-DH problem.

Note that in Figure 8, if $H_B(\mathfrak{y})$ is asked at first and returns a random h, then when $(\mathfrak{y},j')$ is queried to H such that $\mathcal{O}_{\mathsf{siddh}}(\mathfrak{y},j')=1$, it will return h. If $H(\mathfrak{y},j')$ is asked at first and returns a random h, then when $\mathfrak{y}$ is asked to H_B such that $\mathcal{O}_{\mathsf{siddh}}(\mathfrak{y},j')=1$, it will return that h.

Let Ask be the event that $(\mathfrak{g}^{\mathfrak{a}},(\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}})$ is queried to H and $\overline{\mathsf{Ask}}$ be the complement of Ask. If Ask does not happen, there is no way to tell whether h_b is equal to $H(\mathfrak{g}^{\mathfrak{a}},(\mathfrak{g}^{\mathfrak{a}})^{\mathfrak{b}})$ or not. Thus we have that

$$\begin{aligned}
\mathrm{Adv}^{\mathsf{1osidh}}_{\mathcal{A},H} &= \Pr[\mathcal{A}^{H_B(\cdot)}(\mathfrak{b} \leftarrow D_b) = b] - 1/2 \\
&= \Pr[\mathcal{A}^{H_B(\cdot)}(\mathfrak{b} \leftarrow D_b) = b \wedge \mathsf{Ask}] + \Pr[\mathcal{A}^{H_B(\cdot)}(\mathfrak{b} \leftarrow D_b) = b \wedge \overline{\mathsf{Ask}}] - 1/2 \\
&= \Pr[\mathcal{A}^{H_B(\cdot)}(\mathfrak{b} \leftarrow D_b) = b \wedge \mathsf{Ask}] \\
&\leq \Pr[\mathsf{Ask}] \leq q_H \cdot \mathrm{Adv}^{1gsidh}_{\mathcal{B}}.
\end{aligned}$$

□

We now modify the $\mathsf{2PKE}_{sidh}$ and denote the new scheme as $\mathsf{2PKE}_{1osidh}$. The key generation algorithms are the same. In the encryption algorithm, $h\left((\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}_1}\right)$ is replaced by $h\left(\mathfrak{g}^{\mathfrak{b}},(\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}_1}\right)$ and $h\left((\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}_0}\right)$ is replaced by $h\left(\mathfrak{g}^{\mathfrak{b}},(\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}_0}\right)$. Thus the ciphertext is

$$c := \left(\mathfrak{g}^{\mathfrak{b}}, h\left(\mathfrak{g}^{\mathfrak{b}},(\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}_1}\right) \oplus m_1, h\left(\mathfrak{g}^{\mathfrak{b}},(\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}_0}\right) \oplus m_0\right).$$

Lemma 3. *The following holds.*

- *Under the 1-Oracle SI-DH assumption, the scheme $\mathsf{2PKE}_{1osidh}$ is $[\mathsf{OW\text{-}CPA},\cdot]$ secure even $H(pk_0^*,pk_0^{*\mathfrak{b}})$ is given to the adversary besides the challenge ciphertext $c^* = (\mathfrak{c}_1^* = \mathfrak{g}^{\mathfrak{b}}, c_1, c_2)$.*
- *If the $[\mathsf{OW\text{-}CPA},\cdot]$ game is changed as that pk_0^* is generated by the challenger and the corresponding sk_0^* is leaked to the adversary, then under the SI-DDH assumption, $\mathsf{2PKE}_{1osidh}$ satisfies this $[\mathsf{OW\text{-}CPA},\cdot]$ security even $H(pk_0^*,pk_0^{*\mathfrak{b}})$ is given to the adversary besides the challenge ciphertext $c^* = (\mathfrak{c}_1^* = \mathfrak{g}^{\mathfrak{b}}, c_1, c_2)$.*

– *Under the SI-DDH assumption, the scheme* $\mathsf{2PKE}_{1osidh}$ *is* $[\cdot, \mathsf{OW\text{-}CPA}]$ *secure.*

Proof. The $[\cdot, \mathsf{OW\text{-}CPA}]$ security is the same with that in Lemma 2. We reduce the $[\mathsf{OW\text{-}CPA}, \cdot]$ security with leakage to the underlying 1-Oracle SI-DH assumption.

Game 0: This is the original $[\mathsf{OW\text{-}CPA}, \cdot]$ challenge game in Fig. 3. We denote the event that the adversary wins the games as Succ_0.

