Strongly Secure Authenticated Key Exchange from Supersingular Isogeny

Xiu Xu[1,2], Haiyang Xue[1,2⋆], Kunpeng Wang[1,2], Song Tian[1,2], Bei Liang[3], 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 three-pass AKE based on 1-Oracle SIDH assumption whose soundness is guaranteed by a strictly limited gap problem. To enhance the soundness, we propose a two-pass AKE based on standard SIDH assumption. The three-pass AKE achieves about 20% speedup compared with the SIDH variant of FSXY scheme and narrows the bandwidth by approximately 49.3% without lose of security. And the two-pass scheme narrows the bandwidth by around 23% and yields a factor 12% acceleration than the SIDH variant of FSXY scheme.
In the random oracle model, both three-pass AKE and two-pass AKE protocols are secure in the CK model, support arbitrary registration of public key and are resistant to the weak perfect forward secrecy (wPFS) attack, key-compromise impersonation (KCI) attack and maximal exposure (MEX) attack, which solves the open problem provided by Galbraith of looking for new techniques to design and prove security of AKE in 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 [39] 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 [4, 6, 9, 29] and how to implement authenticated key exchange (AKE) with high efficiency [3, 10, 9, 20, 29, 31, 32]. A plenty of security models has been proposed, including BR model [4], CK model [6] and eCK model [29]. CK^+ security model that is known to be one of the "strongest" and most "desirable" security notions [24] for AKE is reformulated by Fujioka *et al.* [9]. The CK^+ model not only covers the security requirement in CK model, but also captures some advanced attacks such as key compromise impersonation (KCI), the breaking of weak perfect forward secrecy (wPFS) and maximal exposure attacks (MEX). Therefore, in some

⋆ Corresponding author email: xuehaiyang@iie.ac.cn

sense 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.

AKE protocols have two types of authentication: the implicit authentication [32] which is implemented by each party individually to generate the session key, or vice versa explicit authentication. Since on the one hand the implicit AKE protocols enable to provide significantly computational savings thus resulting in highly efficient communication, on the other hand by adding key-confirmation flows [40] it is possible to achieve the explicit AKE from an AKE protocol that only provides implicit authentication, many works on constructing implicit AKE protocols based on concrete assumptions [24, 29, 30, 37] as well as studying the generic frameworks [3, 10, 9, 16] has been investigated.

Supersingular Isogeny. Apart from lattice, code, hash and multivariate cryptography, supersingular elliptic curve isogeny is a fresh and 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 [18, 19] proposed a supersingular isogeny Diffie-Hellman key exchange (SIDH) that is considered as one of the post-quantum candidates. There are many following works which mainly focus on the computational efficiency [7, 8, 23], key compression [2, 5, 38], adaptive attacks on SIDH [15, 22], the relationship of underlying complexity problems [13, 35], signature schemes [14, 21, 34, 36] and its standardization [17, 25].

The SIDH protocol is analogous to the traditional Diffie-Hellman protocol and so is vulnerable to the meet-in-middle attack. As in the survey where Galbraith concluded [12], to our best knowledge that there are few papers that investigate the natural problems of designing AKE schemes from the basic SIDH primitive. As we all know one 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[14, 36]. Therefore, it is essential to exploring on designing *implicit* AKE from SIDH primitives. However, as Galbraith [12] pointed out, when working in the SIDH setting there are several challenges arising in order to adapt 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 [31] and HMQV [24], require a richer algebraic structure which supersingular isogeny does have.
- The protocols involving long-term/static secret keys are vulnerable to the adaptive attack [15] where the secret key can be extracted bit-by-bit.
- The gap assumption that holds in discrete logarithm setting is crucial for security proof. But the gap assumption does not hold in SIDH context when polynomial queries are submitted to *unlimited* decisional solver.

We should keep in mind that the adaptive attack introduced by Galbraith, Petit, Shani and Ti [15] works on the situation where the static public key is used. 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. [15] 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.

