Strongly Secure Authenticated Key Exchange from Supersingular Isogeny

Xiu Xu[1,2], Haiyang Xue[1,2⋆], Kunpeng Wang[1,2], Bei Liang[3], Song Tian[1,2], Wei Yu[1,2]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, China
[2] Data Assurance and Communication Security Research Center, Beijing, China
[3] Chalmers University of Technology, Gothenburg, Sweden
{xuxiu,xuehaiyang,wangkunpeng,tiansong,yuwei}@iie.ac.cn, lbei@chalmers.se

Abstract. In this paper, we study the authenticated key exchange (AKE) based on supersingular isogeny problems which are believed to be difficult for quantum computers. We first propose a 3-pass AKE based on 1-Oracle SIDH assumption whose soundness is guaranteed by a strictly limited gap problem. The 1-Oracle SIDH and the limited gap assumptions are of independent interest. To enhance the soundness, we also propose a 2-pass AKE based on standard SIDH assumption, which involves more bandwidth. Both the 3-pass and 2-pass AKE protocols allow arbitrary registrant of public keys, and achieve CK$^+$ security (a security model which covers wPFS security, KCI attack, and MEX attack). Our results move us one step forward to the target set by Galbraith of looking for new techniques to design and prove security of AKE in the SIDH setting with the widest possible adversarial goals.

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

1 Introduction

Authenticated Key Exchange. Key exchange (KE) is a fundamental cryptographic primitive, which enables two parties to agree on a common shared key over a public but possibly insecure channel. The classical Diffie-Hellman (DH) key exchange protocol [7] without authentication is vulnerable to the man-in-the-middle attack. Many studies have investigated how to achieve KE protocols that provide authentication in secure models [3, 5, 11, 29] and how to implement authenticated key exchange (AKE) with high efficiency [2, 11, 12, 20, 29–31] based on classical assumptions. A plenty of security models have been proposed, including BR model [3], CK model [5] and eCK model [29]. CK$^+$ security model known as one of the "strongest" and most "desirable" security notions [24] for AKE is reformulated by Fujioka *et al.* [11]. The CK$^+$ model not only covers

[⋆] Corresponding author email: xuehaiyang@iie.ac.cn 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). Therefore, CK$^+$ model can be theoretically considered as a complete version of the AKE security model since it currently covers the widest possible variety of adversarial methods in some sense.

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. Based on the problem of computing the isogeny between supersingular elliptic curves, which is believed to be difficult, Jao and De Feo [8] proposed a supersingular isogeny Diffie-Hellman key exchange (SIDH). There are many following works which mainly focus on the computational efficiency [6, 9, 25], key compression [4], adaptive attacks on SIDH [16, 22], the relationship of underlying complexity problems [18, 34], signature schemes [15, 21, 33, 35] and its standardization [19, 26].

The SIDH protocol is analogous to the traditional Diffie-Hellman protocol and so it is vulnerable to the man-in-the-middle attack. As in the survey where Galbraith concluded [14], to the best of our knowledge, there are few papers investigating the natural problems of designing AKE schemes from the basic SIDH primitive. A general approach to build AKE is to sign each party's round messages with respect to their long-term public keys by using digital signatures. Unfortunately, there is no practical signature based on the hardness of constructing an isogeny between two isogenous elliptic curves [15, 35]. Therefore, it is essential to explore on designing *implicit* AKE from SIDH primitives. However, as Galbraith [14] pointed out, 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, such as MQV [30] and HMQV [24], require a richer algebraic structure that the supersingular isogeny does not have.
- The protocols involving long-term/static secret keys are vulnerable to the adaptive attack [16] 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. [16] show that a malicious adversary Bob can send (E_Y, R', S') with specified points R' and S', and gradually learn Alice's static secret key.
- The gap assumption that holds in the discrete logarithm setting is crucial for security proof. But the gap assumption does not hold in the SIDH setting when polynomial queries are submitted to an *unlimited* decisional solver.

The State of Art of SIDH AKE. Recently, there are many exciting results both on the generic and non-generic constructions of AKE in the SIDH case [13, 14, 28]. Galbraith [14] and Longa [28] 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, Gonzlez Nieto and Paterson [2] (abbreviated as BCNP scheme), by Fujioka, Suzuki, Xagawa and Yoneyama [12] (abbreviated as FSXY scheme) and by Guilhem, Smart and Warinschi [17] (abbreviated as GSW scheme), to the SIDH setting by inserting an IND-CCA secure KEM based on SIDH. However, these transformations lead to either more isogeny computations or more rounds of communication. The detailed analyses are examined and summarized in the table 1 of [14]. Here we make a more concrete comparison among these resulted SIDH schemes in Table 1.

With respect to non-generic constructions, Galbraith proposed two SIDH-AKE protocols [14], one of which is based on the Jeong-Katz-Lee [20] 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. [13] gave two Diffie-Hellman like isogeny-based AKEs, we call them 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. Whereas, all of these schemes only satisfy the security with limited adversarial abilities, like wPFS security (details are given in section 1.3). Several recognized attacks are not taken into account, including arbitrary registrant for static public keys, KCI attacks, or MEX attacks. In an AKE system, the adversary-controlled parties may register arbitrary public keys and arbitrary registrant allows any party to register arbitrary valid keys (even the same key as some other party) without any validity checks. In fact, the arbitrary registrant for the static public key is not allowed for Gal 1-2 and FTTY 1-2 schemes. 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 SIDH setting, and give full analysis of AKE that includes the widest possible adversarial goals."*, a quote from Galbraith [14], is the main concern in SIDH AKEs area. In this paper, we are motivated to address such an open problem.

1.1 Our Contributions.

In this paper, we present a 3-pass and a 2-pass elegant AKE schemes in the SIDH setting and prove that they allow arbitrary registrant and are secure in the CK$^+$ model.

1. To prepare for 3-pass AKE, we investigate the soundness of the hashed decisional SIDH problem called 1-Oracle SIDH problem, where the adversary is allowed to query a one-time hashed computational SIDH oracle. We reduce the hardness of 1-Oracle SIDH problem to a computational SIDH assumption (called 1-gap SIDH assumption) with a strictly limited decisional oracle which allows queries with ***only one*** $(E_X, R_2, S_2 \in E_X[l_2^{e_2}])$ (that is asked initially) but different curves E_Z to the decisional SIDH (DSIDH) oracle.

2. We propose a strongly secure key encapsulation mechanism (KEM) based on supersingular isogeny, which serves as the core building block of our AKE schemes. Our KEM is *chosen public-key chosen ciphertext*[4] (CPCCA) secure under the standard DSIDH assumption. Based on the 1-Oracle SIDH assumption, it is still CPCCA secure even if some information about the challenge ciphertext is leaked.
3. Equipped with the 1-Oracle SIDH assumption and a strongly secure KEM as the building blocks, we propose a 3-pass AKE $\mathsf{AKE}_{\mathsf{SIDH\text{-}3}}$ and prove that it allows arbitrary registrant and is secure in the CK$^+$ model.
4. To enhance the security of AKE, we also propose a 2-pass AKE $\mathsf{AKE}_{\mathsf{SIDH\text{-}2}}$ and prove its security based on standard DSIDH assumption.

As shown in Table 1, both the 3-pass and 2-pass AKE schemes have advantages compared with existing works. And they achieve great trade-offs between security and efficiency since our security covers the widest possible adversarial goals including arbitrary registrant. We also implement some typical schemes and give a comparison in Section 5. According to the result of our experiment, the 3-pass scheme decreases the bandwidth by 49.3% and is 1.2 times faster than the generic construction in [12] without loss of security. The 2-pass AKE scheme narrows the bandwidth by 23% and is 1.12 times faster.

Scheme	**Key Reg.**	**Model**	**Assum.**	**wPFS**	**KCI**	**MEX**	**Rd**	**Init isog**	**Resp isog**	**Mess Size**
SIDH [8]	-	DSIDH	-	$\times$	$\times$	$\times$	2	2	2	72λ
Gal 1 [14]	Honest	CSIDH	CK	$\checkmark$	$\times$	$\times$	2	3	3	108λ
Gal 2 [14]	Honest	CSIDH	BR	$\checkmark$	$\checkmark$	$\times$	2	4	4	108λ
FTTY 1 [13]	Honest	DSIDH	CK	$\checkmark$	$\times$	$\times$	1	3	3	72λ
FTTY 2 [13]	Honest	di-DSIDH	CK$^+$	$\checkmark$	$\checkmark$	$\checkmark$	1	5	5	72λ
GSW [17]	Arbi.	DSIDH	CK	$\checkmark$	$\times$	$\times$	3	6	6	186λ
BCNP-Lon [2, 28]	Arbi.	DSIDH	CK	$\checkmark$	$\checkmark$	$\times$	2	6	6	148λ
FSXY-Lon [12, 28]	Arbi.	DSIDH	CK$^+$	$\checkmark$	$\checkmark$	$\checkmark$	2	7	6	148λ
$\mathsf{AKE}_{\mathsf{SIDH\text{-}2}}$	Arbi.	DSIDH	CK$^+$	$\checkmark$	$\checkmark$	$\checkmark$	2	6	5	114λ
$\mathsf{AKE}_{\mathsf{SIDH\text{-}3}}$	Arbi.	1-OSIDH	CK$^+$	$\checkmark$	$\checkmark$	$\checkmark$	3	5	5	80λ

Table 1. Comparison of existing AKE protocols on supersingular isogeny. SIDH is an unauthenticated scheme. **Key Reg.** represents registering the static public key. "Arbi" means arbitrary registrant is allowed while "Honest" means only honest registrant is allowed. **Assump.** is the abbreviation of assumptions. **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. −" indicates that the scheme does not consider this property.

[4] inspired by [32], we specify this strong security for supersingular isogeny. Although we use the same name with [32], we note that the adversary is different, where the adversary can choose part of the challenge public key while [32] does not.

1.2 Our Techniques

Our core ideas and techniques are illustrated in Figure 1. E_0 is the starting curve. 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 collaborated computing curves.

The SIDH key exchange works as follows: U_A chooses a secret and 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 and 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 for SIDH with the static and ephemeral component is that every user registers a static public key where U_A's static public key is $(E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2))$ and U_B's static public key is $(E_{B_2}, \phi_{B_2}(P_1), \phi_{B_2}(Q_1))$.

$$\begin{tikzcd}
 & & & \underline{E_{A_1B_2}} & & & \\
 & & E_{A_1} \arrow[dl] \arrow[ur, dashed, red] & & E_{B_2} \arrow[dr] \arrow[ul, dashed, red] & & \\
 & \underline{E_{A_1Y}} \arrow[dl, "h(j(\cdot))"'] & & \underline{E_0} \arrow[ul] \arrow[ur] \arrow[dl] \arrow[dr] \arrow[ul, dashed, red, bend left=10] \arrow[ur, dashed, red, bend right=10] \arrow[dl, dashed, red, bend right=10] \arrow[dr, dashed, red, bend left=10] & & \underline{E_{XB_2}} \arrow[dr, "h(j(\cdot))"] & \\
m_1 \arrow[dr] & & E_Y \arrow[ul] \arrow[dl] \arrow[dr] \arrow[dr, dashed, red, bend right=10] & & E_X \arrow[ur] \arrow[dr] \arrow[dl] \arrow[dl, dashed, red, bend left=10] & & n_1 \arrow[dl] \\
 & K_B & & \underline{E_{XY}} \arrow[dl, "h(X, j(\cdot))"'] \arrow[dr, "h(Y, j(\cdot))"] & & K_A & \\
 & & m_2 \arrow[ul] & & n_2 \arrow[ur] & &
\end{tikzcd}$$

Fig. 1. Illustration of the core idea of our AKEs. The red dashed lines illustrate the core ideas of Gal 1 scheme [14]. E_0 is the base curve.

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}. But it is risky to take $E_{A_1B_2}$ into account. Let us recall the adaptive attack from Galbraith, Petit, Shani and Ti [16]. 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 resulted session key. As shown in Figure 1 with dashed lines, Galbraith [14] involves $E_{A_1B_2}$ and E_{XY} for the session key. Under the adaptive attack [16], 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 still takes effect 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 result session key, the adversary could still extract one bit of the static secret key. The problem comes down to how to check the "validity" of $Y = (E_Y, R, S)$. Our solution is to employ the "re-encryption" technique used in Fujisaki-Okamoto (FO) transformation [10]. Precisely, $C = (Y, y_1, y_2)$ is the ciphertext under public key E_{A_1} and X, where $Y = (E_0/\langle P_2 + [y]Q_2\rangle, \phi_Y(P_1), \phi_Y(Q_1))$, $y_1 = h(j(E_{A_1Y})) \oplus m_1$, $y_2 = h(j(E_{XY})) \oplus m_2$ and $y = g(m_1, m_2)$ for a hash function g, and the encapsulated key is $K_B = H(m_1, m_2)$. 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 so that the adaptive attack fails to work. The requirement is symmetric: U_A computes (X, x_1, x_2) as the ciphertext under public key E_{B_2} and Y, where $X = (E_0/\langle P_1 + [x]Q_1\rangle, \phi_X(P_2), \phi_X(Q_2))$, $x_1 = h(j(E_{XB_2})) \oplus n_1$, $x_2 = h(j(E_{XY})) \oplus n_2$ and $x = g(n_1, n_2)$, and encapsulated key is $K_A = H(n_1, n_2)$. Therefore, extracting the session key from K_A and K_B rather than E_{A_1Y}, E_{XB_2} and E_{XY} prevents it from the adaptive attack.

Although the CCA secure KEM with "re-encryption" avoids the adaptive attack, 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_2^*), while the capability of querying the *session state* and *session key* of non-test sessions implies that it's also allowed to query the decapsulation oracle which decapsulates the ciphertext under several public keys X' (not only the challenge public key). We integrate such an attack manner as the chosen public key and chosen ciphertext attack (CPCCA), inspired by Okamoto [32]. Although the CPCCA adversary is much stronger than CCA adversary, it is not complicated to make our CCA secure KEM be secure against the CPCCA adversary in the random oracle model. What we only need to do is to put the public key in the hashing step when generating the encapsulated key. Precisely, K_B encapsulated in (Y, y_1, y_2) is $H(X, m_1, m_2)$, while K_A encapsulated in (X, x_1, x_2) is $H(Y, n_1, n_2)$.

We almost figure out a resolution except that both X and Y have two functionalities. 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_2) is computed. On the other hand X is part of the ciphertext (X, x_1, x_2) in which K_A is encapsulated under public key (pk_{B_2}, Y). Precisely, in the test session $(X = (E_X, R_2, S_2), x_1, x_2)$ is sent by AKE adversary $\mathcal{A}$, and the simulator $\mathcal{S}$ gets challenge ciphertext (Y^*, y_1^*, y_2^*) from the CPCCA 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_2).

We propose two solutions for this problem. One method is to strengthen the underlying assumptions as 1-Oracle SIDH assumption such that $h(j(E_X/\langle R_2 + [y]S_2\rangle))$ could be leaked. The other one 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_2) is computed, while X is part of the ciphertext (X, x_1) under public key E_{B_2} (we omit Y).

The two solutions result in our 3-pass AKE in section 4.1 and 2-pass AKE in section 4.2, respectively.