Game 1: In this game, we modify the computation of challenge ciphertext. Specifically, $h(\mathfrak{g}^{\mathfrak{b}}, (\mathfrak{g}^{\mathfrak{b}})^{\mathfrak{a}_1}$ is replaced by a random bit $h \leftarrow \{0,1\}^n$. We construct an algorithm $\mathcal{B}$ to solve the 1-Oracle SI-DH problem given an instance $(\mathfrak{g}, \mathfrak{g}_1, \mathfrak{g}_2, h)$, and a one-time oacle $\mathcal{H}_B(\cdot)$, if there exists an algorithm $\mathcal{A}$ to distinguish Game 0 and Game 1.

$\mathcal{B}^{\mathcal{H}_B(\cdot)}(\mathfrak{e}, \mathfrak{g}, \mathfrak{g}_1, \mathfrak{g}_2 = \mathfrak{g}^{\mathfrak{b}}, h)$
01 $pk_1 \leftarrow \mathfrak{g}_1$
02 $pk_0^*, \text{state} \leftarrow \mathcal{A}(pk_1)$
03 $m_1 \leftarrow \{0,1\}^n$, $m_0 \leftarrow \{0,1\}^n$
04 Query $\mathcal{H}_B$ with pk_0^* and get $pk_0^{*\mathfrak{b}}$
04 $\mathfrak{c}_1^* = \mathfrak{g}_2$, $c_2^* = h \oplus m_1$, $c_3^* = h(pk_0^*, pk_0^{*\mathfrak{b}}) \oplus m_0$
05 $c^* = (\mathfrak{c}_1^*, c_2^*, c_3^*)$
06 $m_1' || m_0' \leftarrow \mathcal{A}(\text{state}, C^*)$
07 If $m_1' = m_1$, $b' = 1$, else $b' \leftarrow \{0,1\}$.

If $(\mathfrak{g}, \mathfrak{g}_1, \mathfrak{g}_2, h)$ is a 1-Oracle SI-DH tuple, $\mathcal{B}$ perfectly simulates Game 0, else $\mathcal{B}$ perfectly simulates Game 1. In the 1-Oracle SI-DH challenge, we have

$$\begin{aligned}
\mathrm{Adv}_{\mathcal{B}}^{1osidh} &= \Pr[b = b'] - 1/2 \\
&= 1/2(\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]) \\
&= 1/2(\Pr[b' = 1 | \text{Game 0}] - \Pr[b' = 1 | \text{Game 1}]) \\
&= 1/2(\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]).
\end{aligned}$$

Note that in this game, the $[\mathsf{OW\text{-}CPA}, \cdot]$ advantage is less than $1/2^n$. To Sum up, we have that, $\Pr[\mathsf{Succ}_0] \leq 2\mathrm{Adv}_{\mathcal{B}}^{siddh} + 1/2^n$. □

Similarly, we make the same modification to the $\mathsf{2KEM}_{sidh}$ and denote the new scheme as $\mathsf{2KEM}_{1osidh}$.

Theorem 4. *The following holds in the random oracle model.*

– *Under the 1-Oracle SI-DH assumption, the scheme* $\mathsf{2KEM}_{1osidh}$ *is* $[\mathsf{OW\text{-}CCA}, \cdot]$ *secure in the random oracle model, even* $h(pk_0^*, pk_0^{*\mathfrak{b}})$ *is given to the adversary besides the challenge ciphertext* $c^* = (\mathfrak{c}_1^* = \mathfrak{g}^{\mathfrak{b}}, c_1, c_2)$.
– *If the* $[\mathsf{OW\text{-}CCA}, \cdot]$ *game is changed as that* pk_0^* *is generated by challenger and the corresponding* sk_0^* *is leaked to the adversary, then under the SI-DDH assumption,* $\mathsf{2KEM}_{1osidh}$ *satisfies this* $[\mathsf{OW\text{-}CPA}, \cdot]$ *security even* $H(pk_0^*, pk_0^{*\mathfrak{b}})$ *is given to the adversary besides the challenge ciphertext* $c^* = (\mathfrak{c}_1^* = \mathfrak{g}^{\mathfrak{b}}, c_1, c_2)$.
– *Under the SI-DDH assumption, the scheme* $\mathsf{2KEM}_{1osidh}$ *is* $[\cdot, \mathsf{OW\text{-}CPA}]$ *secure.*