Motivation. As shown by Galbraith [12] and Longa [28], the generic constructions of secure AKE from basic primitives like IND-CCA encryption/KEMs and MACs, PRFs etc, such as the schemes proposed by Boyd, Cliff, Gonzlez Nieto and Paterson [3] (abbreviated as BCNP scheme), by Fujioka, Suzuki, Xagawa and Yoneyama [10] (abbreviated as FSXY scheme)

and by Guilhem, Smart and Warinschi [16] (abbreviated as GSW scheme) are easily to be adapted to the SIDH context by plugging an IND-CCA secure KEM based on SIDH. However such transformations lead to more isogeny computations or more round of communication which are examined and summarised in the table 1 of [12]. Here we reexamine the security model they are satisfied, the assumption they are based on as well as the computation they cost, and make more concrete comparison among these resulted SIDH schemes in the following Table 1.

With respect to non-generic construction, both the Jeong-Katz-Lee scheme [20] and NAXOS scheme [29] only involve the computations like g^{xy} rather than g^{ax+b}, thus being suitable to the SIDH case. Furthermore, Galbraith [12] proposed two SIDH-AKE schemes, and in this paper we called them Gal 1 which is a variant of the Jeong-Katz-Lee scheme [20] TS2 and Gal 2 which is closely related to NAXOS scheme. But their schemes could only satisfy the security with limited adversarial abilities (details are given in section 1.3). Several recognized attacks are not considered, including arbitrary registrant for static public keys, KCI attacks, and MEX attacks. In a AKE system, the adversary-controlled parties may register arbitrary public keys. KCI attacks mean that if a static secret key is revealed, an adversary can try to impersonate any other honest party in order to fool the owner of the exposed secret key. In MEX, an adversary tries 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.

For example, the arbitrary registration for the static public key is not allowed for Gal 1 scheme. Otherwise Gal 1 scheme can not prevent adaptive attack: one malicious static public key can extract one bit of a target secret key. Furthermore, Gal 1 is not resistant to KCI and MEX attacks. Thus, this paper is to promote solving the open problem given by Galbraith [12]:

"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."

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 both of security in the CK^+ model, which includes the widest possible adversarial goals.

1. As preparation work for 3-pass AKE, we investigate the soundness of the hashed decisional SIDH problem where the adversary is allowed to query a one-time hashed computational SIDH oracle, which we call 1-Oracle SIDH problem. 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 at the first time) to the decisional SIDH (DSIDH) oracle.
2. We propose a strongly secure key encapsulation mechanism (KEM) based on supersingular isogeny, which is served as the core building block of our AKE schemes. Our KEM is *chosen public-key chosen ciphertext* (CPCCA) secure under the standard DSIDH assumption. Based on the 1-Oracle SIDH assumption, it is still CPCCA secure even some information of challenge ciphertext is leaked.
3. Equipped with 1-Oracle SIDH assumption and strongly secure KEM as the building block, we propose a 3-pass AKE $\mathsf{AKE}_{\mathsf{SIDH\text{-}3}}$ and prove its security in the CK^+ model.

4. To enhance the soundness 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 in security and efficiency compared with existing works. Then we implement some typical schemes and give a comparison in Section 6. The 3-pass scheme decreases the bandwidth by 49.3% and is 1.2 times faster than the generic construction in [10] without loss of security. The 2-pass AKE scheme narrows the bandwidth by 23% and is 1.12 times faster.