- Solution 1: We enhance the underlying DSIDH assumption to the 1-Oracle SIDH assumption to allow the leakage of $h(j(E_X/\langle R_2 + [y]S_2\rangle))$. The 1-Oracle SIDH assumption can be considered as a hashed DSIDH assumption where a one-time hashed CSIDH 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 $1 \leq u \leq l_2^{e_2}$ 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 our 3-pass AKE.
- Solution 2: We add an extra X_0 to take the position of X as part of the public key (E_{A_1}, X_0) under which the ciphertext (Y, y_1, y_2) 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 2 leads to our 2-pass AKE.

1.3 Related Works and Their Analysis.

Galbraith [14] proposed two SIDH variants of AKE, namely Gal 1 from Jeong-Katz-Lee protocol [20] and Gal 2 from NAXOS protocol [29]. Considering the adaptive attack on static secret keys, Gal 1 protocol only allows honest registrant of static public keys and it is also vulnerable to the KCI attack. So far, there hasn't been any concrete MEX attack on Gal 1, neither has there been 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 registrant 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. [13] gave two Diffie-Hellman like isogeny-based AKEs, 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 registrant and can not resist the KCI attack. FTTY 2 is secure in CK+ secure model, but it also only allows honest registrant.

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 aim 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 [16], 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 by adversary with 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 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 does not allow arbitrary registrant. Here we correct the conjecture made by Fujioka et al. [13] that FTTY 2 seems to allow arbitrary registrant. Gal 2 doesn't 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 one party of the test session at least. 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}, 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 Key Exchange

The SIDH protocol [8] inherits the construction of general Diffie-Hellman protocol, but it has an obvious technical difference. The endomorphism of a supersingular elliptic curve is isomorphic to an order in a quaternion algebra, which is not communicative. Thus, extra information must be transferred as part of the ciphertext in order to get the same shared key.

We reiterate the key exchange protocol briefly and adopt the notations in [8] for the most part. First, choose a prime of the form $p = l_1^{e_1} l_2^{e_2} \cdot f \pm 1$ where l_1 and l_2 are small primes and f is a cofactor. We fix a supersingular elliptic curve E_0 defined over $\mathbb{F}_{p^2}$. The cardinality of E_0 is $|E_0(\mathbb{F}_{p^2})| = (l_1^{e_1} l_2^{e_2} \cdot f)^2$. Furthermore, $E_0[l_1^{e_1}] = \mathbb{Z}/l_1^{e_1}\mathbb{Z} \oplus \mathbb{Z}/l_1^{e_1}\mathbb{Z} = \langle P_1, Q_1 \rangle$, $E_0[l_2^{e_2}] = \mathbb{Z}/l_2^{e_2}\mathbb{Z} \oplus \mathbb{Z}/l_2^{e_2}\mathbb{Z} = \langle P_2, Q_2 \rangle$. The above parameters are public. Then A secretly chooses two random elements $m_A, n_A \in \mathbb{Z}/l_1^{e_1}\mathbb{Z}$, not both divisible by l_1, and computes an isogeny

$\phi_A : E_0 \to E_A$ with kernel $\langle [m_A]P_1 + [n_A]Q_1 \rangle$. A sends E_A to B with two points $\phi_A(P_2), \phi_A(Q_2)$. Similarly, B secretly chooses two random elements $m_B, n_B \in \mathbb{Z}/l_2^{e_2}\mathbb{Z}$, not both divisible by l_2, and computes an isogeny $\phi_B : E_0 \to E_B$ with kernel $\langle [m_B]P_2 + [n_B]Q_2 \rangle$. B sends E_2 to A with two points $\phi_B(P_1), \phi_B(Q_1)$. Upon receiving from B, A computes an isogeny $\phi'_A : E_B \to E_{BA}$ with kernel $\langle [m_A]\phi_B(P_1) + [n_A]\phi_B(Q_1) \rangle$. B computes an isogeny $\phi'_B : E_A \to E_{AB}$ with kernel $\langle [m_B]\phi_A(P_2) + [n_B]\phi_A(Q_2) \rangle$. When it comes to computing the shared key, we take B as an example. We denote $\langle [m_B]P_2 + [n_B]Q_2 \rangle = \langle R_B \rangle$ and $\langle [m_A]P_1 + [n_A]Q_1 \rangle = \langle R_A \rangle$ for convenience.

$$
\begin{aligned}
E_{AB} &= E_A/\langle [m_B]\phi_A(P_2) + [n_B]\phi_A(Q_2) \rangle = E_A/\langle \phi_A([m_B]P_2 + [n_B]Q_2) \rangle \\
&= (E_0/\langle R_A \rangle)/(\langle R_A, R_B \rangle / \langle R_A \rangle).
\end{aligned}
$$

Due to the isomorphism theorem, $(E_0/\langle R_A \rangle)/(\langle R_A, R_B \rangle/\langle R_A \rangle) = E_0/\langle R_A, R_B \rangle$. Similarly, we know $E_{BA} = E_0/\langle R_A, R_B \rangle$. Therefore, A and B share the same j-invariant as $j(E_{AB}) = j(E_{BA})$.

According to Lemma 1 in [16], we know that for some $(m, n) \in \mathbb{Z}^2$ (not simultaneously even), we have that $(m, n) \sim (1, a)$ or $(m, n) \sim (a, 1)$ for some $a \in \mathbb{Z}$. We call this private key normalized. Throughout the rest of this paper, we note that the private keys are normalized and without loss of generality, we fall into the former case.

2.2 CK$^+$ Security Model

We recall the CK$^+$ model introduced by [24] and later refined by [11], which is a CK model [5] 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 [5] 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. [29] 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. The adversary $\mathcal{A}$ is modeled in the following 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.
- $\mathsf{SessionKeyReveal}(\mathsf{sid})$: if the session sid is completed, $\mathcal{A}$ obtains the session key SK for sid.
- $\mathsf{SessionStateReveal}(\mathsf{sid})$: The adversary $\mathcal{A}$ 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. This query can be issued at any stage to a completed, fresh and unexpired session *sid.* A bit b is picked randomly. If $b = 1$, the oracle generates a random value in the key space; if $b = 0$, it reveals the session key. The adversary can continue to issue queries except that it cannot expose the test session. The adversary wins the game if the session is fresh and the guess of the adversary is correct, i.e., $b' = b$. The advantage of the adversary $\mathcal{A}$ is defined as $\mathsf{Adv}_{\Pi}^{\mathrm{CK}^+}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}$.

Definition 1. *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 1, and these ten cases cover wPFS, resistance to KCI, and MEX as follows: E_1, E_4, $E_{7\text{-}1}$, $E_{7\text{-}2}$, $E_{8\text{-}1}$ and $E_{8\text{-}2}$ capture KCI, since the adversary obtains either only the static secret key of one party or both the static secret key of one party and the ephemeral secret key of the other party of the test session. E_5 captures wPFS. E_2, E_3 and E_6 capture MEX, since the adversary obtains the ephemeral secret key of one party of the test session at least.

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 (sk_B) means the static secret key of A (B). ek_A (ek_B) is the ephemeral secret key of A (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.

3 1-Oracle SIDH Assumptions and Implied Strongly Secure KEM

In this section, we recall the complexity problems related to supersingular isogeny, where the first five are standard assumptions, the sixth is straightforwardly followed by adding a hash function and the last two are new. We analyze their relations and plausibility. After that we propose a strongly secure CPCCA KEM, which is the core build block for our AKEs, from the standard SIDH assumption and 1-Oracle assumption.

3.1 Standard SIDH Problems and Their Relations

Recall that E_0 is the base curve and there are fixed basis pairs $\langle P_1, Q_1\rangle = E_0[l_1^{e_1}]$ and $\langle P_2, Q_2\rangle = E_0[l_2^{e_2}]$.

Definition 2 (SI isogeny problem [18]). *The A-SI problem is given $(E_0, P_1, Q_1, P_2, Q_2;\ E_A, R_2, S_2)$ where $R_2, S_2 \in E_A[l_2^{e_2}]$, to find $\phi_A : E_0 \to E_A$ of degree $l_1^{e_1}$ such that $R_2 = \phi_A(P_2)$ and $S_2 = \phi_A(Q_2)$. The B-SI problem is given $(E_0, P_1, Q_1, P_2, Q_2;\ E_B, R_1, S_1)$ where $R_1, S_1 \in E_B[l_1^{e_1}]$, to find $\phi_B : E_0 \to E_B$ of degree $l_2^{e_2}$ such that $R_1 = \phi_B(P_1)$ and $S_1 = \phi_B(Q_1)$.*

Definition 3 (Decisional SI problem [18]). *Let E_A be any elliptic curve. The Decisional A-SI problem is given $(E_0, P_1, Q_1, P_2, Q_2;\ E_A, R_2, S_2)$ where $R_2, S_2 \in E_A[l_2^{e_2}]$, to determine whether or not there exists an isogeny $\phi_A : E_0 \to E_A$ of degree **dividing** $l_1^{e_1}$ such that $R_2 = \phi_A(P_2)$ and $S_2 = \phi_A(Q_2)$. The Decisional B-SI can be defined similarly.*

Definition 4 (Computational SIDH (CSIDH) problem [8]). *Let $\phi_A : E_0 \to E_A$ be an isogeny whose kernel is $G_A = \langle P_1 + [a]Q_1\rangle$, and let $\phi_B : E_0 \to E_B$ be an isogeny whose kernel is $G_B = \langle P_2 + [b]Q_2\rangle$. Given $(E_0, P_1, Q_1, P_2, Q_2; E_A, \phi_A(P_2), \phi_A(Q_2);\ E_B, \phi_B(P_1), \phi_B(Q_1))$, find the j-invariant of $E_0/\langle G_A, G_B\rangle$.*

Definition 5 (Decisional SIDH (DSIDH) problem [8]). *Given $(E_0, P_1, Q_1, P_2, Q_2; E_A, \phi_A(P_2), \phi_A(Q_2); E_B, \phi_B(P_1), \phi_B(Q_1); E_C)$, where E_A, E_B, G_A, G_B are that in CSIDH problem, E_C is computed as $E_C \cong E_0/\langle G_A, G_B\rangle$ or $E_C \cong E_0/\langle P_1 + [a']Q_1, P_2 + [b']Q_2\rangle$ with probability 1/2, where a' (respectively b') is chosen at random from $\mathbb{Z}/l_1^{e_1}\mathbb{Z}$ (respectively $\mathbb{Z}/l_2^{e_2}\mathbb{Z}$) and not both divisible by l_1 (respectively l_2). The DSIDH problem is to decide how E_C is computed.*

Definition 6 (A/B-DSIDH problem [34]). *Given $(E_0, P_1, Q_1, P_2, Q_2)$. Suppose that E_B and $\phi_B(P_1), \phi_B(Q_1) \in E_B[l_1^{e_1}]$ are known, then given a curve E_X, a basis pair $R_2, S_2 \in E_X[l_2^{e_2}]$ and a curve E_Z, determine whether the tuple (E_X, E_B, E_Z) is a valid SIDH tuple, in the sense that there is a map $\psi_X : E_0 \to E_X$ of degree (dividing) $l_1^{e_1}$ such that $\psi_X(P_2) = R_2, \psi_X(Q_2) = S_2$, $E_Z = E_0/\langle \mathsf{ker}\psi_X, G_B\rangle$. This is the A-DSIDH problem.*

Given $(E_0, P_1, Q_1, P_2, Q_2)$. Suppose that E_A and $\phi_A(P_2), \phi_A(Q_2) \in E_A[l_2^{e_2}]$ are known, then given a curve E_Y, a basis pair $R_1, S_1 \in E_Y[l_1^{e_1}]$ and a curve E_Z, determine whether the tuple (E_A, E_Y, E_Z) is a valid SIDH tuple, in the sense that there is a map $\psi_Y : E_0 \to E_Y$ of degree (dividing) $l_2^{e_2}$ such that $\psi_Y(P_1) = R_1, \psi_Y(Q_1) = S_1$ and $E_Z = E_0/\langle G_A, \mathsf{ker}\psi_Y\rangle$. This is the B-DSIDH problem.

Remark 1: Note that in [34] the A-DSIDH problem is to decide whether there is a map $\psi_X : E_0 \to E_X$ of degree *dividing* $l_1^{e_1}$ such that $\psi_X(P_2) = R_2$ and $\psi_X(Q_2) = S_2$. In the following we define it to decide whether there is a map of degree *equal to* $l_1^{e_1}$. The same holds for B-SIDH problem.

The above are standard problems and previous works have analyzed their relations. Galbraith and Vercauteren [18] show that the computational SI isogeny is equivalent to the decisional SI problem since that l_1 queries to the decisional SI solver will help to decide the result of the first $e_1 - 1$ steps. Precisely, let $u \in \mathbb{Z}$ be such that $ul_1 \equiv 1 \mod l_2^{e_2}$ and $R_2' = [u]\psi(R_2), S_2' = [u]\psi(S_2)$, given the computational SI instance $(E_0, P_1, Q_1, P_2, Q_2; E_A, R_2, S_2)$, choose one l_1-isogeny $\psi : E_A \to E'$ and query the decisional SI solver on $(E_0, P_1, Q_1, P_2, Q_2; E', R_2', S_2')$ to decide the last l_1-isogeny ψ. Then querying the decisional SI solver polynomial times with polynomial different (E', R_2', S_2') will help to determine the path from E_0 to E_A.

Urbanik and Jao [34] show that under randomized polynomial time reductions, a solver of the A-SI problem is equivalent to a solver which solves both the CSIDH and A-DSIDH problems.

Galbraith *et al.* [16] proposed an adaptive attack which intends to attack the SIDH key exchange if $(E_A, \phi_A(P_2), \phi_A(Q_2))$ is a static key. They point out that the known pubic key validation methods are insufficient and the attacker could learn one bit of the secret key of ϕ_A by querying the B-DSIDH solver once. After polynomial times of queries with polynomial different $(R_1, S_1) \in E_Y[l_1^{e_1}]$ and polynomial different curves E_Z to the decisional B-DSIDH problem, the secret key of ϕ_A will be extracted bit-by-bit. Thus the CSIDH problem is equivalent to the A-DSIDH or B-DSIDH problem under reduction for randomized polynomial time at the same time.

As we have mentioned in Remark 1, Fujioka et al. also set the A/B-DSIDH problem as to decide whether there is a map $\psi_X : E_0 \to E_X$ of degree equal to $l_1^{e_1}$, rather than *dividing* $l_1^{e_1}$, and call it degree-sensitive. They also extend Galbraith and Vercauteren's work [18] to prove the equivalence of CSIDH with degree-sensitive A/B-DSIDH problem.

3.2 1-Oracle SIDH and 1-gap SIDH Problems

We first give a straightforward variant of the DSIDH problem by adding a hash function, then propose the 1-Oracle SIDH problem and reduce its soundness to the hardness of solving the 1-gap problem.

In the supersingular isogeny area, the classical "gap" problem does not hold, namely computational SIDH problem (resp. SI) does not hold [16, 18] if queries with polynomial different $(R_2, S_2) \in E_X[l_2^{e_2}]$ and polynomial different curves E_Z to the decisional A/B-DSIDH problem (resp. Decisional SI) solver are allowed.