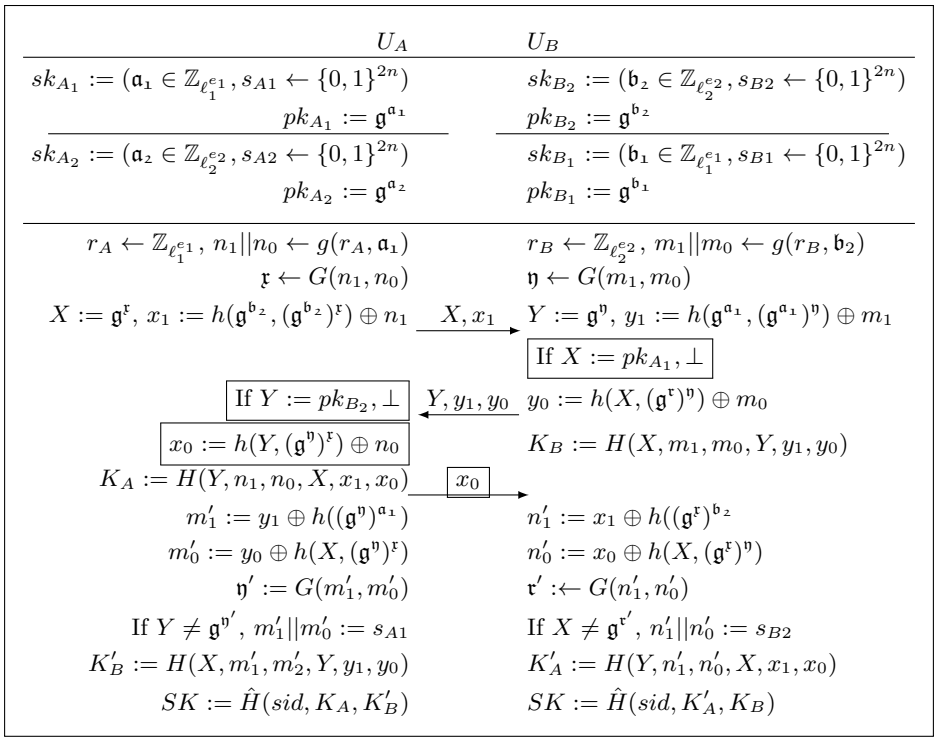

Fig. 9. A Compact 3-pass AKE SIAKE_3 based on SIDH. Here sid is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, x_0, Y, y_1, y_0\big)$. The boxed arguments are the main differences with SIAKE_2. Besides, the input of h includes the first part of the public key.

5.2 A Three-pass AKE based on 1-Oracle SI-DH Assumption

The three-pass AKE SIAKE_3 is shown in Fig. 9. The public parameters and register are the same with those for SIAKE_2.

Phase 1: User U_A chooses ephemeral randomness r_A. Let $n_1||n_0 \leftarrow g(r_A, \mathfrak{a}_1)$ and $\mathfrak{x} \leftarrow G(n_1, n_0)$. Then U_A computes $X := \mathfrak{g}^{\mathfrak{x}}$, $x_1 := h(g^{b_2}, (\mathfrak{g}^{\mathfrak{b}_2})^{\mathfrak{x}}) \oplus n_1$, and sends X, x_1 to U_B.

Phase 2: User U_B chooses ephemeral randomness $r_B \leftarrow \mathbb{Z}_{\ell_2^{e_2}}$ and computes $m_1||m_0 \leftarrow g(r_B, \mathfrak{b}_2)$, $\mathfrak{y} \leftarrow G(m_1, m_0)$, and $Y := \mathfrak{g}^{\mathfrak{y}}$. On receiving (X, x_1, X_0) from U_A, if $X := pk_{B_2}$, aborts, else U_B computes $y_1 := h(g^{a_1}, (\mathfrak{g}^{\mathfrak{a}_1})^{\mathfrak{y}}) \oplus m_1$, $y_0 := h(X, (\mathfrak{g}^{\mathfrak{r}})^{\mathfrak{y}}) \oplus m_0$, $K_B := H(X, m_1, m_0, Y, y_1, y_0)$, and sends (Y, y_1, y_0) to U_A.