Scheme	Model	Assum.	Key Reg.	wPFS	KCI	MEX	Rd	Init isog	Resp isog	Mess Size
SIDH [19]	-	DSIDH	-	×	×	×	2	2	2	72λ
Gal 1 [12]	CK	CSIDH	Honest	✓	×	×	2	3	3	108λ
Gal 2 [12]	BR	CSIDH	Honest	✓	✓	×	2	4	4	108λ
GSW [16]	CK	DSIDH	Honest	✓	×	×	3	6	6	186λ
BCNP-Lon [3, 28]	CK	DSIDH	Arbi.	✓	✓	×	2	6	6	148λ
FSXY [10]	CK^+	DSIDH	Arbi.	✓	✓	✓	2	7	6	148λ
$\mathsf{AKE}_{\mathsf{SIDH\text{-}2}}$	CK^+	DSIDH	Arbi.	✓	✓	✓	2	6	5	114λ
$\mathsf{AKE}_{\mathsf{SIDH\text{-}3}}$	CK^+	1-O SIDH	Arbi.	✓	✓	✓	3	5	5	80λ

Table 1. Comparison of existing AKE protocols on supersingular isogeny. SIDH is an unauthenticated scheme. Gal 1 is the Jeong-Katz-Lee variant in [12]. Gal 2 is the variant of the NAXOS-like scheme in the appendix of [12]. **Assump.** is the shortage of assumptions. **Key Reg.** means whether the static public key allows arbitrary registration or not, where "Arbi" allows arbitrary registration and "Honest" only allows honest registration. **Rd** is the pass number of protocol. **Init isog** and **Resp isog** represent the number of isogeny that the initiator and responder have to compute. **Mess Size** represents the total message size. "✓" presents that the scheme can resist this kind of attack and "×" cannot. "−" shows that this kind of security model does not consider this query.

1.2 Our Techniques

Our core ideas and techniques are illustrated in Figure 1. E_0 is the starting point.

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, for example 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 $(E_{B_2}, \phi_{B_2}(P_1), \phi_{B_2}(Q_1))$.

As shown in Figure 1, there is a natural way to extract a session key from 4 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 [15]. 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 result session key. As shown in Figure 1 with dashed lines, Galbraith [12] involves $E_{A_1B_2}$ and E_{XY} for the session key. Under the adaptive attack [15], 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 registration is allowed.

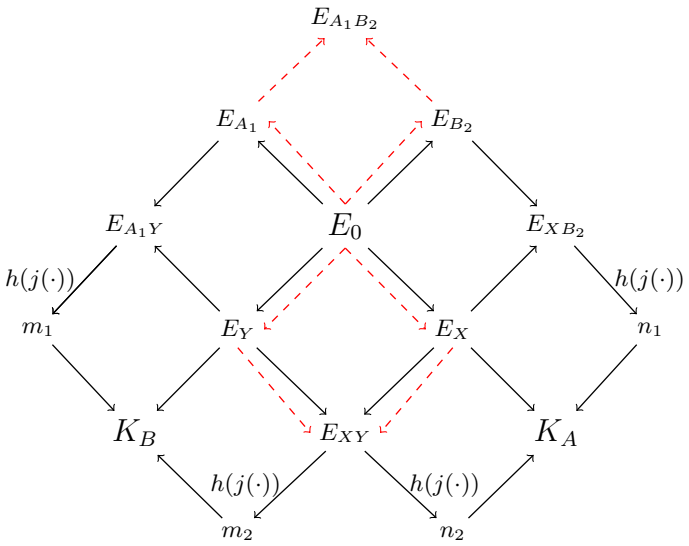

Fig. 1. Illustration of the core idea of our 3-pass AKE. The red dashed lines illustrate the core ideas of Gal 1 scheme [12].

Although now only E_{A_1Y}, E_{XB_2}, and E_{XY} are involved in the session key, the adaptive attack still works 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 still could 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 the "re-encryption" approach used in Fujisaki-Okamoto (FO) transformation [11]. 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 such that the adaptive attack does not work now. 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 the adaptive attack.prepr