In spite of the failure to have the classical "gap" assumption in supersingular isogeny setting, it is promising to have the "gap" assumption with a strictly limited decisional oracle. Precisely, we believe that the CSIDH is still hard even if queries with only one $(E_X, R_2, S_2 \in E_X[l_2^{e_2}])$ (that may be chosen by CSIDH solver) and polynomial different curves E_Z to the decisional A-DSIDH problem are allowed. The certainty comes from the fact that the queries to the decisional

A-DSIDH solver with only one curve and one pair of basis leak at most log $poly(\lambda)$ bits of secret key of ϕ_B, which would not dramatically harm the soundness of CSIDH problem.

Definition 7 (Hashed DSIDH problem). *Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function. Given $(E_0, P_1, Q_1, P_2, Q_2; E_A, \phi_A(P_2), \phi_A(Q_2); E_B, \phi_B(P_1), \phi_B(Q_1); h)$, where E_A, E_B, G_A, G_B are that in CSIDH problem and h is computed as $h = H(A, j(E_{AB}))$ where $A = (E_A, \phi_A(P_2), \phi_A(Q_2))$ and $E_{AB} \cong E_0/\langle G_A, G_B\rangle$ or $h \leftarrow \{0,1\}^\lambda$ with probability 1/2. The Hashed DSIDH problem is to decide how h is computed.*

1-Oracle SIDH Assumption. Let $\phi_B : E_0 \to E_B$ be an isogeny with kernel $G_B = \langle P_2 + [b]Q_2\rangle$. And $\phi_B(P_1), \phi_B(Q_1) \in E_B[l_1^{e_1}]$ are known. Let the $\mathcal{O}_B$ be a SIDH key exchange oracle that given the input of a curve E_X and a basis pair $\langle R_2, S_2\rangle \in E_X[l_2^{e_2}]$, computes and outputs a curve $E_Z = E_X/\langle R_2 + [b]S_2\rangle$.

Let $\mathcal{H}_B$ be a one-time Hashed SIDH oracle which given the input of a curve E_X and a basis pair $\langle R_2, S_2\rangle \in E_X[l_2^{e_2}]$, outputs $H(X, j(E_Z))$ where $X = (E_X, R_2, S_2)$ and $E_Z \leftarrow \mathcal{O}_B(E_X, R_2, S_2)$. Suppose that given $(E_0, P_1, Q_1, P_2, Q_2; E_A, \phi_A(P_2), \phi_A(Q_2); E_B, \phi_B(P_1), \phi_B(Q_1))$, the adversary's goal is to compute $H(A, j(E_{AB}))$ where $A = (E_A, \phi_A(P_2), \phi_A(Q_2))$ and $E_{AB} = E_0/\langle G_A, G_B\rangle$. Now, as long as the one-time oracle $\mathcal{H}_B$ doesn't allow $(E_A, \phi_A(P_2), \phi_A(Q_2))$ to be queried, this one-time oracle seems useless. We formalize it as follows.

Definition 8 (1-Oracle SIDH problem). *Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function. Given $(E_0, P_1, Q_1, P_2, Q_2; E_A, \phi_A(P_2), \phi_A(Q_2); E_B, \phi_B(P_1), \phi_B(Q_1); h)$, where E_A, E_B, G_A, G_B are that in CSIDH problem and h is computed as $h = H(A, j(E_{AB}))$ where $A = (E_A, \phi_A(P_2), \phi_A(Q_2)), B = (E_B, \phi_B(P_1), \phi_B(Q_1))$, $E_{AB} \cong E_0/\langle G_A, G_B\rangle$ or $h \leftarrow \{0,1\}^\lambda$ with probability 1/2. The 1-Oracle SIDH problem is to decide how h is computed when the adversary $\mathcal{A}$ can query one-time Hashed SIDH oracle $\mathcal{H}_B$ with $(E_X, R_2, S_2) \neq (E_A, \phi_A(P_2), \phi_A(Q_2))$. The advantage of $\mathcal{A}$ is*

$$\begin{aligned}\mathbf{Avd}_{\mathcal{A}}^{\textit{1-OSIDH}} = \Pr[&\mathcal{A}^{\mathcal{H}_B}\big(A, B, H(A, j(E_{AB}))\big) = 1] - \\ &\Pr[\mathcal{A}^{\mathcal{H}_B}\big(A, B, h \leftarrow \{0,1\}^\lambda\big) = 1].\end{aligned}$$

We emphasize that the adversary is allowed to query the Hashed SIDH oracle $\mathcal{H}_B$ only once and $(E_X, R_2, S_2) \neq (E_A, \phi_A(P_2), \phi_A(Q_2))$. If he can query for polynomial times, then the 1-Oracle SIDH problem can be solved using the adaptive attack in [18].

Please also note that the hash function involves $(E_A, \phi_A(P_2), \phi_A(Q_2))$ or (E_X, R_2, S_2) as input besides the j-invariant. Otherwise the 1-Oracle SIDH problem is not hard, since the attacker can choose $E_X = E_A$ and some R_2, S_2 such that they give the same j-invariant, and query $\mathcal{H}_B$ with (E_X, R_2, S_2). For example, as $\langle R_2 + [y]S_2\rangle = \langle [u]R_2 + [y][u]S_2\rangle$ for any integer $1 \leq u \leq l_2^{e_2}$ and coprime to l_2, given $(E_A, \phi_A(P_2), \phi_A(Q_2))$, the attacker sets $E_X = E_A$, $R_2 = [u]\phi_A(P_2)$ and $S_2 = [u]\phi_A(Q_2)$, and will get the same j-invariant. However, when taking (E_X, R_2, S_2) as input of the hash function H, any query with

$(E_X, R_2, S_2) \neq (E_A, \phi_A(P_2), \phi_A(Q_2))$ to $\mathcal{H}_B$ will get a totally different hash value.

Remark 2: In definition 8, the 1-Oracle SIDH solver is associated with one-time oracle $\mathcal{H}_B$. We may also associate the solver with one-time oracle $\mathcal{H}_A$ which will return $H((E_Y, R_1, S_1), j(E_Y/\langle R_1 + [a]S_1\rangle))$, given the input a curve E_Y and a basis pair $(R_1, S_1) \in E_Y[l_1^{e_1}]$. Concretely, in definition 8, 1-Oracle SIDH should be called 1-Oracle A-SIDH. If the $\mathcal{H}_B$ is replaced by $\mathcal{H}_A$, it is called 1-Oracle B-SIDH.

1-gap SIDH Assumption. Given $(E_0, P_1, Q_1, P_2, Q_2; E_B, \phi_B(P_1), \phi_B(Q_1))$, let $\mathcal{O}_B((\cdot), \cdot)$ be a highly limited decisional oracle that takes as inputs one curve E_X, one basis pair $R_2, S_2 \in E_X[l_2^{e_2}]$ (that is queried at the first time) and the j-invariant j' are allowed to be queried. It outputs 1, if $j' = j(E_X/\langle R_2 + [b]S_2\rangle)$, and 0 otherwise. We formalize the oracle by adding count which is initiated as 0 as follows.

```
$\mathcal{O}_B((\cdot), \cdot)$ with count=0:
On receiving $E'_X, R'_2, S'_2 \in E'_X[l_2^{e_2}]$ and a $j$-invariant $j'$
if count=0
   $(E_X, R_2, S_2) := (E'_X, R'_2, S'_2)$, count=1
   if $j(E_X/\langle R_2 + [b]S_2\rangle) = j'$
     return 1 else 0.
if count = 1
   if $(E_X, R_2, S_2) \neq (E'_X, R'_2, S'_2)$, abort
   else if $j(E_X/\langle R_2 + [b]S_2\rangle) = j'$
           return 1 else 0.
```

Fig. 2. The limited decisional oracle $\mathcal{O}_B((\cdot), \cdot)$

The intuition of 1-gap SIDH assumption is that the query to the decisional A-DSIDH solver with only one curve and one pair of basis leaks at most $\log poly(\lambda)$ bits of the secret key of ϕ_B, and would not dramatically harm the soundness of CSIDH problem. We formalize this as follows.

Definition 9 (1-gap SIDH Problem). *Given $(E_0, P_1, Q_1, P_2, Q_2; E_A, \phi_A(P_2), \phi_A(Q_2); E_B, \phi_B(P_1), \phi_B(Q_1))$, where E_A, E_B, G_A, G_B are that in CSIDH problem, and a limited oracle $\mathcal{O}_B((\cdot), \cdot)$, find the j-invariant of $E_0/\langle G_A, G_B\rangle$. The advantage of solver $\mathcal{A}$ is*

$$\mathbf{Avd}_{\mathcal{A}}^{\textit{1-gSIDH}} = \Pr[\mathcal{A}^{\mathcal{O}_B}\big(E_A, \phi_A(P_2), \phi_A(Q_2), E_B, \phi_B(P_1), \phi_B(Q_1)\big) = j(E_{AB})],$$

where $E_{AB} \cong E_0/\langle G_A, G_B\rangle$.

We emphasize that if the adversary is allowed to query $\mathcal{O}_B((\cdot), \cdot)$ with unlimited numbers of (E'_X, R'_2, S'_2), 1-gap SIDH problem can be solved using the adaptive attack in [16]. Thus we require a highly limited oracle $\mathcal{O}_B((\cdot), \cdot)$.

Remark 3: Analogous to what we note in Remark 2, 1-gap SIDH can also be specified as 1-gap A-SIDH or 1-gap B-SIDH depending on the limited oracle $\mathcal{O}_B((\cdot), \cdot)$ or $\mathcal{O}_A((\cdot), \cdot)$.

The following theorem shows that the 1-gap SIDH assumption implies the 1-Oracle SIDH assumption when the hash function H is modeled as a random oracle. The proof is inspired by the analysis of the Oracle Diffie-Hellman assumption given by Abdalla, Bellare and Rogaway [1].

Theorem 1. *Let E_0 is the base curve and there are fixed basis pairs $P_1, Q_1 \in E_0[l_1^{e_1}]$, $P_2, Q_2 \in E_0[l_2^{e_2}]$. In the random oracle model, let q be the total number of queries to H-oracle. Then for any algorithm $\mathcal{A}$ against the 1-Oracle SIDH problem there exists an algorithm $\mathcal{B}$ against the 1-gap SIDH problem such that*

$$\mathbf{Avd}_{\mathcal{B}}^{\textit{1-gSIDH}}(\lambda) \geq 1/q \cdot \mathbf{Avd}_{\mathcal{A},H}^{\textit{1-OSIDH}}(\lambda).$$

Proof. Let $\mathcal{A}$ be any algorithm against the 1-Oracle SIDH problem. We can construct an algorithms $\mathcal{B}$ for the 1-gap SIDH problem using $\mathcal{A}$ as a sub-routine in Figure 3. The problem for $\mathcal{B}$ is how to maintain the hash list so as to keep the consistency with the one-time oracle $\mathcal{H}_B$, while the limited oracle $\mathcal{O}_B((\cdot), \cdot)$ would help $\mathcal{B}$ to fix it.

```
Algorithm $\mathcal{B}^{\mathcal{O}_B((\cdot),\cdot)}(A = (E_A, \phi_A(P_2), \phi_A(Q_2)); B = (E_B, \phi_B(P_1), \phi_B(Q_1)))$
  $h_0, h_1 \leftarrow \{0,1\}^\lambda$
  $b \leftarrow \{0,1\}$
  Run $\mathcal{A}^{\mathcal{H}_B(\cdot),H}(A, B, h_b)$
  a. For one-time query $\mathcal{H}_B$
       do as one-time $\mathcal{H}_B$
  b. For the $H$-query
       do as $H((E_X, R_2, S_2), j')$
  c. Let $b'$ be the output of $\mathcal{A}$
   return $j \leftarrow L_H$

One time $\mathcal{H}_B$
  given the one-time query $(E_X, R_2, S_2)$
  if $\exists(E_X, R_2, S_2, j', h') \in L_H \wedge \mathcal{O}_B((E_X, R_2, S_2), j') = 1$
  return $h'$
  else $h \leftarrow \{0,1\}^\lambda$, $L_B = L_B \cup \{E_X, R_2, S_2, j', h\}$
  return $h$

$H((E_X, R_2, S_2), j')$
  if $\exists(E_X, R_2, S_2, j', h') \in L_H$ return $h'$
  else if $\exists((E_X, R_2, S_2), h) \in L_B \wedge \mathcal{O}_B((E_X, R_2, S_2), j') = 1$
        return $h$, $L_H = L_H \cup \{(E_X, R_2, S_2, j', h)\}$
  otherwise $h \leftarrow \{0,1\}^\lambda$
  return $h$, $L_H = L_H \cup \{(E_X, R_2, S_2, j', h)\}$
```

Fig. 3. Algorithm $\mathcal{B}$ for attacking the 1-gap SIDH problem.

Note that in Figure 3, if $\mathcal{H}_B(E_X, R_2, S_2)$ is asked at first and returns a random h, then when (E_X, R_2, S_2, j') is queried to H such that $\mathcal{O}_B((E_X, R_2, S_2), j')$ $= 1$, it will return h. If $H((E_X, R_2, S_2), j')$ is asked at first and returns a random h, then when (E_X, R_2, S_2) is asked to $\mathcal{H}_B$ such that $\mathcal{O}_B((E_X, R_2, S_2), j') = 1$, it will return that h.

Let Ask be the event that $((E_A, \phi_A(P_2), \phi_A(Q_2)), j(E_0/\langle G_A, G_B\rangle))$ is queried to H and $\overline{\mathsf{Ask}}$ be the complement of Ask. If Ask does not happen, which means $((E_A, \phi_A(P_2), \phi_A(Q_2)),\ j(E_0/\langle G_A, G_B\rangle))$ is not queried by $\mathcal{A}$ to H, there is no way to tell whether h_b is equal to $H((E_A, \phi_A(P_2), \phi_A(Q_2)),\ j(E_0/\langle G_A, G_B\rangle))$ or not. Let $(E_A, \phi_A(P_2), \phi_A(Q_2), E_B, \phi_B(P_1),\ \phi_B(Q_1); H(j_{E_{AB}}))$ be a SIDH distribution, and $(E_A, \phi_A(P_2),\ \phi_A(Q_2), E_B, \phi_B(P_1), \phi_B(Q_1); h \leftarrow \{0,1\}^\lambda))$ be a non-SIDH distribution. Thus we have that

$$\begin{aligned}\mathbf{Avd}^{\text{1-OSIDH}}_{\mathcal{A},H} =&\Pr[\mathcal{A}(\mathsf{SIDH}) = 1] - \Pr[\mathcal{A}(\mathsf{non\text{-}SIDH}) = 1]\\ =&\Pr[\mathcal{A}(\mathsf{SIDH}) = 1 \wedge \mathsf{Ask}] - \Pr[\mathcal{A}(\mathsf{non\text{-}SIDH}) = 1 \wedge \mathsf{Ask}]\\ &+ \Pr[\mathcal{A}(\mathsf{SIDH}) = 1 \wedge \overline{\mathsf{Ask}}] - \Pr[\mathcal{A}(\mathsf{non\text{-}SIDH}) = 1 \wedge \overline{\mathsf{Ask}}]\\ =&\Pr[\mathcal{A}(\mathsf{SIDH}) = 1 \wedge \mathsf{Ask}] - \Pr[\mathcal{A}(\mathsf{non\text{-}SIDH}) = 1 \wedge \mathsf{Ask}]\\ \leq& q\Pr[\mathsf{Ask}] \leq q\mathbf{Avd}^{\text{1-gSIDH}}_{\mathcal{B}}.\end{aligned}$$

□

3.3 CPCCA KEM from Supersingular Isogeny

We now propose a strongly secure KEM from supersingular isogeny which is the core building block for our AKEs. At first, we propose an elegant KEM from supersingular isogeny. Then with the purpose of fitting the requirement of CK$^+$ secure AKE, we specify the chosen public key chosen ciphertext security (CPCCA) definition for this KEM. In the CPCCA game, the adversary not only can choose part of the challenge public key but also can query a strong decryption oracle which will decrypt the ciphertext under several public keys rather than only the challenge public key. At last, we prove its security under DSIDH assumption and 1-Oracle SIDH assumption.