Phase 3: On receiving (Y, y_1, y_0) from U_B, if $Y := pk_{A_1}$, aborts, else U_A decrypts Y, y_1, y_0 to extract $m_1'||m_0'$ and $\mathfrak{y}' \leftarrow G(m_1', m_0')$. If $Y \neq \mathfrak{g}^{\mathfrak{y}}$, then $m_1'||m_0' := s_{A1}$. U_A computes $K_B' := H(X, m_1', m_0', Y, y_1, y_0)$ and the session key as $SK := \hat{H}(sid, K_A, K_B')$, where sid is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, x_0, Y, y_1, y_0\big)$.

Phase 4: If $X := pk_{B_2}$, then aborts, else U_B decrypts X, x_1, x_0 to extract n_1', n_0' and $\mathfrak{r}' \leftarrow G(n_1', n_0')$. If $X \neq \mathfrak{g}^{\mathfrak{r}'}$, then $n_1'||n_0' := s_{B2}$. Let $K_A' := H(Y, n_1', n_0', X, x_1, x_0)$. The session key is computed as $SK := \hat{H}(sid, K_A', K_B)$ where sid is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, Y, y_1, y_0, x_0\big)$.

The session state of U_A consists of the ephemeral secret key r_A, the decapsulated key K'_B and the encapsulated key K_A. The session state of U_B consists of the ephemeral secrete key r_B, the encapsulated key K_B and the decapsulated key K'_A.

Theorem 5. *Under the 1-Oracle SI-DH assumption, SIAKE_3 is CK$^+$ secure in the random oracle model. Precisely, if the number of users is N and there are at most l sessions between any two users, for any PPT adversary $\mathcal{A}$ against SIAKE_3 with q times of hash oracle queries, there exists $\mathcal{B}$ s.t.*

$$Adv^{\mathrm{CK}^+}_{\mathsf{SIAKE}_3}(\mathcal{A}) \leq 1/2 + N^2 \cdot l \cdot q \cdot \left(4Adv^{\mathsf{1osidh}}_{\mathcal{B}} + \frac{q+1}{2^n} + negl\right).$$

Proof sketch: Obviously, U_A sends $[\mathsf{OW\text{-}CCA}, \mathsf{OW\text{-}CPA}]$ secure X, x_1, x_0 under public keys pk_{B_2} and Y to U_B. U_B responds with $[\mathsf{OW\text{-}CCA}, \mathsf{OW\text{-}CPA}]$ secure ciphertexts Y, y_1, y_0 under public keys pk_{A_1} and X_0 to U_A. The proof of wPFS security is exactly the same as that of SIAKE_2, but different for other security cases. The main observation is the same: since the underlying KEM is $[\mathsf{OW\text{-}CCA}, \cdot]$ secure, the simulator could query the strong decapsulation oracle to get the encapsulated key and session key and simulate the $\mathsf{SessionKeyReveal}$ and $\mathsf{SessionStateReveal}$. However, this is not sufficient. Take E_3 for example, given Y^*, y_1^*, y_0^* as the challenge ciphertext, the simulator obviously does not know the randomness of Y^*, but in the test session Y^* is the public key of (X, x_1, x_0). Fortunately, the underlying 1-Oracle SI-DH assumption provides this capability to encapsulate one ciphertext.

Proof. The proof for wPFS security is almost the same as that of SIAKE_2. The other cases can be easily extended or modified from the proof of E_3. We omit them and only show the proof for case E_3 here. For E_1, E_2, and E_4 when the adversary $\mathcal{A}$ makes Send query in the test session, the second part of the challenge public key is chosen by $\mathcal{A}$. For E_6, $E_{7\text{-}1}$, $E_{7\text{-}2}$, $E_{8\text{-}1}$, $E_{8\text{-}2}$, the messgaes in test session are honestly generated. Thus, the second part of the challenge public key is chosen by challenger $\mathcal{C}$.

Table 4 presents the outline of reduction.