Although the CCA secure KEM with "re-encryption" avoids the adaptive attack, it is still not enough 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 it can choose part of the challenge public key X^* for (Y^*, y_1^*, y_2^*). The capability of querying the *session state and session key* of non-test session means that it can query the decapsulation oracle which decapsulates the ciphertext under several public keys X' (not only the challenge public key). We integrate such attack manner as the chosen public key and chosen ciphertext attack (CPCCA), inspired by Okamoto [33]. Although the CPCCA adversary is much stronger than CCA adversary, in the random oracle model the solution to make our CCA secure KEM

remain secure against CPCCA adversary is not complex. We only need 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 solve the problem 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) which encapsulates K_A. In the test session, (X, x_1, x_2) is sent by AKE adversary $\mathcal{A}$, the simulator 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, $\mathcal{S}$ 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 $h(j(E_X/\langle R_2^* + [y]S_2^* \rangle))$ can 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 and 2-pass AKE in section 5, 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 is a hashed DSIDH assumption that a one-time hashed CSIDH oracle is allowed. 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.

Galbraith [12] proposed two SIDH variants of AKE from Jeong-Katz-Lee protocol [20] and NAXOS protocol [29]. Considering the adaptive attack on static secret keys, Gal 1 protocol only allows honest registrant of static public keys, and can not stand against KCI and MEX attacks. 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 MEX attacks.

Some generic AKE schemes can be adapted to SIDH variant. Boyd et al. [3] proposed a simple scheme (called BCNP) consisting of two KEM schemes and an additional unauthenticated key exchange in parallel. But it can not resist MEX attacks. Guilhem, Smart and Warinschi [16] gave a three round AKE, which encrypts the message of unauthenticated key exchange to achieve CK security. It is vulnerable to the KCI and MEX attacks. Fujioka et al. [10] provided a CK$^+$ secure AKE protocol (called FSXY) combining two CCA secure KEMs with a chosen plaintext secure KEM. The SIDH variant of FSXY frame achieves the highest secure level against the widest possible adversarial goals. Recently, Longa [28] proposed some experiment result for the SIDH variant for the BCNP-frame.

2 Preliminaries

2.1 SIDH Key Exchange Based on Supersingular Isogeny

The SIDH protocol [18, 19] inherits the construction of general Diffie-Hellman protocol, but it has 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, where we adopt the notations in [19] 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, $E_0[l_1^{e_1}] = Z/l_1^{e_1}Z \oplus Z/l_1^{e_1}Z = \langle P_1, Q_1 \rangle$, $E_0[l_2^{e_2}] = Z/l_2^{e_2}Z \oplus Z/l_2^{e_2}Z = \langle P_2, Q_2 \rangle$. The above parameters are public. Then A secretly chooses two random elements $m_A, n_A \in Z/l_1^{e_1}Z$, 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 $m_B, n_B \in Z/l_2^{e_2}Z$, 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, $(E_0/\langle R_A \rangle)/(\langle R_A, R_B \rangle/\langle R_A \rangle) = E_0/\langle R_A, R_B \rangle$. In the same way, 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 [15], 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 reminder 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 [9], which is a CK model [6] integrated with the weak PFS, resistance to KCI and MEX properties. We focus on **three and two-pass protocols** in this paper. For simplicity, we only show the model specified to two-pass protocols. As for three pass protocol, we can extend it by adding an extra message in the matching session identifier and Send definitions.

In 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 (ssk_i, spk_i), where the static public key is linked 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 [6] 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, then may computes 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): $\mathcal{A}$ could send message 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 queries: 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 if the guess of the adversary is correct, i.e., $b' = b$. The advantage of the adversary $\mathcal{A}$ is defined as $\mathsf{Adv}_{\Pi}^{CK^+}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}$.

Definition 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 $\mathcal{A}$, $\mathsf{Adv}_{\Pi}^{CK^+}(\mathcal{A})$ *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^*	$\overline{\mathsf{sid}}^*$	ssk_A	esk_A	esk_B	ssk_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. ssk_A (ssk_B) means the static secret key of A (B). esk_A (esk_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

In this section, we recall the complexity problems related to supersingular isogeny, where the first five are standard assumptions, the sixth is straightforward by adding a hash function and the last two are new. We also analyze their relations and plausibility.