Public parameters. Choose $p = l_1^{e_1} l_2^{e_2} \cdot f \pm 1, E_0, \{P_1, Q_1\}, \{P_2, Q_2\}$ as above. Let $h : \{0,1\}^* \to \{0,1\}^{2\lambda}$, $g : \{0,1\}^* \to \{0,1\}^*$ and $H : \{0,1\}^* \to \{0,1\}^\lambda$ be hash functions, where λ is the security parameter. The KEM is shown in Figure 4.

$\mathsf{KeyGen}(\lambda)$. Choose a random element $a_1 \leftarrow \mathbb{Z}/l_1^{e_1}\mathbb{Z}$ and compute the isogeny $\phi_{A_1} : E_0 \to E_{A_1}$ with the kernel $\langle P_1+[a_1]Q_1\rangle$. Let $A_1 = \big(E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2)\big)$. Similarly, choose a random element $x \leftarrow \mathbb{Z}/l_1^{e_1}\mathbb{Z}$ and compute the isogeny $\phi_x : E_0 \to E_X$ with the kernel $\langle P_1 + [x]Q_1\rangle$. Let $X = \big(E_X, \phi_X(P_2), \phi_X(Q_2)\big)$. The encapsulation key is defined as $pk = (A_1, X)$ and the decapsulation key is $sk = (a_1, x)$.

$\mathsf{Encaps}(pk)$. Choose two random elements $m_1, m_2 \leftarrow \{0,1\}^\lambda$ and let $y = g(m_1, m_2)$. First compute the isogeny $\phi_Y : E_0 \to E_Y$ with the kernel $\langle P_2 + [y]Q_2\rangle$ and let $Y = (E_Y, \phi_Y(P_1), \phi_Y(Q_1))$. Then with pk, compute two isogenies $\phi_{YA_1} : E_{A_1} \to E_{YA_1}$ and $\phi_{YX} : E_X \to E_{YX}$ with kernels $\langle\phi_{A_1}(P_2) + [y]\phi_{A_1}(Q_2)\rangle$ and $\langle\phi_X(P_2) + [y]\phi_X(Q_2)\rangle$, respectively. The ciphertext is $C = (Y, y_1, y_2)$, where $y_1 = h(j(E_{YA_1})) \oplus m_1$ and $y_2 = h(X, j(E_{YX})) \oplus m_2$. The session key is $K = H(X, m_1, m_2, C)$.

$\mathsf{Dec}(sk, C)$. With the ciphertext and sk, compute isogenies $\phi_{A_1Y} : E_Y \to E_{A_1Y}$ and $\phi_{XY} : E_Y \to E_{XY}$. Then, compute $m_1' = y_1 \oplus h(j(E_{A_1Y}))$, $m_2' = y_2 \oplus h(X, j(E_{XY}))$ and $y' = g(m_1', m_2')$. Compute the isogeny $\phi_Y' : E_0 \to E_Y'$ and let $E_Y' = E_0/\langle P_2 + [y']Q_2\rangle$. If $Y = (E_Y', \phi_Y'(P_1), \phi_Y'(Q_1))$, then return the session key $H(X, m_1', m_2', C)$, else $\perp$.

Remark 4: In this scheme, those points in the public key are chosen from subgroups of order $l_2^{e_2}$, while those points in the first part of the ciphertext are

```
KeyGen(λ)
$a_1 \leftarrow \mathbb{Z}/l_1^{e_1}\mathbb{Z}$
$E_{A_1} = E_0/\langle P_1 + [a_1]Q_1 \rangle$
$\phi_{A_1}(P_2), \phi_{A_1}(Q_2)$
$A_1 = \big(E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2)\big)$
$x \leftarrow \mathbb{Z}/l_1^{e_1}\mathbb{Z}$
$E_X = E_0/\langle P_1 + [x]Q_1 \rangle$
$\phi_X(P_2), \phi_X(Q_2)$
$X = \big(E_X, \phi_X(P_2), \phi_X(Q_2)\big)$
$pk = (A_1, X), sk = (a_1, x)$
```

```
Encaps(pk)
$m_1, m_2 \leftarrow \{0,1\}^\lambda$
$y = g(m_1, m_2)$
$E_Y = E_0/(P_2 + [y]Q_2)$
$Y = \big(E_Y, \phi_Y(P_1), \phi_Y(Q_1)\big)$
$E_{YA_1} = E_{A_1}/\langle \phi_{A_1}(P_2) + [y]\phi_{A_1}(Q_2) \rangle$
$E_{YX} = E_X/\langle \phi_X(P_2) + [y]\phi_X(Q_2) \rangle$
$y_1 = h(j(E_{YA_1})) \oplus m_1$
$y_2 = h(X, j(E_{YX})) \oplus m_2$
$C = (Y, y_1, y_2), K = H(X, m_1, m_2, C)$
```

```
Dec(sk, C)
$(c_1, c_2, c_3) \leftarrow C$
$E_{A_1Y} = E_Y/\langle \phi_Y(P_1 + [a_1]Q_2) \rangle$
$E_{XY} = E_Y/\langle \phi_Y(P_1 + [x]Q_2) \rangle$
$m_1' = y_1 \oplus h(j(E_{A_1Y}))$
$m_2' = y_2 \oplus h(Y, j(E_{XY}))$
$y' = g(m_1', m_2')$
$E_Y' = E_0/\langle P_2 + [y']Q_2 \rangle$
if $Y = (E_Y', \phi_Y'(P_1), \phi_Y'(Q_1))$
return $H(X, m_1', m_2', C)$
else $\perp$
```

Fig. 4. The CPCCA secure KEM scheme KEM_{dsidh} with a strong decryption oracle.

chosen from subgroups of order $l_1^{e_1}$. We can change the subgroups by replacing A with B and X with Y. The following correctness and security still hold. And now the underlying assumption is altered from the 1-Oracle B-SIDH to the 1-Oracle A-SIDH as in Remark 2.

Correctness. For the purpose of getting the correct key K, we should make sure that $m_1' = m_1$ and $m_2' = m_2$. That is to say, we have to prove the isomorphism between curves E_{A_1Y}, E_{XY} and E_{YA_1}, E_{YX}, respectively. Let $P_1 + [a_1]Q_1 = R_{A_1}$, $P_1 + [x]Q_1 = R_X$, $P_2 + [y]Q_2 = R_Y$ and $P_2 + [b_2]Q_2 = R_{B_2}$ for simplicity.

$$\begin{aligned}
E_{A_1Y} &= E_Y/\langle \phi_Y(P_1) + [a_1]\phi_Y(Q_1) \rangle = (E_0/\langle R_Y \rangle)/(\langle R_{A_1}, R_Y \rangle/\langle R_Y \rangle) \\
&= E_0/\langle R_{A_1}, R_Y \rangle \\
E_{XY} &= E_Y/\langle \phi_Y(P_1) + [x]\phi_Y(Q_1) \rangle = (E_0/\langle R_Y \rangle)/(\langle R_X, R_Y \rangle/\langle R_Y \rangle) \\
&= E_0/\langle R_X, R_Y \rangle \\
E_{YA_1} &= E_{A_1}/\langle \phi_{A_1}(P_2) + [y]\phi_{A_1}(Q_2) \rangle = (E_0/\langle R_{A_1} \rangle)/(\langle R_{A_1}, R_Y \rangle/\langle R_{A_1} \rangle) \\
&= E_0/\langle R_{A_1}, R_Y \rangle \\
E_{YX} &= E_X/\langle \phi_X(P_2) + [y]\phi_X(Q_2) \rangle = (E_0/\langle R_X \rangle)/(\langle R_X, R_Y \rangle/\langle R_X \rangle) \\
&= E_0/\langle R_X, R_Y \rangle
\end{aligned}$$

According to the isomorphism theorem, the curves E_{A_1Y} and E_{YA_1} can be generated by the kernel subgroup $\langle R_{A_1}, R_Y \rangle$, and the curves E_{XY} and E_{YX} are produced by the kernel subgroup $\langle R_X, R_Y \rangle$.

In Figure 5, we give the *chosen pulic key CCA* (CPCCA) game for KEM_{dsidh} with the strong decryption oracle. The CPCCA definition is inspired by Okamoto [32]. Although we use the same name as [32], here the adversary is allowed to choose part of the challenge public key X^* but the adversary in [32] is not allowed. Concretely, $\mathcal{A}$ has the permission to choose X^* and query the strong decryption oracle that will decrypt the ciphertext under general public keys generated by the challenger.

```
Game CPCCA
$a_1 \leftarrow \mathbb{Z}/l_1^{e_1}\mathbb{Z}$
$A_1 = \big(E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2)\big)$
$(state, X^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{i\text{-}dec}}}(A_1)$
$b \leftarrow \{0,1\}, K_1^* \leftarrow \{0,1\}^\lambda$
$(C^*, K_0^*) \leftarrow \mathsf{Encaps}(A_1, X^*)$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{i\text{-}dec}}}(state, C^*, K_b^*)$
return $b' \overset{?}{=} b$
```

```
$\mathcal{O}_{\mathsf{i\text{-}dec}}$
$L = \{-,-\}$
$x_i \leftarrow \mathbb{Z}/l_1^{e_1}\mathbb{Z}$ sends to $\mathcal{A}$
$X_i = \{E_{X_i}, \phi_{X_i}(P_1), \phi_{X_i}(Q_1)\}$
$L = L \cup \{X_i, x_i\}$
On receiving $(X', C')$ from $\mathcal{A}$
If $\exists X_j \in L \wedge X_j = X'$
    $K' = \mathsf{Decaps}((a_1, x_i), C')$
else $\perp$
```

Fig. 5. The CPCCA Security with Strong Decryption Oracle.

The advantage of $\mathcal{A}$ in this game is defined as $\mathbf{Adv}^{\mathrm{CPCCA}}_{\mathcal{A},\mathsf{KEM}_{dsidh}} = \Pr[b = b'] - 1/2$. We say KEM_{dsidh} is CPCCA secure if for any PPT adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{CPCCA}}_{\mathcal{A}}$ is negligible.

Theorem 2. *Under the DSIDH assumption, KEM_{dsidh} is CPCCA secure in the random oracle model. Precisely, for any PPT CPCCA adversary $\mathcal{A}$ with at most q_H queries to H oracle, there exists an algorithm $\mathcal{B}$ solving DSIDH problem such that*

$$\mathbf{Adv}^{CPCCA}_{\mathcal{A},\mathsf{KEM}_{dsidh}} \leq \frac{q_H + 1}{2^\lambda} + 4\mathbf{Avd}^{DSIDH}_{\mathcal{B}}.$$

Proof sketch: By the classical FO transformation [10], the KEM_{dsidh} scheme is obviously classical one-way CCA secure under the DSIDH assumption. But note that here the CPCCA adversary is more powerful than classical CCA adversary in two aspects: 1) it can choose part of the challenge public key X^*; 2) it can query a decryption oracle with the ciphertext under several public keys. To overcome this obstacle, when generating the encapsulated key, we put the public key X in the hashing step.

Since $K = H(X, m_1, m_2, C)$, the adversary has advantages in the random oracle model only if the event that (X', m_1', m_2', C') is queried to H and $(X', m_1', m_2', C') = (X^*, m_1^*, m_2^*, C^*)$ happens (we denote such event as bad). Although X^* is chosen by $\mathcal{A}$ which will help $\mathcal{A}$ to determine m_2^*, the event that it queries H with m_1^* is connected with solving DSIDH problem. Precisely, for public parameters $(E_0, P_1, Q_1, P_2, Q_2)$, given a DSIDH challenge $E_A, \phi_A(P_2), \phi_A(Q_2); E_Y, \phi_Y(P_1), \phi_Y(Q_1)$ and E_C, $\mathcal{S}$ simulates the CPCCA game for $\mathcal{A}$ and transforms the advantage of $\mathcal{A}$ to the advantage of solving the DSIDH problem. $\mathcal{S}$ first sets $A_1 = \big(E_A, \phi_A(P_2), \phi_A(Q_2)\big)$, and on receiving one challenge public key $X^* = (E_X^*, R_2^*, S_2^*)$, $\mathcal{S}$ sets $(Y^*, h(j(E_C)) \oplus m_1^*, y_2^*)$ as the challenge ciphertext, where

$Y^* = \big(E_Y, \phi_Y(P_1),\, \phi_Y(Q_1)\big)$ and $m_1^*, y_2^* \leftarrow \{0,1\}^\lambda$. If bad happens, we can utilize it to solve the DSIDH problem. □

The following theorem states that the CPCCA security for KEM_{dsidh} is still satisfied even $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$ is leaked where $X^* = (E_X^*, R_2^*, S_2^*)$ is part of challenge public key chosen by CPCCA adversary.

Theorem 3. *Under 1-Oracle SIDH assumption,* KEM_{dsidh} *is CPCCA secure in the random oracle model, even if both the challenge ciphertext is given and* $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$ *is leaked. Precisely, for any PPT CPCCA adversary* $\mathcal{A}$ *who also gets* $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$*, with at most* q_H *queries to* H *oracle, there exists an algorithm* $\mathcal{B}$ *such that*

$$\mathbf{Adv}_{\mathcal{A},\mathsf{KEM}_{dsidh}}^{CPCCA} \leq \frac{q_H + 1}{2^\lambda} + 4\mathbf{Avd}_{\mathcal{B}}^{1\text{-}OSIDH}.$$

By the proof sketch of Theorem 2, if the DSIDH challenge is replaced by the 1-oracle SIDH instance, and the challenge ciphertext is $(Y^*, h(j(E_C)) \oplus m_1^*, y_2^*)$, we also query the one-time oracle $\mathcal{H}_Y$ and give $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$ to the adversary $\mathcal{A}$. The other proof proceeds the same with Theorem 2.

4 AKEs from Supersingular Isogeny

Equipped with the CPCCA secure KEM in section 3 as a core building block, we propose a 3-pass and a 2-pass AKE in this section depending on the usage of 1-Oracle SIDH assumption or DSIDH assumption.

4.1 A Three-pass AKE based on 1-Oracle SIDH Assumption

Our AKE utilizes two KEM_{dsidh} that are combined together. With one KEM_{dsidh}, the initiator Alice sets public keys $A_1 = \big(E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2)\big)$ as her static secret key, and $X = \big(E_X, \phi_X(P_2), \phi_X(Q_2)\big)$ as her ephemeral information. With one KEM_{dsidh}, the responder Bob sets public keys $B_2 = \big(E_{B_2}, \phi_{B_2}(P_1), \phi_{B_2}(Q_1)\big)$ as his static secret key, and $Y = \big(E_Y, \phi_Y(P_1), \phi_Y(Q_1)\big)$ as her ephemeral information. Note that X (resp. Y) is not only part of the public keys of Alice (resp. Bob), but also part of the ciphertext under the public key (B_2, Y) (resp. (A_1, X)) of Bob (resp. Alice). More details are given in Figure 6.