Assumption	2-Key KEM	Cases in Table 2
SI-DDH	$[\cdot, \mathsf{OW\text{-}CPA}]$, $pk_0 = \mathfrak{g}^{\mathfrak{r}_0}$	E_5
1-Oracle SI-DH	$[\mathsf{OW\text{-}CCA}, \cdot]$ with leakage, $pk_1 = \mathfrak{g}^{\mathfrak{a}_1}$, $pk_0^* \leftarrow \mathcal{A}$	E_3, E_4
SI-DDH	$[\mathsf{OW\text{-}CCA}, \cdot]$, $pk_1 = \mathfrak{g}^{\mathfrak{a}_1}$, $pk_0^* \leftarrow \mathcal{C}$	E_6, $E_{7\text{-}2}$, $E_{8\text{-}1}$
1-Oracle SI-DH	$[\mathsf{OW\text{-}CCA}, \cdot]$ with leakage, $pk_1 = \mathfrak{g}^{\mathfrak{a}_1}$, $pk_0^* \leftarrow \mathcal{A}$	E_1, E_2
SI-DDH	$[\mathsf{OW\text{-}CCA}, \cdot]$, $pk_1 = \mathfrak{g}^{\mathfrak{a}_1}$, $pk_0^* \leftarrow \mathcal{C}$	$E_{7\text{-}1}$, $E_{8\text{-}2}$

Table 4. The outline of security reduction for SIAKE_3.

MEX E_3. In this case, X, x_1 and x_0 are sent by the adversary. Both the static secret keys of U_B and the randomness r_B are leaked to $\mathcal{A}$.

The proof proceeds via a sequence of Games 0 to 2. We denote the event that the AKE adversary $\mathcal{A}$ outputs b' such that $b = b'$ as Succ_i in Game i.

This is the original CK$^+$ game in case E_3. In the test session, r_B is leaked to the adversary assuming the test session is between U_A and U_B.

Game 1: In this game, we change the way to generate $m_1||m_0$ in the test session from $m_1||m_0 \leftarrow g(r_B, \mathfrak{b}_2)$ to $m_1||m_0 \leftarrow \{0,1\}^{2n}$. Although r_B is leaked to $\mathcal{A}$, since g is a random

oracle, $\mathcal{A}$ will not detect this change if not querying g with $r_B, \mathfrak{b}_2$. We denote Askg as the event $\mathcal{A}$ queries g with $r_B, \mathfrak{b}_2$. If event Askg happens, we can extract $\mathfrak{b}_2$ and utilize it to solve the underlying SI-DDH problem (trivially solve the underlying 1-Oracle SI-DH problem). Precisely, given $(\mathfrak{g}, \mathfrak{g}_1, \mathfrak{g}_2, j)$, $\mathcal{B}$ randomly chooses U_B as a guess of the responder in the test session with success probability $\frac{1}{2N}$. $\mathcal{B}$ sets $pk_{B_2} = \mathfrak{g}_2$. When event Askg happens, $\mathcal{B}$ just uses $\mathfrak{b}_2$ to compute and output $j \stackrel{?}{=} \mathfrak{g}_1^{\mathfrak{b}_2}$. Thus we have

$$\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1] \leq 2N \cdot \mathrm{Adv}_{\mathcal{B}}^{\mathsf{siddh}} \leq 2N \cdot \mathrm{Adv}_{\mathcal{B}}^{\mathsf{1osidh}}.$$

Game 2: In this game, we change the session key in the test session from $\hat{H}(sid, K'_A, K_B)$ to a random bit-string in $\{0,1\}^n$. Obviously, $\Pr[\mathsf{Succ}_2] = 1/2$.

We construct an algorithm $\mathcal{B}$ to solve the $[\mathsf{OW\text{-}CCA}, \cdot]$ security of $\mathsf{2KEM}_{\mathsf{1osidh}}$ with leakage, if there exists an algorithm $\mathcal{A}$ to distinguish Game 1 and Game 2.

On receiving the public key pk_1 from the $[\mathsf{OW\text{-}CCA}, \cdot]$ challenger, to simulate the CK^+ game, $\mathcal{B}$ randomly chooses two parties U_A, U_B and the i-th session as a guess of the test session with success probability $1/N^2 l$. $\mathcal{B}$ computes and sets all the static secret and public key pairs by himself for all N users U_P as both responder and initiator except the static public key for U_A as initiator.