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 [13]). *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 [13]). *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 [18]). *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 [18]). *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') are chosen at random from $Z/l_1^{e_1}Z$ (respectively $Z/l_2^{e_2}Z$) and not both divisible by l_1 (respectively l_2). The DSIDH problem is decide how E_C is computed.*

Definition 6 (A/B-DSIDH problem [35]). *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 DSIDH problem is different with both A-DSIDH and B-DSIDH problems, since DSIDH problem requires that $R_2 = \phi_A(P_2), S_2 = \phi_A(Q_2)$, while A-DSIDH does not require this "public key validation" [35].
Remark 2: Note that in [35] 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 [13] show that the computational SI isogeny is equal to the decisional SI problem. They show that l_1 queries to 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 call 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 [35] show that a solver of A-SI problem is equivalent under randomized polynomial time reductions to a solver which solves both CSIDH and A-DSIDH problems.

Galbraith *et al.* [15] 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 found that the known pubic key validation methods are insufficient and show that the attacker will 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 under reduction for randomized polynomial times to A-DSIDH or B-DSIDH problem.

3.2 1-Oracle SIDH Problem

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

In the supersingular isogeny area, the classical "gap" problem does not hold, namely computational SIDH problem (resp. SI) does not hold [15, 13] if allowing 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-DSIDH problem (resp. Decisional SI) solver.

Although the classical "gap" assumption does not hold, we believe that the "gap" assumption may hold with a **strictly limited** decisional oracle. Precisely, we believe that the CSIDH is still hard even if allowing queries with ***only one*** $(E_X, R_2, S_2 \in E_X[l_2^{e_2}])$ and polynomial different curves E_Z to the decisional A-DSIDH problem. 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 $h = H(j(E_{AB}))$ where $E_{AB} \cong E_0/\langle G_A, G_B\rangle$ 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 such that given the input a curve E_X and a basis pair $\langle R_2, S_2\rangle \in E_X[l_2^{e_2}]$, compute and output 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 a curve E_X and a basis pair $\langle R_2, S_2\rangle \in E_X[l_2^{e_2}]$, outputs $H(j(E_Z))$ where $E_Z \leftarrow \mathcal{O}_B(E_X, R_2, S_2)$ and H is a hash function. Suppose 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 $H(j(E_{AB}))$ where $E_{AB} = E_0/\langle G_A, G_B\rangle$. Now, as long as the one-time oracle $\mathcal{H}_B$ can not be queried with $(E_A, \phi_A(P_2), \phi_A(Q_2))$, this one-time oracle seems useless. We formalize this 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 $h = H(j(E_{AB}))$ where $E_{AB} \cong E_0/\langle G_A, G_B\rangle$ 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*

$$\begin{aligned}\mathbf{Avd}_{\mathcal{A}}^{\text{1-OSIDH}} = &\Pr[\mathcal{A}^{\mathcal{H}_B}\big(E_A, \phi_A(P_2), \phi_A(Q_2), E_B, \phi_B(P_1), \phi_B(Q_1); H(j(E_{AB}))\big) = 1] - \\ &\Pr[\mathcal{A}^{\mathcal{H}_B}\big(E_A, \phi_A(P_2), \phi_A(Q_2), E_B, \phi_B(P_1), \phi_B(Q_1); h \leftarrow \{0,1\}^\lambda\big) = 1].\end{aligned}$$

We emphasize that the adversary is allowed to query Hashed SIDH oracle $\mathcal{H}_B$ only one time 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 [13].