Public parameters. Choose $p = l_1^{e_1} l_2^{e_2} \cdot f \pm 1, E_0, \{P_1, Q_1\}, \{P_2, Q_2\}$ as above. Let hash functions be $G : \{0,1\}^* \to \{0,1\}^{4\lambda}$, $g : \{0,1\}^* \to \{0,1\}^*$, $h : \{0,1\}^* \to \{0,1\}^{2\lambda}$, $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$ and $\hat{H} : \{0,1\}^* \to \{0,1\}^\lambda$, where λ is the security parameter.

Static secret and public keys. To solve the three-body problem [14] that one party can play both the roles of initiator and responder in a multi-user setting, we should distribute two pairs of static keys for the involved parties. The static secret-public key of U_A as initiator is $(a_1; E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2))$. The static secret-public key of U_A as responder is $(a_2; E_{A_2}, \phi_{A_2}(P_2), \phi_{A_2}(Q_2))$. The static secret-public key of U_B as responder is $(b_2; E_{B_2}, \phi_{B_2}(P_1), \phi_{B_2}(Q_1))$. The static secret-public key of U_B as initiator is $(b_1; E_{B_1}, \phi_{B_1}(P_2), \phi_{B_1}(Q_2))$. We distinguish the party as initiator or responder by the subscript. The subscript represents which subgroup their secret keys lie in.

Step 1. U_A selects a random element $r_1 \in \{0,1\}^{\lambda}$, parses $G(r_1, a_1)$ into two equal bitstrings $n_1||n_2$, and then hashes the concatenation $g(n_1, n_2)$ to an element x in $\mathbb{Z}/l_1^{e_1}\mathbb{Z}$. Then U_A invokes the isogeny computation algorithm to compute the isogeny ϕ_X with the kernel $\langle P_1 + [x]Q_1 \rangle$ and publishes $X = \big(E_X, \phi_X(P_2), \phi_X(Q_2)\big)$. U_A further computes ϕ_{XB_2} with the kernel $\langle \phi_{B_2}(P_1) + [x]\phi_{B_2}(Q_1) \rangle$. Half of the secret key n_1 is hidden in the message $x_1 = h(j(E_{XB_2})) \oplus n_1$.Then U_A sends to U_B the identities of U_A and U_B as well as (X, x_1).

Step 2. U_B chooses a random element r_2 in $\{0,1\}^{\lambda}$, parses $G(r_2, b_2)$ to two bitstrings $m_1||m_2$ of equal length, and then hashes (m_1, m_2) to get a secret key y for these isogenies. U_B computes three $l_2^{e_2}$-degree isogenies ϕ_Y, ϕ_{YA_1} and ϕ_{YX} with kernels $\langle P_2 + [y]Q_2 \rangle$, $\langle \phi_{A_1}(P_2) + [y]\phi_{A_1}(Q_2) \rangle$ and $\langle \phi_X(P_2) + [y]\phi_X(Q_2) \rangle$, respectively. And U_B defines the secret key $K_B = H(X, m_1, m_2, Y, y_1, y_2)$, where $y_1 = h(j(E_{YA_1})) \oplus m_1$ and $y_2 = h(X, j(E_{YX})) \oplus m_2$. Then U_B forwards the identity messages of U_A and U_B along with (Y, y_1, y_2) to U_A.

Step 3. Upon receiving the message from U_B, U_A computes isogenies ϕ_{A_1Y} and ϕ_{XY} to get the hash values of j-invariants $\big(h(j(E_{A_1Y})),\ h(X, j(E_{XY}))\big)$ with the secret keys a_1 and x. Then U_A could extract m_1' and m_2' by computing $h(j(E_{A_1Y})) \oplus y_1$ and $h(X, j(E_{XY})) \oplus y_2$ respectively. Hence, U_A can retrieve what U_B has done and verify the validation of the ciphertext Y. If the validation passes, then U_A obtains the key $K_B' = H(X, m_1', m_2', Y, y_1, y_2)$ and transmits messages $x_2 = h(Y, j(E_{XY})) \oplus n_2$ to U_B.

The key K_A is $H(Y, n_1, n_2, X, x_1, x_2)$. U_A sets the session identity $sid = \big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, x_2, Y, y_1, y_2\big)$ and completes the session with the session key $SK = \hat{H}(sid, K_A, K_B')$.

Step 4. U_B computes ϕ_{B_2X} with the static secret key b_2 and the public key X from U_A during the first round. Then U_B can obtain both n_1' and n_2' by $h(j(E_{B_2X})) \oplus x_1$ and $h(Y, j(E_{YX})) \oplus x_2$. Hence, U_B likewise recomputes what U_A has computed and verifies the validation of the ciphertext X. If the validation passes, U_B gets the key $K_A' = H(Y, n_1', n_2', X, x_1, x_2)$, sets the session identity $sid = \big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, x_2, Y, y_1, y_2\big)$ and completes the session with the session key $SK = \hat{H}(sid, K_A', K_B)$.

The session state of sid owned by U_A consists of the ephemeral secret key r_1, the decapsulated key K_B' and the encapsulated key K_A. The session state of sid owned by U_B consists of the ephemeral secrete key r_2, the encapsulated key K_B and the decapsulated key K_A'.

Correctness. To show that both U_A and U_B can agree on the same session key, we present a thorough and detailed analysis. The different parts of keys K_A , K_B' of U_A and K_A', K_B of U_B are the values n_1, n_2, m_1', m_2' and n_1', n_2', m_1, m_2. We have to show that $n_1' = n_1, n_2' = n_2$ and $m_1' = m_1, m_2' = m_2$. That is to say we are required to prove the isomorphism between curves E_{XB_2}, E_{XY}, E_{A_1Y} and E_{B_2X}, E_{YX}, E_{YA_1}, respectively. We have proved the isomorphism of E_{XY}, E_{A_1Y} and E_{YX}, E_{YA_1} in Section 4.1.

U_A		U_B
$sk_{A_1} : a_1 \in \mathbb{Z}/l_1^{e_1}\mathbb{Z}$		$sk_{B_2} : b_2 \in \mathbb{Z}/l_2^{e_2}\mathbb{Z}$
$pk_{A_1} : E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2)$		$pk_{B_2} : E_{B_2}, \phi_{B_2}(P_1), \phi_{B_2}(Q_1)$
$sk_{A_2} : a_2 \in \mathbb{Z}/l_2^{e_2}\mathbb{Z}$		$sk_{B_1} : b_1 \in \mathbb{Z}/l_1^{e_1}\mathbb{Z}$
$pk_{A_2} : E_{A_2}, \phi_{A_2}(P_1), \phi_{A_2}(Q_1)$		$pk_{B_1} : E_{B_1}, \phi_{B_1}(P_2), \phi_{B_1}(Q_2)$
$r_1 \leftarrow \{0,1\}^\lambda$. $n_1 \| n_2 \leftarrow G(r_1, a_1)$		
$x \leftarrow g(n_1, n_2)$		$r_2 \leftarrow \{0,1\}^\lambda$. $m_1 \| m_2 \leftarrow G(r_2, b_2)$
$E_X = E_0/\langle P_1 + [x]Q_1\rangle$		$y \leftarrow g(m_1, m_2)$
$X = (E_X, \phi_X(P_2), \phi_X(Q_2))$		$E_Y = E_0/\langle P_2 + [y]Q_2\rangle$
$E_{XB_2} = E_{B_2}/\langle \phi_{B_2}(P_1) + [x]\phi_{B_2}(Q_1)\rangle$		$Y = (E_Y, \phi_Y(P_1), \phi_Y(Q_1))$
$x_1 = h(j(E_{XB_2})) \oplus n_1$	$\xrightarrow{X, x_1}$	$E_{YA_1} = E_{A_1}/\langle \phi_{A_1}(P_2) + [y]\phi_{A_1}(Q_2)\rangle$
		$E_{YX} = E_X/\langle \phi_X(P_2) + [y]\phi_X(Q_2)\rangle$
		$y_1 = h(j(E_{YA_1})) \oplus m_1$
		$y_2 = h(X, j(E_{YX})) \oplus m_2$
	$\xleftarrow{Y, y_1, y_2}$	$K_B = H(X, m_1, m_2, Y, y_1, y_2)$
$E_{A_1Y} = E_Y/\langle \phi_Y(P_1) + [a_1]\phi_Y(Q_1)\rangle$		
$E_{XY} = E_Y/\langle \phi_Y(P_1) + [x]\phi_Y(Q_1)\rangle$		
$m_1' = h(j(E_{A_1Y})) \oplus y_1$		
$m_2' = h(X, j(E_{XY})) \oplus y_2$		
$y' = g(m_1', m_2')$		
$E_Y' = E_0/\langle P_2 + [y']Q_2\rangle$		
If $Y \neq (E_Y', \phi_Y'(P_1), \phi_Y'(Q_1)), \perp$		
$K_B' = H(X, m_1', m_2', Y, y_1, y_2)$		
$x_2 = h(Y, j(E_{XY})) \oplus n_2$	$\xrightarrow{x_2}$	
		$E_{B_2X} = E_X/\langle \phi_X(P_2) + [b_2]\phi_X(Q_2)\rangle$.
		$n_1' = h(j(E_{B_2X})) \oplus x_1$
		$n_2' = h(Y, j(E_{YX})) \oplus x_2$
		$x' = g(n_1', n_2')$
		$E_X' = E_0/\langle P_1 + [x']Q_1\rangle$
		If $X \neq (E_X', \phi_X'(P_2), \phi_X'(Q_2)), \perp$
$K_A = H(Y, n_1, n_2, X, x_1, x_2)$		$K_A' = H(Y, n_1', n_2', X, x_1, x_2)$
$SK = \hat{H}(sid, K_A, K_B')$		$SK = \hat{H}(sid, K_A', K_B)$

Fig. 6. A 3-Pass AKE $\mathsf{AKE}_{\mathsf{SIDH\text{-}3}}$ Based on 1-Oracle SIDH. Here sid is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, Y, y_1, y_2, x_2\big)$.

$$\begin{aligned}E_{XB_2} &= E_{B_2}/\langle\phi_{B_2}(P_1)+[a'_1]\phi_{B_2}(Q_1)\rangle = (E_0/\langle R_{B_2}\rangle)/(\langle R_X, R_{B_2}\rangle/\langle R_{B_2}\rangle)\\ &= E_0/\langle R_X, R_{B_2}\rangle\\ E_{B_2X} &= E_X/\langle\phi_X(P_2)+[b_2]\phi_X(Q_2)\rangle = (E_0/\langle R_X\rangle)/(\langle R_X, R_{B_2}\rangle/\langle R_X\rangle)\\ &= E_0/\langle R_X, R_{B_2}\rangle\end{aligned}$$

It is easy to see that the two curves are isomorphic. Hence, they definitely arrive at the same key.

Theorem 4. *Under the 1-Oracle SIDH assumption, the 3-pass AKE* $\mathsf{AKE}_{\mathit{SIDH\text{-}3}}$ *supports arbitrary registrant and 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* $\mathsf{AKE}_{\mathit{SIDH\text{-}3}}$ *with* q_{hash} *times of hash oracle queries and* q *times of CK*$^+$ *queries, there exists* $\mathcal{S}$ *s.t.*

$$\mathbf{Adv}^{\mathrm{CK}^+}_{\mathsf{AKE}_{\mathit{SIDH\text{-}3}}}(\mathcal{A}) \leq 1/2 + N^2lq(\frac{q_{hash}+1}{2^\lambda} + 4\mathbf{Avd}^{\mathit{1\text{-}OSIDH}}_{\mathcal{S}}).$$

Proof. Let Succ be the event that the guess of $\mathcal{A}$ against the test session is correct. Let AskH be the event that $\mathcal{A}$ poses $(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, Y, y_1, y_2, x_2, K_A, K_B)$ to $\hat{H}$, where $(X, x_1, Y, y_1, y_2, x_2)$ is the view of the test session and K_A, K_B is the key encapsulated in the test session. Let $\overline{\mathsf{AskH}}$ be the complement of AskH. Then,

$$\Pr[\mathsf{Succ}] = \Pr[\mathsf{Succ}\wedge\overline{\mathsf{AskH}}] + \Pr[\mathsf{Succ}\wedge\mathsf{AskH}] \leq \Pr[\mathsf{Succ}\wedge\overline{\mathsf{AskH}}] + \Pr[\mathsf{AskH}],$$

where the probability is taken over the randomness used in CK$^+$ experiment.

Lemma 1. *If* H *is modeled as a random oracle, we have* $Pr[\mathit{Succ}\wedge\overline{\mathit{AskH}}] \leq 1/2$.

Proof of Lemma 1: If $\Pr[\overline{\mathsf{AskH}}] = 0$ then the claim is straightforward, otherwise we have $\Pr[\mathsf{Succ}\wedge\overline{\mathsf{AskH}}] = \Pr[\mathsf{Succ}|\overline{\mathsf{AskH}}]\Pr[\overline{\mathsf{AskH}}] \leq \Pr[\mathsf{Succ}|\overline{\mathsf{AskH}}]$. Let sid be any completed session owned by an honest party such that sid $\neq$ sid* and sid is not the matching session of sid*. The inputs to sid are different from those of sid* and $\overline{sid}^*$ (if there exists the matching session of sid*). If $\mathcal{A}$ does not explicitly query the view and keys to the oracle, then $\hat{H}(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, Y, y_1, y_2, x_2, K_A, K_B)$ is completely random from $\mathcal{A}$'s point of view. Therefore, the probability that $\mathcal{A}$ wins when AskH does not occur is exactly $1/2$.

We then show that $\Pr[\mathsf{AskH}]$ is negligible (as in Lemma 2) in all the events (listed in Table 2) of the CK$^+$ model. Followed by Lemma 2, we achieve the security of AKE in the CK$^+$ model. Thus, we only need to prove Lemma 2 in the following.

Lemma 2. *If the 1-Oracle SIDH assumption holds, the probability of the occurrence of event* AskH *defined above is negligible. Precisely,*

$$\Pr[\mathit{AskH}] \leq N^2lq(\frac{q_{hash}+1}{2^\lambda} + 4\mathbf{Avd}^{\mathit{1\text{-}OSIDH}}_{\mathcal{S}}).$$

We give the proof sketch here, and for formal proof please refer to Appendix A. As in Theorem 3, under 1-Oracle SIDH assumption, KEM_{dsidh} is CPCCA secure even $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$ is leaked, we first bound the advantage of AskH using the CPCCA security of KEM_{dsidh} and further bound it with $\mathbf{Avd}^{\text{1-OSIDH}}$.

Let AskG be the event that the static secret key of one user, for example sk_{B_2}, is queried by the CK$^+$ adversary $\mathcal{A}$ to G. In order to bound the probability of AskH, we handle the event $\mathsf{AskH} \wedge \mathsf{AskG}$, as well as the events $\mathsf{AskH} \wedge \overline{\mathsf{AskG}} \wedge E_i$ for $1 \leq i \leq 8$ one by one, where E_i is listed in Table 2.