- Specifically, $\mathcal{B}$ sets the static secret and public key pairs (pk_{A_2}, sk_{A_2}) for U_A as responder, and sets pk_1 (receiving from the $[\mathsf{OW\text{-}CPA}, \cdot]$ challenger) for U_A as initiator.
- In the test session, on receiving (X, x_1) from $\mathcal{A}$, $\mathcal{B}$ sets $pk_0^* = X$ and sends pk_0^* to $[\mathsf{OW\text{-}CCA}, \cdot]$ challenger and receives a challenge ciphertext $C^* = (Y^*, y_1, y_0)$. Then $\mathcal{B}$ returns C^* to $\mathcal{A}$ as response.
- $\mathcal{B}$ simulates all the communications and $\mathsf{SessionStateReveal}$ and $\mathsf{SessionKeyReveal}$ queries as those in Game 1 except that U_A acts as initiator (since $\mathcal{B}$ does not know sk_{A_1}).
- For those $\mathsf{SessionStateReveal}$ and $\mathsf{SessionKeyReveal}$ queries that involve U_A as initiator (for example, U_A honestly sends out X', x'_1 and receives (Y', y'_1, y'_0)). $\mathcal{B}$ queries the $\mathcal{O}$ oracle with $(X'_0; Y', y'_1, y'_0)$ provided by $[\mathsf{OW\text{-}CCA}, \cdot]$ challenger to the extract encapsulated key and maintains the consistency of $\hat{H}$ list with $\mathsf{SessionStateReveal}$ and $\mathsf{SessionKeyReveal}$ queries.
- For the test session, on receiving x_0, $\mathcal{B}$ queries $[\mathsf{OW\text{-}CCA}, \cdot]$ challenger with X to extract $h(X, X^{\mathfrak{y}})$ (where $Y^* = \mathfrak{g}^{\mathfrak{y}}$). Then $\mathcal{B}$ uses $h(X, X^{\mathfrak{y}})$ and $\mathfrak{b}_2$ to encapsulate X, x_1, x_0.

Finally, $\mathcal{B}$ checks the hash list queried by $\mathcal{A}$. If there exists some $(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, C^*, x_0, K_A, K_B)$ in the list such that K_A is the key encapsulated in (X, x_1, x_0) ($\mathcal{B}$ can compute K_A with $h(X, X^{\mathfrak{y}})$ and $\mathfrak{b}_2$), $\mathcal{B}$ chooses a random one and outputs K_B, otherwise $\perp$. Denote flag as the event that $\mathcal{A}$ explicitly queries $(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, C^*, x_0, K_A, K_B)$ to the oracle $\hat{H}$ such that K_A is the key encapsulated in (X, x_1, x_0) and K_B is the key encapsulated in C^*. If flag does not happen, $\mathcal{B}$ perfectly simulates both Game 1 and Game 2. Thus,

$$\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2] \leq \Pr[\mathsf{flag}] \leq N^2 \cdot l \cdot \mathrm{Adv}_{\mathsf{2KEM}_{\mathsf{1osidh}}}^{[\mathsf{OW\text{-}CCA}, \cdot]}(\mathcal{C}).$$

By Theorem 4, to sum up,

$$\Pr[\mathsf{Succ}_0] \leq 1/2 + N^2 \cdot l \cdot q \cdot \left(4\mathrm{Adv}_{\mathcal{B}}^{\mathsf{1osidh}} + 1/2^n + negl\right).$$

□

6 Implementation and Comparison

We implement SIAKE_2 and SIAKE_3, and compare their performance with [FSXY13], [BCNP09], [Lon18] and the lattice-based Kyber-AKE [BDK+17].

We adopt the curve SIKEp751 in SIKE [JAC17] that is proceeding the second round of NIST's post-quantum standardization. The performance is benchmarked on an Intel(R) Core i7-6567U CPU @3.30GHz processor supporting the Skylake micro-architecture. Kyer-AKE is an AKE based on lattice and others are all considered in the SIDH setting.

It is easy to see in Table 5 that SIAKE_2's bandwidth is reduced by 22.9% and SIAKE_3's bandwidth is reduced by 49.3% compared to FSXY [FSXY13] and BCNP [BCNP09,Lon18]. In Table 6, we present the performance of our protocols compared with the FSXY scheme [FSXY13] and the BCNP-Lon scheme [BCNP09,Lon18]. They are median cycles over 1,000 measurements. It shows that our 2-pass scheme is 1.12 times faster than that of FSXY and 1.3 times faster than that of BCNP-Lon. Our 3-pass AKE is more efficient since it is 1.2 times faster than FSXY and 1.4 times faster than BCNP.

Scheme	$A \to B$	$B \to A$	$A \to B$	**total(byte)**
Kyber-AKE [BDK+17]	2272	2368		4640
FSXY [FSXY13]	1160	1160	-	2320
BCNP [BCNP09,Lon18]	1160	1160	-	2320
SIAKE_2	1160	628	-	1788
SIAKE_3	596	628	32	1176

Table 5. Comparison of message sizes. We adopt the parameters chosen in [JAC17], taking into account the efficiency. "-" stands for no messages to be transmitted. The message sizes are counted in byte.

Scheme	A(**initial**)	B	A(**end**)	B(**end**)	**total**
FSXY [FSXY13]	6,238	14,779	10,124		31,141
BCNP-Lon [BCNP09]	11,146	20,092	9,563		40,801
SIAKE_2	6,828	13,917	6,641		27,386
SIAKE_3	5,966	4,429	4,922	9,575	24,892

Table 6. Comparison of cycle counts. Benchmarks are performed on a Intel(R) Core i7-6567U CPU @3.30GHz processor. Cycle counts are rounded to 10^6 cycles by taking the average of 1,000 trials.

7 Conclusion and Open Problem

In this paper, we propose two AKEs based on supersingular isogeny assumptions, one is two-pass and one is three-pass. Both of them achieve CK^+ security and support arbitrary registration in the classical random oracle model. However, to fully explain their quantum-resistance, their security in the quantum random oracle model should be analyzed. We leave it as an open problem and future work.

Acknowledgements.

Haiyang Xue is supported by the National Natural Science Foundation of China (No. 61602473, No. 61672019), and the National Cryptography Development Fund MMJJ20170116. Xiu Xu is supported by the National Natural Science Foundation of China (No.61872442). Man Ho Au is supported by the Research Grant Council of Hong Kong (Grant No. 25206317). Song Tian is supported by the National Natural Science Foundation of China (No. 61802401).

References

ABR01. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache D. (eds) CT-RSA 2001. LNCS, vol 2020, pp. 143-158. Springer, Heidelberg (2001).

BCNP09. Boyd, C., Cliff, Y., Nieto, J. M. G., Paterson, K. G.: Efficient One-Round Key Exchange in the Standard Model. In: Mu Y., Susilo W., Seberry J. (eds) ACISP 2008. LNCS, vol 5107, pp 69-83. Springer, Heidelberg (2008).

BDK+17. 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 IEEE Symposium on Security and Privacy, pp. 353-367.

BR93. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson D.R. (eds) CRYPTO 1993. LNCS, vol 773, pp. 232-249. Springer, Heidelberg (1993).

CJL17. Costello, C., Jao, D., Longa, P., Naehrig, M., Renes, J., Urbanik, D.: Efficient compression of SIDH public keys. In: Coron JS., Nielsen J. (eds) EUROCRYPT 2017. LNCS, vol 10210, pp. 679-706. Springer, Cham (2017).

CK01. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann B. (eds) EUROCRYPT 2001. LNCS, vol 2045, PP.453-474. Springer, Heidelberg (2001).

CLN16. Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for supersingular isogeny Diffie-Hellman. In: Robshaw M., Katz J. (eds) CRYPTO 2016. LNCS, vol 9814, pp. 572-601. Springer, Heidelberg (2016).

JD14. 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).

EHL18. Eisentrger, K., Hallgren, S., Lauter, K., Morrison, T., Petit, C.: Supersingular isogeny graphs and endomorphism rings: reductions and solutions. In: Nielsen, J., Rijmen, V. (eds) EUROCRYPT 2018. LNCS, vol 10822, pp 329-368. Springer, Cham (2018).

FLO17. Faz-Hernádnez, A., López, J., Ochoa-Jimenez, E., Rodríguez-Henríquez, F.: A faster software implementation of the supersingular isogeny Diffie-Hellman key exchange protocol. IEEE Transactions on Computers, 67(11), 1622-1636 (2018).

FO99. Fujisaki, E., Okamoto, T. : Secure integration of asymmetric and symmetric encryption schemes. In: Wiener M. (eds) CRYPTO 1999. LNCS, vol 1666, pp. 537-554. Springer, Heidelberg (1999).

FSXY12. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin M., Buchmann J., Manulis M. (eds) PKC 2012. LNCS, vol 7293, pp. 467-484. Springer, Heidelberg (2012).

FSXY13. 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.

FTTY18. Fujioka, A., Takashima, K., Terada, S., Yoneyama, K.: Supersingular Isogeny Diffie-Hellman Authenticated Key Exchange. In: Lee K. (eds) ICISC 2018. LNCS, vol 11396, pp. 177-195. Springer, Cham (2018).