If AskG happens (meaning the secret key b_2 of E_{B_2} is known), then given the 1-Oracle SIDH (actually the DSIDH assumption is enough) challenge $(E_A, \phi_A(P_2), \phi_A(Q_2), E_B, \phi_B(P_1), \phi_B(Q_1); h)$, $\mathcal{S}$ sets the public key of U_B as responder to be $pk_{B_2} = \big(E_B, \phi_B(P_1), \phi_B(Q_1)\big)$. If b_2 is queried by $\mathcal{A}$, $\mathcal{S}$ could compute E_{AB} by himself and solve the 1-Oracle SIDH problem. Thus $\Pr[\mathsf{AskG}] \leq \mathbf{Adv}_{\mathcal{S}}^{\text{1-OSIDH}}$.

In the following, we assume that $\overline{\mathsf{AskG}}$ (the complement of AskG) happens with the events $\mathsf{AskH} \wedge \overline{\mathsf{AskG}} \wedge E_i$, for $1 \leq i \leq 8$. Here, we only take $\mathsf{AskH} \wedge \overline{\mathsf{AskG}} \wedge E_3$ as an example to explain in detail. For the other cases we can deal with them in the same way. In the case of $\mathsf{AskH} \wedge \overline{\mathsf{AskG}} \wedge E_3$, the 1-Oracle SIDH adversary $\mathcal{S}$ performs as follows. It simulates the CK$^+$ games, and transfers the probability that the event AskH performed by $\mathcal{A}$ to the advantage of solving the 1-Oracle SIDH problem. Based on Theorem 3, we transform the event AskH performed by $\mathcal{A}$ to the advantage of against CPCCA security of KEM_{dsidh} with the public key as (pk_{A_1}, X) and the ciphertext as (Y, y_1, y_2).

In order to simulate the random oracles, $\mathcal{S}$ maintains three lists for $\hat{H}$ and G oracle and $\mathsf{SessionKeyReveal}$, respectively. The $\hat{H}$-oracle and $\mathsf{SessionKeyReveal}$ are related, which means the adversary may ask $\mathsf{SessionKeyReveal}$ without the encapsulated keys at first, and then may ask $\hat{H}$-oracle with the encapsulated keys. Thus, the reduction must ensure consistency with the random oracle queries to $\hat{H}$ and $\mathsf{SessionKeyReveal}$. The strong decapsulation oracle for KEM_{dsidh} would help to maintain the consistency of $\hat{H}$-oracle and $\mathsf{SessionKeyReveal}$.

On receiving the public key $\big(E_A, \phi_A(P_2), \phi_A(Q_2)\big)$ from the CPCCA challenger in Theorem 3, in order to simulate the CK$^+$ game, $\mathcal{S}$ randomly chooses two parties U_A, U_B and the i-th session as a guess of the test session with success probability $1/N^2 l$. $\mathcal{S}$ 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 U_A for whom $\mathcal{S}$ only computes and sets the static public key as responder. Specially, $\mathcal{S}$ sets the static secret and public key pairs (pk_{B_2}, sk_{B_2}) for U_B as responder, and sets $pk_{A_1} = (E_A, \phi_A(P_2), \phi_A(Q_2))$ for U_A as initiator.

Without knowing the secret key of U_A as initiator, $\mathcal{S}$ chooses totally random r_1 as part of the ephemeral secret key and totally random x. Since G is a hash function and sk_{A_1} is not queried, the difference between simulation with modification of r_1 and a real game can not be detected by the adversary. When a session state of a session owned by U_A is queried, $\mathcal{S}$ returns r_1 of this session as part of the ephemeral secret key.

On receiving $(X^* = (E_X^*, R_2^*, S_2^*), x_1)$ of the i-th session from U_A (that is sent by $\mathcal{A}$ in the CK$^+$ game), $\mathcal{S}$ returns X^* to the CPCCA challenger and receives the challenge ciphertext (Y^*, y_1^*, y_2^*) (under public key pk_{A_1} and X^*) and $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$ as extra leakage. Then $\mathcal{S}$ returns (Y^*, y_1^*, y_2^*) to U_A as the response of i-th session from U_B. $\mathcal{S}$ chooses a totally independent randomness r_2 as the ephemeral secret key of U_B for C^* and leaks it to the adversary $\mathcal{A}$. Since G is a hash function, the difference between simulation with modification of r_2 and the real game can not be detected by the adversary.

$\mathcal{S}$ simulates the oracle queries of $\mathcal{A}$ and maintains the hash lists. Specially, when AskH happens, which means $\mathcal{A}$ poses $(U_A, U_B, pk_{A_1}, pk_{B_2}, X^*, x_1, Y^*, y_1^*, y_2^*, x_2, K_A, K_B)$ to $\hat{H}$, where $(X^*, x_1, Y^*, y_1^*, y_2^*, x_2)$ is the view of the test session and K_B is the key encapsulated in (X^*, x_1, x_2) (this can be detected by $\mathcal{S}$ since it has sk_{B_2} and $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$ from CPCCA challenger), $\mathcal{S}$ returns K_A as the guess of K^* encapsulated in (Y^*, y_1^*, y_2^*), which contradicts with the CPCCA security of KEM_{dsidh} and further the 1-Oracle SIDH assumption. □

4.2 A Two-Pass AKE based on SIDH Assumption

Although the 3-pass AKE has the advantage of less communication, it relies on a non-standard assumption, 1-Oracle SIDH assumption. To enhance the security reliability of AKE, we propose a 2-pass AKE with a little more communication (which is still competitive with these existing schemes) based on the DSIDH assumption. Since we add an ephemeral public key X_0, the hash function h for x_2, y_2 does not need X and Y as extra input.

Intuition of 2-Pass AKE. The reason why we need the non-standard 1-Oracle SIDH assumption is that, 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_2) is computed; on the other hand X is part of the ciphertext (X, x_1, x_2) which encapsulates K_A. An extra X_0 instead of X as part of the public key (pk_{A_1}, X_0) could effectively help to get rid of the 1-Oracle SIDH assumption. Besides, it is also necessary to delete x_2 and set (X, x_1) as the ciphertext under the public key pk_{B_2} rather than (pk_{B_2}, Y). Then the $h(j(E_X/\langle R_2^* + [y]S_2^*\rangle))$ is not needed any longer.

Two-Pass AKE. The public parameters and static secret-public keys are defined the same as those in our 3-pass AKE in Section 4.1.

Step 1. U_A randomly selects two elements $r_1 \in \{0,1\}^\lambda$ and $r_{X_0} \in \mathbb{Z}/l_1^{e_1}\mathbb{Z}$. Let $n_1 = G(r_1, a_1)$ and $x = g(n_1)$. Then U_A computes an $l_1^{e_1}$-degree isogeny ϕ_X with the kernel equal to $\langle P_1 + [x]Q_1\rangle$ and an $l_1^{e_1}$-degree isogeny ϕ_{XB_2} with kernel $\langle \phi_{B_2}(P_1) + [x]\phi_{B_2}(Q_1)\rangle$. Then U_A generates an ephemeral public key $X_0 = \big(E_{X_0}, \phi_{X_0}(P_2), \phi_{X_0}(Q_2)\big)$ with the secret key r_{X_0}. U_A sends (X, x_1, X_0) to U_B, where $(X, x_1, X_0) = \big(E_X, \phi_X(P_2), \phi_X(Q_2); h(j(E_{XB_2}))\oplus n_1; E_{X_0}, \phi_{X_0}(P_2), \phi_{X_0}(Q_2)\big)$.

Step 2. On receiving the message from U_A, U_B chooses a random element r_2 in $\{0,1\}^\lambda$, parses $G(r_2, b_2)$ as two bit strings $m_1||m_2$ of the same length and then computes the hash of the concatenation of (m_1, m_2) to get a secret key y. U_B computes three $l_2^{e_2}$-degree isogenies with kernels $\langle P_2 + [y]Q_2\rangle$, $\langle \phi_{A_1}(P_2) + [y]\phi_{A_1}(Q_2)\rangle$ and $\langle \phi_{X_0}(P_2) + [y]\phi_{X_0}(Q_2)\rangle$, respectively. U_B sets the message $Y = \big(E_Y, \phi_Y(P_1), \phi_Y(Q_1)\big)$ and the secret key $K_B = H(X_0, m_1, m_2, Y, y_1, y_2)$, where $y_1 = h(j(E_{YA_1}))\oplus m_1$ and $y_2 = h(j(E_{YX_0}))\oplus m_2$. Then U_B forwards the identity messages of U_A and U_B with (Y, y_1, y_2) to U_A.

U_B computes the isogeny ϕ_{B_2X} with the static secret key b_2 and gets n_1' by $h(j(E_{B_2X})) \oplus x$ to compute $x' = g(n_1')$. Then U_B recomputes $E_X' = E_0/\langle P_1 + [x']Q_1\rangle$. If $(E_X', \phi_X'(P_2), \phi_X'(Q_2))$ is not equal to X, output $\perp$. Otherwise, set $K_A' = H(n_1')$. The session identity is $sid = \big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x, X_0, Y, y_1, y_2\big)$. Finally, U_B completes the session with the session key $SK = \hat{H}(sid, K_A', K_B)$.

Step 3. Upon receiving the message from U_B, U_A computes isogenies ϕ_{A_1Y} and $\phi_{X'Y}$ with the secret keys a_1 and $r_{X'}$. U_A obtains m_1' and m_2' by $h(j(E_{A_1Y}))\oplus y_1$ and $h(j(E_{X'Y})) \oplus y_2$. Then U_A recomputes $E_Y' = E_0/\langle P_2 + [y']Q_2\rangle$. If $Y \neq (E_Y', \phi_Y'(P_1), \phi_Y'(Q_1))$, output $\perp$. Otherwise, set $K_B' = H(X_0, m_1', m_2', Y, y_1, y_2)$. Then U_A sets the session identity $sid = \big(U_A, U_B, ek_{A_1}, ek_{B_2}, X, x, X', Y, y_1, y_2\big)$. Similarly, U_A completes the session with the session key $SK = \hat{H}(sid, K_A, K_B')$.

The session state of sid owned by U_A consists of the ephemeral secret key r_1, r_{X_0}, the decapsulated key K_B' and the encapsulated key K_A. The session state of sid owned by U_B consists of the ephemeral secrete key r_2 and the encapsulated key K_B, but does not include the decapsulated key K_A'.

Correctness. This property can refer to the proof of 3-pass AKE for reference, but there is one different point that the two pairs of curves are not exactly the same. We only have to prove the isomorphism of $E_{X'Y}$ and $E_{YX'}$. We define $P_1 + [r_{X'}]Q_1 = R_{X'}$ for simplicity.

$$\begin{aligned}
E_{X'Y} &= E_Y/\langle \phi_Y(P_1) + [r_{X'}]\phi_Y(Q_1)\rangle = (E_0/\langle R_Y\rangle)/(\langle R_{X'}, R_Y\rangle/\langle R_Y\rangle) \\
&= E_0/\langle R_{X'}, R_Y\rangle \\
E_{YX'} &= E_{X'}/\langle \phi_{X'}(P_2) + [b_2']\phi_{X'}(Q_2)\rangle = (E_0/\langle R_{X'}\rangle)/(\langle R_{X'}, R_Y\rangle/\langle R_{X'}\rangle) \\
&= E_0/\langle R_{X'}, R_Y\rangle
\end{aligned}$$

It is easy to see that the two curves $E_{X'Y}$ and $E_{YX'}$ are isomorphic. So they own the same j-invariant and then share the same session key.

Theorem 5. *Under the DSIDH assumption, the 2-pass AKE* $\mathsf{AKE}_{\mathit{SIDH\text{-}2}}$ *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* $\mathsf{AKE}_{\mathit{SIDH\text{-}2}}$ *with q_{hash} times of hash oracle queries and q times of CK*$^+$ *queries, there exists $\mathcal{S}$ s.t.*

$$\mathbf{Adv}^{\mathrm{CK}^+}_{\mathsf{AKE}_{\mathit{SIDH\text{-}2}}}(\mathcal{A}) \leq 1/2 + N^2lq(\frac{q_{hash}+1}{2^\lambda} + 4\mathbf{Avd}^{\mathit{1\text{-}DSIDH}}_{\mathcal{S}}).$$

Proof sketch: The proof proceeds similarly to that of 3-pass AKE and the main difference is the proof of Lemma 2, but much easier. In the 2-pass AKE, we add an extra X_0 to take the position as part of the public key (pk_{A_1}, X_0) under which the ciphertext (Y, y_1, y_2) is computed, delete x_2 and set (X, x_1) to be the ciphertext under public key pk_{B_2} rather than (pk_{B_2}, Y). Now in order to compute K_A encapsulated in (X, x_1), $h(j(E_X/\langle R_2^* + [y]S_2^*\rangle))$ is not required any longer and sk_{B_1} is enough. In other cases, for example E_1, (X^*, x_1^*) is the challenge ciphertext in the test session. Since X_0 is generated by $\mathcal{S}$, on receiving (Y, y_1, y_2) it can query the CPCCA decapsulation oracle with $(X_0; Y, y_1, y_2)$ to extract K_B'.

We omit the details here. And based on Theorem 2, the security relies on a standard DSIDH assumption.

U_A		U_B
$sk_{A_1} : a_1 \in \mathbb{Z}/l_1^{e_1}\mathbb{Z}$		$sk_{B_2} : b_2 \in \mathbb{Z}/l_2^{e_2}\mathbb{Z}$
$pk_{A_1} : E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2)$		$pk_{B_2} : E_{B_2}, \phi_{B_2}(P_1), \phi_{B_2}(Q_1)$
$sk_{A_2} : a_2 \in \mathbb{Z}/l_2^{e_2}\mathbb{Z}$		$sk_{B_1} : b_1 \in \mathbb{Z}/l_1^{e_1}\mathbb{Z}$
$pk_{A_2} : E_{A_2}, \phi_{A_2}(P_1), \phi_{A_2}(Q_1).$		$pk_{B_1} : E_{B_1}, \phi_{B_1}(P_2), \phi_{B_1}(Q_2).$
$r_1 \leftarrow \{0,1\}^{\lambda}$, $r_{X_0} \leftarrow \mathbb{Z}/l_1^{e_1}\mathbb{Z}.$		
$n_1 \leftarrow G(r_1, a_1), x = g(n_1)$		$r_2 \leftarrow \{0,1\}^{\lambda}$
$E_X = E_0/\langle P_1 + [x]Q_1\rangle$		$m_1 \Vert m_2 \leftarrow G(r_2, b_2)$
$E_{XB_2} = E_{B_2}/\langle \phi_{B_2}(P_1) + [x]\phi_{B_2}(Q_1)\rangle$		$y \leftarrow g(m_1, m_2)$
$E_{X_0} = E_0/\langle P_1 + [r_{X_0}]Q_1\rangle$		$E_Y = E_0/\langle P_2 + [y]Q_2\rangle$
$X = (E_X, \phi_X(P_2), \phi_X(Q_2))$		$Y = (E_Y, \phi_Y(P_1), \phi_Y(Q_1))$
$X_0 = \big(E_{X_0}, \phi_{X_0}(P_2), \phi_{X_0}(Q_2)\big)$		$E_{YA_1} = E_{A_1}/\langle \phi_{A_1}(P_2) + [y]\phi_{A_1}(Q_2)\rangle$
$x_1 = h(j(E_{XB_2})) \oplus n_1$		$E_{YX_0} = E_{X_0}/\langle \phi_{X_0}(P_2) + [y]\phi_{X_0}(Q_2)\rangle$
$K_A = H(n_1)$		$y_1 = h(j(E_{YA_1})) \oplus m_1$
	$\xrightarrow{X, x_1; X_0}$	$y_2 = h(j(E_{YX_0})) \oplus m_2$
		$K_B = H(X_0, m_1, m_2, Y, y_1, y_2)$
	$\xleftarrow{Y, y_1, y_2}$	
$E_{A_1Y} = E_Y/\langle \phi_Y(P_1) + [a_1]\phi_Y(Q_1)\rangle$		$E_{B_2X} = E_X/\langle \phi_X(P_2) + [b_2]\phi_X(Q_2)\rangle.$
$E_{X_0Y} = E_Y/\langle \phi_Y(P_1) + [r_{X_0}]\phi_Y(Q_1)\rangle.$		
$m_1' = h(j(E_{A_1Y})) \oplus y_1$		
$m_2' = h(j(E_{X_0Y})) \oplus y_2$		$n_1' = h(j(E_{B_2X})) \oplus x_1$
$y' = g(m_1', m_2')$		$x' = g(n_1')$
$E_Y' = E_0/\langle P_2 + [y']Q_2\rangle$		$E_X' = E_0/\langle P_1 + [x']Q_1\rangle$
If $Y \neq (E_Y', \phi_Y'(P_1), \phi_Y'(Q_1)), \perp$		If $X \neq (E_X', \phi_X'(P_2), \phi_X'(Q_2)), \perp$
$K_B' = H(X_0, m_1', m_2', Y, y_1, y_2)$		$K_A' = H(n_1')$
$SK = \hat{H}(sid, K_A, K_B')$		$SK = \hat{H}(sid, K_A', K_B)$

Fig. 7. A Compact 2-pass AKE Based on SIDH. Here *sid* is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x, X_0, Y, y_1, y_2\big)$.