Gal18. Galbraith, S. D.: Authenticated key exchange for SIDH. IACR Cryptology ePrint Archive 2018/266.

GPS16. Galbraith, S. D., Petit, C., Silva, J.: Identification protocols and signature schemes based on supersingular isogeny problems. In: Takagi T., Peyrin T. (eds) ASIACRYPT 2017. LNCS, vol 10624, pp. 3-33. Springer, Cham (2017).

GPST16. Galbraith, S. D., Petit, C., Shani, B., Ti, Y. B.: On the security of supersingular isogeny cryptosystems. In: Cheon J., Takagi T. (eds) ASIACRYPT 2016. LNCS, vol 10031, pp. 63-91. Springer, Heidelberg (2016).

GSW17. Guilhem, C. D. S., Smart, N. P., Warinschi, B.: Generic Forward-Secure Key Agreement Without Signatures. In: Nguyen P., Zhou J. (eds) ISC 2017. LNCS, vol 10599, pp. 114-133. Springer, Cham (2017).

GV17. Galbraith, S. D., Vercauteren, F.: Computational problems in supersingular elliptic curve isogenies. IACR Cryptology ePrint Archive 2017/774.

JAC17. Jao, D., Azarderakhsh, R., Campagna, M., et al.: Supersingular Isogeny Key Encapsulation. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions.

JKL04. Jeong, I. R., Katz, J., Lee, D. H.: One-round Protocols for Two-Party Authenticated Key Exchange. In: Jakobsson M., Yung M., Zhou J. (eds) ACNS 2004. LNCS, vol 3089, pp. 220-232. Springer, Heidelberg (2004).

JS14. Jao, D., Soukharev, V.: Isogeny-based quantum-resistant undeniable signatures. In: Mosca M. (eds) PQCrypto 2014. LNCS, vol 8772, pp. 160-179. Springer, Cham (2014).

Kra05. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup V. (eds) CRYPTO 2005. LNCS, vol 3621, pp. 546-566. Springer, Heidelberg (2005).

KAM18. Koziel, B., Azarderakhsh, R., Mozaffari-Kermani, M.: A High-Performance and Scalable Hardware Architecture for Isogeny-Based Cryptography. IEEE Transactions on Computers, Volume 67, pp.1594-1609 (2018).

KLM15. 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. In Workshop on Cybersecurity in a Post-Quantum World (2015).

Leg16. LeGrow, J.: Post-Quantum Security of Authenticated Key Establishment Protocols. Master's thesis, University of Waterloo (2016).

Lon18. Longa, P.: A Note on Post-Quantum Authenticated Key Exchange from Supersingular Isogenies. IACR Cryptology ePrint Archive 2018/267.

LLM07. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo W., Liu J.K., Mu Y. (eds) ProvSec 2007. LNCS, vol 4784, pp. 1-16. Springer, Heidelberg (2007).

MQV95. Menezes, A., Qu, M., Vanstone, S.: Some new key agreement protocols providing mutual implicit authentication. Selected Areas in Cryptography (1995).

MTI86. Matsumoto, T., Takashima, Y., Imai, H.: On seeking smart public-key-distribution systems. IEICE TRANSACTIONS (1976-1990), 69(2), 99-106 (1986).

Oka07. Okamoto, T.: Authenticated key exchange and key encapsulation in the standard model. In: Kurosawa K. (eds) ASIACRYPT 2007. LNCS, vol 4833, pp. 474-484. Springer, Heidelberg (2007).

STW12. Sun, X., Tian, H., Wang, Y.: Toward quantum-resistant strong designated verifier signature from isogenies. In INCoS 2012, pp. 292-296.

UJ18. Urbanik, D., Jao, D.: SoK: The problem landscape of SIDH. IACR Cryptology ePrint Archive 2018/336.

XLL+18. Xue, H., Lu, X., Li, B., Liang, B., He, J.: Understanding and Constructing AKE via Double-Key Key Encapsulation Mechanism. In: Peyrin T., Galbraith S. (eds) ASIACRYPT 2018. LNCS, vol 11273, pp. 158-189. Springer, Cham (2018).

YAJ17. Yoo, Y., Azarderakhsh, R., Jalali, A., Jao, D., Soukharev, V.: A post-quantum digital signature scheme based on supersingular isogenies. In: Kiayias A. (eds) FC 2017. LNCS, vol 10322, pp. 163-181. Springer, Cham (2017).