5 Parameters, Implementation and Comparison

If we demand λ bits of quantum security and adopt the parameters chosen in [6] which are considered to be the most efficient choices, then the prime is of bit-length 6λ. Each field element needs 12λ bits since the curve is defined over $\mathbb{F}_{p^2}$. Then considering the Kummer arithmetic, the A-coefficient and a point both require 12λ bits. In the FSXY scheme where both U_A and U_B would like to share a session key, they need to transmit 148λ bits. Fortunately, from Figure 6 we can see that this 3-pass scheme narrows the bandwidth to 80λ bits and reduces the computation cost to 5 isogenies for per party but at the expense of one more round of interaction with parties. Furthermore, in the 2-pass AKE we narrow the bandwidth to 114λ bits, reducing the size of the uncompressed public keys by approximately 23%. If considering the public key compression [4], we can compress the total bandwidth to 69λ bits. As the compression will bring high computation cost, we will not consider this way here.

To evaluate the performance of our proposed two authenticated key exchange protocols, we write a supporting program based on the optimized implementation of SIKE [19]. It is written in portable C only and makes use of efficient algorithms for fast isogeny computation and field arithmetic implementation. We adopt the curve SIKE751 for 128-bit quantum security. The SIKE751 fixes the prime $p = 2^{372}3^{239} - 1$ and $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$ for $i^2 = -1$. The supersingular elliptic curve is the Montgomery curve $E_0 : y^2 = x^3 + x$. The generator points are selected as $P_1 = [3^{239}](11, \sqrt{11^3 + 11}), Q_1 = \tau(P_1)$; $P_2 = [2^{372}](6, \sqrt{6^3 + 6}), Q_2 = \tau(P_2)$, where τ is an endomorphism mapping (x, y) to $(-x, iy)$. This prime p of bitlength 751 provides quantum 124-bit security and classically 186-bit security.

The performance is benchmarked on an Intel(R) Core i7-6567U CPU @3.30GHz processor supporting the Skylake micro-architecture. We perform the test experiments of the four schemes on the same platform, in order to compare their performance more intuitively and credibly. The reason we choose thses four schemes is that they allow arbitary registrant and achieve high security. Although FTTY 2 can be proved to be CK$^+$ secure, it only allows honest registrant.

In terms of implementation, the hash functions used in the authenticated key exchange are all instantiated with the SHA-3 function cSHAKE256 [23]. The size of one SIDH protocol public key are 564 bytes and the size of the additional hash value transmitted together with public keys are 32 bytes. Message sizes are shown in Table 3. It is easy to see that our 3-pass AKE protocol reduces the bandwidth almost to the half of both FSXY [12] and BCNP-Lon [2, 28].

In Table 4, we present the performance of our protocols comparing with the FSXY scheme [12] and the BCNP-Lon scheme [2, 28]. 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-Lon.

Scheme	$A \rightarrow B$	$B \rightarrow A$	$A \rightarrow B$	**total(byte)**
FSXY [12]	1160	1160	-	2320
BCNP-Lon [2, 28]	1160	1160	-	2320
$\mathsf{AKE}_{\mathsf{SIDH\text{-}2}}$	1160	628	-	1788
$\mathsf{AKE}_{\mathsf{SIDH\text{-}3}}$	596	628	32	1176

Table 3. Comparison of message sizes. We adopt the parameters chosen in [19], taking into account the efficiency. "-" stands for no messages to be transmitted. Only $\mathsf{AKE}_{\mathsf{SIDH\text{-}3}}$ is a 3-pass one and then has a message from A to B again. The message sizes are counted in byte.

Scheme	A(**initial**)	B	A(**end**)	B(**end**)	**total**
FSXY [12]	6,238	14,779	10,124		31,141
BCNP-Lon [2, 28]	11,146	20,092	9,563		40,801
$\mathsf{AKE}_{\mathsf{SIDH\text{-}2}}$	6,828	13,917	6,641		27,386
$\mathsf{AKE}_{\mathsf{SIDH\text{-}3}}$	5,966	4,429	4,922	9,575	24,892

Table 4. 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.

6 Conclusion

In this paper, we investigate 1-Oracle SIDH problem and propose a CPCCA secure KEM. Then we build a compact and efficient 3-pass AKE based on 1-Oracle SIDH assumption and a 2-pass one based on DSIDH assumption. They are proved to be secure under the strongest CK$^+$ model and have an excellent property that arbitary registrant is allowed.

We have proved the security in random oracles, but not considered the standard model. It is non-trival for this proof because of the lack of a ring structure in SIDH. Hence, one of the future work is to prove the security in the standard model, and another direction is to consider the security in quantum security models in which the adversary can deliver quantum superpositions of messages, analogous to the one in [27].

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle diffie-hellman assumptions and an analysis of DHIES. In CT-RSA, pp. 143-158.
2. Boyd, C., Cliff, Y., Nieto, J. M. G., Paterson, K. G.: Efficient One-Round Key Exchange in the Standard Model. In ACISP 2008, pp 69-83.
3. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In CRYPTO 1993, pp. 232-249.
4. Costello, C., Jao, D., Longa, P., Naehrig, M., Renes, J., Urbanik, D.: Efficient compression of SIDH public keys. In EUROCRYPT 2017, pp. 679-706.
5. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In TCC 2010, pp. 453-474.

6. Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for supersingular isogeny Diffie-Hellman. In CRYPTO 2016, pp. 572-601.
7. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 64454 (1976).
8. De Feo, L., Jao, D., Plut, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology, 8(3), 209-247 (2014).
9. Faz-Hernádnez, A., López, J., Ochoa-Jimenez, E., Rodriguez-Henriquez, F.: A faster software implementation of the supersingular isogeny diffie-hellman key exchange protocol. IEEE Transactions on Computers (2017).
10. Fujisaki, E., Okamoto, T. : Secure integration of asymmetric and symmetric encryption schemes. In CRYPTO 1999, pp. 537-554.
11. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In PKC 2012, pp. 467-484.
12. 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.
13. Fujioka, A., Takashima, K., Terada, S., Yoneyama, K.: Supersingular Isogeny Diffie-Hellman Authenticated Key Exchange. IACR Cryptology ePrint Archive 2018/730.
14. Galbraith, S. D.: Authenticated key exchange for SIDH. IACR Cryptology ePrint Archive 2018/266.
15. Galbraith, S. D., Petit, C., Silva, J.: Identification protocols and signature schemes based on supersingular isogeny problems. In ASIACRYPT 2017, pp. 3-33.
16. Galbraith, S. D., Petit, C., Shani, B., Ti, Y. B.: On the security of supersingular isogeny cryptosystems. In ASIACRYPT 2016, pp. 63-91.
17. Guilhem, C. D. S., Smart, N. P., Warinschi, B.: Generic Forward-Secure Key Agreement Without Signatures. In ISC 2017, pp. 114-133.
18. Galbraith, S. D., Vercauteren, F.: Computational problems in supersingular elliptic curve isogenies. IACR Cryptology ePrint Archive 2017/774.
19. Jao, D., Azarderakhsh, R., Campagna, M., et al: Supersingular Isogeny Key Encapsulation. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions.
20. Jeong, I. R., Katz, J., Lee, D. H.: One-round Protocols for Two-Party Authenticated Key Exchange. In ACNS 2004, pp. 220-232.
21. Jao, D., Soukharev, V.: Isogeny-based quantum-resistant undeniable signatures. In Post-Quantum Cryptography, 2014, pp. 160-179.
22. Koziel, B., Azarderakhsh, R., Jao, D.: Side-Channel Attacks on Quantum-Resistant Supersingular Isogeny Diffie-Hellman. In SAC 2017, pp. 64-81.
23. Kelsey, J.: SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash. NIST Special Publication, 800, 185 (2016).
24. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In CRYPTO 2005, pp. 546-566.
25. Koziel, B., Azarderakhsh, R., Mozaffari-Kermani, M.: A High-Performance and Scalable Hardware Architecture for Isogeny-Based Cryptography. IEEE Transactions on Computers (2018).
26. 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).
27. LeGrow, J.: Post-Quantum Security of Authenticated Key Establishment Protocols. Master's thesis, University of Waterloo (2016).

28. Longa, P.: A Note on Post-Quantum Authenticated Key Exchange from Supersingular Isogenies. IACR Cryptology ePrint Archive 2018/267.
29. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In ProvSec 2007, pp. 1-16.
30. Menezes, A., Qu, M., Vanstone, S.: Some new key agreement protocols providing mutual implicit authentication. Selected Areas in Cryptography (1995).
31. Matsumoto, T., Takashima, Y., Imai, H.: On seeking smart public-key-distribution systems. IEICE TRANSACTIONS (1976-1990), 69(2), 99-106 (1986).
32. Okamoto, T.: Authenticated key exchange and key encapsulation in the standard model. In CRYPTO 2007, pp. 474-484.
33. Sun, X., Tian, H., Wang, Y.: Toward quantum-resistant strong designated verifier signature from isogenies. In INCoS 2012, pp. 292-296.
34. Urbanik, D., Jao, D.: SoK: The problem landscape of SIDH. IACR Cryptology ePrint Archive 2018/336.
35. Yoo, Y., Azarderakhsh, R., Jalali, A., Jao, D., Soukharev, V.: A post-quantum digital signature scheme based on supersingular isogenies. In FC 2017, pp. 163-181.

Appendix A, Proof of Lemma 2

In order to bound the probability of AskH, we investigate the events $\mathsf{AskH} \wedge \overline{\mathsf{AskG}} \wedge E_i$ for $1 \leq i \leq 8$ one by one.

Event $\mathsf{AskH} \wedge \overline{\mathsf{AskG}} \wedge E_3$
In the event E_3, the test session sid^* has no matching session, and the ephemeral secret key of U_B is given to $\mathcal{A}$. In case $\mathsf{AskH} \wedge \overline{\mathsf{AskG}} \wedge E_3$, the adversary $\mathcal{S}$ for solving 1-Oracle SIDH problem (actually based on Theorem 3, we consider $\mathcal{S}$ against CPCCA security of KEM_{dsidh}) as follows. It simulates the CK$^+$ games, and transforms the happening of event AskH performed by $\mathcal{A}$ to the advantage of solving 1-Oracle SIDH problem.

In order to simulate the random oracles, $\mathcal{S}$ maintains hash list L_G, $L_{\hat{H}}$ and L_{sk}, corresponding to the queries and answers of the G-oracle, $\hat{H}$-oracle and $\mathsf{SessionStateReveal}$, $\mathsf{SessionKeyReveal}$. $L_{\hat{H}}$ and L_{sk} are related. For example the adversary may ask L_{sk} without the encapsulated keys firstly, then ask $L_{\hat{H}}$ with the encapsulated keys. Thus, the reduction must ensure consistency with the random oracle queries to $L_{\hat{H}}$ and L_{sk}. The strong decapsulation oracle for KEM_{dsidh} could help to maintain the consistency as done in $\hat{H}$-oracle and $\mathsf{SessionKeyReveal}$ in the following.

On receiving the public key $(E_A, \phi_A(P_2), \phi_A(Q_2))$ from the CPCCA challenger in Theorem 3, to simulate the CK$^+$ game, $\mathcal{S}$ randomly chooses two parties U_A, U_B and the i-th session as a guess of the test session with success probability $1/N^2 l$. $\mathcal{S}$ 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 for U_A and only computes and sets the static public key for U_A as responder. Specially, $\mathcal{S}$ sets the static secret and public key pairs (pk_{B_2}, sk_{B_2}) for U_B as responder, and sets $pk_{A_1} = (E_A, \phi_A(P_2), \phi_A(Q_2))$ for U_A as initiator.

Without knowing the secret key of U_A as initiator, $\mathcal{S}$ chooses totally random r_1 as part of ephemeral secret key and totally random x. Since G is a hash function and sk_{A_1} is not queried, the difference between simulation with modification of r_1 and the real game can not detected by adversary. When a session state of a session owned by U_A is queried, $\mathcal{S}$ returns r_1 of this session as part of the ephemeral secret key.

On receiving the i-th session $(X^* = (E_X^*, R_2^*, S_2^*), x_1)$ from U_A (that is sent by $\mathcal{A}$ in the CK$^+$ game), $\mathcal{S}$ returns X to the CPCCA challenger and receives the challenge ciphertext (Y^*, y_1^*, y_2^*) (under public key pk_{A_1} and X^*) and $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$. Then $\mathcal{S}$ returns (Y^*, y_1^*, y_2^*) to U_A as the response of i-th session from U_B. $\mathcal{S}$ chooses a totally independent randomness r_2 as the ephemeral secret key of U_B for C^* and leaks it to adversary $\mathcal{A}$. Since G is a hash function, the difference between simulation with modification of r_2 and the real game can not be detected by the adversary.

$\mathcal{S}$ simulates the oracle queries of $\mathcal{A}$ and maintains the hash lists $L_G, L_{\hat{H}}, L_{sk}$ as follows. Specially, when AskH happens, which means $\mathcal{A}$ poses $(U_A, U_B, pk_{A_1}, pk_{B_2}, X^*, x_1, Y^*, y_1^*, y_2^*, x_2, K_A, K_B)$ to $\hat{H}$, where $(X^*, x_1, Y^*, y_1^*, y_2^*, x_2)$ is the views of the test session and K_B is the key encapsulated in (X^*, x_1, x_2) (this can be detected by $\mathcal{S}$ since it has sk_{B_2} and $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$ from CPCCA challenger), $\mathcal{S}$ returns K_A as the guess of K^* encapsulated in (Y^*, y_1^*, y_2^*), which contradicts with the CPCCA security of KEM_{dsidh} and further 1-Oracle SIDH assumption.

- Querying G-oracle with (r_i, ab_i) :

  1. If there exists a tuple $(r_i, ab_i, g_i) \in L_G$, $\mathcal{S}$ returns g_i, otherwise $\mathcal{S}$ randomly chooses g_i, returns g_i and records (r_i, ab_i, g_i) in L_G. Note that in the security definition $\mathcal{A}$ does not query $\mathsf{SessionStateReveal}(sid^*)$, thus $\mathcal{A}$ does not know any information of r_1.
  2. As shown in the following (in the Send queries, or when $\mathcal{S}$ generates the encapsulated key and the ciphertext using randomness), when $\mathcal{S}$ queries G-oracle with (r_1, a_1), if there does not exist $(r_1, a_1, \cdot) \in L_G$, generates randomness g, returns g and adds (r_1, a_1, g) into L_G
- Querying $\hat{H}$-oracle with $(U_P, U_Q, X, x_1, Y, y_1, y_2, x_2, K_P, K_Q)$
  1: If $P = A, Q = B, (Y, y_1, y_2) = (Y^*, y_1^*, x_2^*)$, and $(\mathit{\Pi}, I, U_A, U_B, X, x_1, Y, y_1, y_2, x_2)$ is the i-th session of U_A, and K_B is the key encapsulated in (X, x_1, x_2) (this can be judged by $\mathcal{S}$, since it has sk_{B_2} and $h(X^*, j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$ from CPCCA challenger), then $\mathcal{S}$ outputs the K_A as the key encapsulated in the challenge ciphertext (Y^*, y_1^*, x_2^*) of CPCCA games, that is K^*, sets $\mathsf{flag} = ture$.
  2: Else if $\exists\ (U_P, U_Q, X, x_1, Y, y_1, y_2, x_2, K_P, K_Q, h) \in L_{\hat{H}}$, return h,
  3: Else if $P = A$ and $\exists\ (U_A, U_Q, X, x_1, Y, y_1, y_2, x_2, K_A, K_Q, h) \in L_{sk}$:

     1. if (X, x_1, x_2) is sent by $\mathcal{A}$, $\mathcal{S}$ with the knowledge of sk_{Q_2} and y extracts K'_A encapsulated in X, x_1 and x_2. Since (Y, y_1, y_2) is generated by himself, $\mathcal{S}$ has the knowledge of encapsulated key K'_Q.

2. if (Y, y_1, y_2) is sent by $\mathcal{A}$, $\mathcal{S}$ with the knowledge of y queries the decapsulation oracle with X and ciphertext (Y, y_1, y_2) to extract the encapsulated key K'_Q;
3. if both (X, x_1, x_2) and (Y, y_1, y_2) are sent by $\mathcal{S}$, it has the knowledge of corresponding encapsulated key K'_A, K'_B.

If $(K_A, K_Q) = (K'_A, K'_Q)$, then $\mathcal{S}$ returns h and records $(U_A, U_Q, X, x_1, Y, y_1, y_2, x_2,\ K_P, K_Q, h)$ in the list $L_{\hat{H}}$.

4: Else if $Q = A$ and $\exists\ (U_P, U_A, X, x_1, Y, y_1, y_2, x_2, K_P, K_A, h) \in L_{sk}$: Since $\mathcal{S}$ has the static secret key of U_A as responder and static secret keys of U_P, it can extract encapsulated key K'_A in (Y, y_1, y_2) and encapsulated key K'_P in (X, x_1, x_2).
If $(K_P, K_A) = (K'_P, K'_A)$, then $\mathcal{S}$ returns h and records $(U_P, U_A, X, x_1, Y, y_1, y_2, x_2,\ K_P, K_A, h)$ in the list $L_{\hat{H}}$.

5: Else if $A \neq P, Q$ and $\exists\ (U_P, U_Q, X, x_1, Y, y_1, y_2, x_2,\ K_P, K_Q, h) \in L_{sk}$: Since $\mathcal{S}$ has the static secret keys of U_Q and static secret keys of U_P, it can extract encapsulated key K'_Q in (Y, y_1, y_2) and encapsulated key K'_P in (X, x_1, x_2).
If $(K_P, K_Q) = (K'_P, K'_Q)$, then $\mathcal{S}$ returns h and records $(U_P, U_Q, X, x_1, Y, y_1, y_2, x_2,\ K_P, K_Q, h)$ in the list $L_{\hat{H}}$;

6: otherwise, $\mathcal{S}$ returns a random value h and records $(U_P, U_Q, X, x_1, Y, y_1, y_2, x_2, K_P,\ K_Q, h)$ in the list $L_{\hat{H}}$.

– $\mathsf{Send}(\Pi, I, U_P, U_Q)$:
  1. If $P = A$, $\mathcal{S}$ generates two independent randomness $(r_1, n_1||n_2)$ (to pretend that $n_1||n_2 = G(r_1, sk_{A_1})$, although $\mathcal{S}$ does not know sk_{A_1}). This will not be detected by $\mathcal{A}$ as $\mathcal{A}$ does not ask G with sk_{A_1}). $\mathcal{S}$ computes (X, x_1) as in the protocol and sends (X, x_1) out.
  2. Otherwise, $\mathcal{S}$ proceeds as the protocols.

– $\mathsf{Send}(\Pi, R, U_Q, U_P, X, x_1)$:
  1. If $Q = B$ and this session is the i-th session of U_B, $\mathcal{S}$ sends X to CPCCA challenger as part of challenge public key, and gets (Y^*, y_1^*, y_2^*) as challenge ciphertext and also gets $h(X, j(E_X/\langle R_2 + [y]S_2\rangle))$. Then $\mathcal{S}$ returns (Y^*, y_1^*, y_2^*).
  2. Otherwise, $\mathcal{S}$ chooses SK randomly.

– $\mathsf{Send}(\Pi, R, U_Q, U_P, X, x_1, Y, y_1, y_2)$: $\mathcal{S}$ computes the session key and maintains the session key list L_{sk} as follows.
  1. If $P = A$, $\mathcal{S}$ computes x_2 (using secret key x) and with the knowledge of $n_1||n_2$ computes the key K'_A encapsulated in (X, x_1, x_2). $\mathcal{S}$ queries the CPCCA decapsulation oracle with X and Y, y_1, y_2. Since $(X, Y, y_1, y_2) \neq (X^*, Y^*, y_1^*, y_2^*)$, the decapsulation oracle will return K'_Q encapsulated in ciphertext (Y, y_1, y_2) under public key (pk_{A1}, X). If $\exists(U_A, U_Q, X, x_1, Y, y_1, y_2, x_2, K_A, K_Q, h) \in L_{\hat{H}}$, $\mathcal{S}$ does the following: if $(K'_A, K'_Q) = (K_A, K_Q)$, sets $SK = h$.
  2. Otherwise, $\mathcal{S}$ computes and returns x_2. $\mathcal{S}$ has the knowledge of sk_{P_1}, x, and sk_{Q_2}, and can extract the encapsulated key K'_P and K'_Q in (X, x_1, x_2) and (Y, y_1, y_2). If $\exists(U_A, U_Q, X, x_1, Y, y_1,\ y_2, x_2, K_A, K_Q, h) \in L_{\hat{H}}$, $\mathcal{S}$ does the following: if $(K'_P, K'_Q) = (K_P, K_Q)$, sets $SK = h$.

3. Otherwise, $\mathcal{S}$ chooses SK randomly.

$\mathcal{S}$ records this as the completed session and adds $(U_A, U_Q, X, x_1, Y, y_1, y_2, x_2, SK)$ to the session key list L_{sk}.

- $\mathsf{Send}(\Pi, I, U_P, U_Q, X, x_1, Y, y_1, y_2, x_2)$: With the knowledge of sk_{Q_2} and y, $\mathcal{S}$ could extract K'_P. The key K'_Q encapsulated in (Y, y_1, y_2) is computed by $\mathcal{S}$ himself. If there exists $((U_A, U_Q, X, x_1, Y, y_1, y_2, x_2, K_P, K_Q, h) \in L_{\hat{H}}$ and $K'_P = K_P, K'_Q = K_Q$, set $SK = h$. Otherwise, $\mathcal{S}$ chooses SK randomly. $\mathcal{S}$ records this as the completed session and adds $(U_P, U_Q, X, x_1, Y, y_1, y_2, x_2, SK)$ to the session key list L_{sk}.
- Querying $\mathsf{SessionKeyReveal}(\mathsf{sid})$: The session key list L_{sk} is maintained as in the Send queries.
  1. If the session sid is not completed, $\mathcal{S}$ aborts.
  2. Else if sid is recorded in the list L_{sk}, $(U_P, U_Q, X, x_1, Y, y_1, y_2, x_2, SK) \in L_{sk}$, then returns SK.
  3. Otherwise, $\mathcal{S}$ returns a random value SK and records it in L_{sk}.
- Querying $\mathsf{SessionStateReveal}(\mathsf{sid})$: As the definition of freshness, sid is not the test session.
  1. If the owner of sid is A, and A is an initiator. The session state is generated by himself or extractable from the decapsulation oracle. $\mathcal{S}$ just returns them.
  2. If the owner of sid is A, and A is a responder. The session state is generated by himself. $\mathcal{S}$ just returns them.
  3. Otherwise, $\mathcal{S}$ holds the secret key of other users and could return the session state as the definition.
- Querying $\mathsf{Corrupt}(U_P)$

  $\mathcal{S}$ returns the static secret key of U_P.
- $\mathsf{Test}(\mathsf{sid})$

  If sid is not the i-th session of U_A, $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ responds to the query as the definition above.
- If $\mathcal{A}$ outputs a guess b', $\mathcal{S}$ aborts with failure.

The simulator $\mathcal{S}$ maintains the consistency of $\hat{H}$-oracle, h-oracle, $\mathsf{SessionStateReveal}$ and $\mathsf{SessionKeyReveal}$ with the decryption oracle of KEM_{dsidh}. Note that in the first case in the $\hat{H}$-oracle, if $\mathsf{flag} = ture$, then $\mathcal{S}$ would succeed in the CPCCA game. Thus $\Pr[\mathsf{AskH} \wedge E_3] \leq N^2 l \cdot \mathbf{Adv}^{\text{CPCCA}}_{\mathsf{KEM}_{dsidh}}(\mathcal{S}) \leq N^2 l(\frac{q_{hash}+1}{2^\lambda} + 4\mathbf{Avd}^{\text{1-OSIDH}}_{\mathcal{S}})$.

Event $\mathsf{AskH} \wedge E_1$

In the event E_1, the test session sid^* (with owner as initiator) has no matching session, and the static secret key of U_A is given to $\mathcal{A}$. In case $\mathsf{AskH} \wedge E_1$, the 1-Oracle (A-)SIDH problem is replaced with the 1-Oracle B-SIDH problem as noted in Remark 5. The 1-Oracle SIDH adversary $\mathcal{S}$ simulates the CK$^+$ games and transforms the happening of event AskH performed by $\mathcal{A}$ to the advantage of solving 1-Oracle SIDH problem.

The difference with Event $\mathsf{AskH} \wedge E_3$ is that the underlying assumption is replaced to 1-Oracle B-SIDH and the static secret key of U_A as initiator is unknown. The other part of analysis is the same.

Event $\mathsf{AskH} \wedge E_2$

In the event E_2, the test session sid^* (with owner as initiator) has no matching session, and the ephemeral secret key of U_A is given to $\mathcal{A}$. In case $\mathsf{AskH} \wedge E_2$, the 1-Oracle SIDH adversary $\mathcal{S}$ simulates the CK$^+$ games, and transforms the happening of event AskH performed by $\mathcal{A}$ to the advantage of solving 1-Oracle SIDH problem.

The only difference with Event $\mathsf{AskH} \wedge E_1$ is that the ephemeral secret key r_1 is leaked to $\mathcal{S}$ rather than sk_{A_1}. This is fixed by the hash function G which is modeled as a random oracle.

Event $\mathsf{AskH} \wedge E_4$

In the event E_4, the test session sid^* (with owner as responder) has no matching session, and the static secret key of U_B is given to $\mathcal{A}$. In case $\mathsf{AskH} \wedge E_4$, the 1-Oracle SIDH adversary $\mathcal{S}$ simulates the CK$^+$ games, and transforms the happening of event AskH performed by $\mathcal{A}$ to the advantage of solving 1-Oracle SIDH problem.

Event $\mathsf{AskH} \wedge E_5$

In event E_5, 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 $\mathcal{A}$. In this case, the DSIDH adversary $\mathcal{S}$ performs as follows. It simulates the CK$^+$ games, and transforms the happening of event AskH performed by $\mathcal{A}$ to the advantage of attacking DSIDH problem. Since we know that if the 1-Oracle SIDH assumption holds, the DSIDH assumption holds. This event is also bounded by $\mathbf{Avd}_{\mathcal{B}}^{\mathsf{1\text{-}OSIDH}}$.

Event $\mathsf{AskH} \wedge E_6$

In event E_6, the test session sid^* has a matching session $\overline{\mathsf{sid}}^*$. Both ephemeral secret keys of the initiator and the responder are leaked to $\mathcal{A}$. This is almost the same with Event $\mathsf{AskH} \wedge E_3$.

Event $\mathsf{AskH} \wedge E_{7\text{-}1}$

In event $E_{7\text{-}1}$, the test session sid^* has a matching session $\overline{\mathsf{sid}}^*$. Both the ephemeral secret key of the responder and the static secret key of the initiator are leaked to $\mathcal{A}$. This is almost the same with Event $\mathsf{AskH} \wedge E_1$. In this case, the only difference is that the ephemeral secret key of U_B is leaked to $\mathcal{A}$, which does not affect the proof.

Event $\mathsf{AskH} \wedge E_{7\text{-}2}$

In event $E_{7\text{-}2}$, the test session sid^* has a matching session $\overline{\mathsf{sid}}^*$. Both the ephemeral secret key of the initiator and the static secret key of the responder are leaked to $\mathcal{A}$. This is almost the same with Event $\mathsf{AskH} \wedge E_4$. In this case, the only difference is that the ephemeral secret key of U_A is leaked to $\mathcal{A}$, which does not affect the proof.

Event $\mathsf{AskH} \wedge E_{8\text{-}1}$

In event $E_{8\text{-}1}$, the test session sid^* has a matching session $\overline{\mathsf{sid}}^*$. Both the ephemeral secret key of the initiator and the static secret key of the responder are leaked to $\mathcal{A}$. This is almost the same with Event $\mathsf{AskH} \wedge E_{7\text{-}2}$. In this case, the only difference is the owner of the test session, which does not affect the proof.

Event $\mathsf{AskH} \wedge E_{8\text{-}2}$

In event $E_{8\text{-}2}$, the test session sid^* has a matching session $\overline{\mathsf{sid}}^*$. Both the static secret key of the initiator and the ephemeral secret key of the responder are leaked to $\mathcal{A}$. This is almost the same with Event $\mathsf{AskH} \wedge E_{7\text{-}1}$. In this case, the only difference is the owner of the test session, which does not affect the proof.