Remarks 3: In definition 8, 1-Oracle SIDH solver is associated with one-time oracle $\mathcal{H}_B$. We may also associate the solver with one-time oracle $\mathcal{H}_A$. 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 only 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 $\exists E_Z$ such that $j(E_Z) = j'$ and $E_Z \cong E_X/\langle R_2 + [b]S_2\rangle$, 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:
01 On receiving $E'_X, R'_2, S'_2 \in E'_X[l_2^{e_2}]$ and a j-invariant j'
02 if count=0
03 $(E_X, R_2, S_2) := (E'_X, R'_2, S'_2)$, count=1
04 if $\exists E_Z$ s.t. $j(E_Z) = j' \wedge E_Z \cong E_X/\langle R_2 + [b]S_2\rangle$
05 return 1 else 0.
06 if count = 1
07 if $(E_X, R_2, S_2) \neq (E'_X, R'_2, S'_2)$, **abort**
08 else if $\exists E_Z$ s.t. $j(E_Z) = j' \wedge E_Z \cong E_X/\langle R_2 + [b]S_2\rangle$
09 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 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). *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 C-SIDH 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 adversary $\mathcal{A}$ is*

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

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 [15]. Thus we require a highly limited oracle $\mathcal{O}_B((\cdot),\cdot)$.

Remark 4: As in Remark 3, 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 $\mathcal{O}_A((\cdot),\cdot)$.

3.3 1-Oracle SIDH and 1-gap SIDH

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 analysis of the Oracle Diffie-Hellman assumption given by Abdalla, Bellare and Rogaway [1]. More specifically, we show that in the random oracle model if there is a algorithm $\mathcal{A}$ to solve the 1-Oracle SIDH there is a algorithm $\mathcal{B}$ to solve the 1-gap SIDH problem.

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

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

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 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.

Note that in Figure 3, if $\mathcal{H}_B(E_X, R_2, S_2)$ is asked at first which returns a random h, and later j' is queried to H such that $\mathcal{O}_B((E_X, R_2, S_2), j') = 1$, it will return h. If $H(j')$ is asked at first which returns a random h, and later (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 $j(E_0/\langle G_A, G_B\rangle)$ is queried to H and $\overline{\mathsf{Ask}}$ be the complement of Ask. If Ask does not happen, which means $j(E_0/\langle G_A, G_B\rangle)$ is not queried by $\mathcal{A}$ to H, there is no way to tell whether h_b is equal to $H(j(E_0/\langle G_A, G_B\rangle))$ 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 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 $\mathsf{non\text{-}SIDH}$ distribution. Thus we have that

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

□

Algorithm $\mathcal{B}^{\mathcal{O}_B((\cdot),\cdot)}(A=(E_A,\phi_A(P_2),\phi_A(Q_2));B=(E_B,\phi_B(P_1),\phi_B(Q_1)))$	
01 $h_0,h_1 \leftarrow \{0,1\}^\lambda$	<u>One time $\mathcal{H}_B$</u>
02 $b \leftarrow \{0,1\}$	10 given one-time query (E_X,R_2,S_2)
03 Run $\mathcal{A}^{\mathcal{H}_B(\cdot),H}(A,B,h_b)$	11 if $\exists(j',h') \in L_H \wedge \mathcal{O}_B((E_X,R_2,S_2),j')=1$
04 a. For one-time query $\mathcal{H}_B$	12 return h'
05 do as one-time $\mathcal{H}_B$	13 else $h \leftarrow \{0,1\}^\lambda$, $L_B = L_B \cup \{h\}$
06 b. For the H-query	14 return h
07 do as $H(j')$	<u>$H(j')$</u>
08 c. Let b' be the output of $\mathcal{A}$	15 if $\exists(j',h') \in L_H$ return h'
09 return $j \leftarrow L_H$	16 else if $\exists((E_X,R_2,S_2),h) \in L_B \wedge \mathcal{O}_B((E_X,R_2,S_2),j')=1$
	17 return h, $L_H = L_H \cup \{(j',h)\}$
	18 otherwise $h \leftarrow \{0,1\}^\lambda$
	19 return h, $L_H = L_H \cup \{(j',h)\}$

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

4 A Three-Pass AKE from Supersingular Isogeny

We propose a strongly secure 3-pass AKE from supersingular isogeny whose security relies on the 1-Oracle SIDH assumption.

4.1 Chosen Public Key CCA Secure KEM

We first propose a KEM from supersingular isogeny which is the core building block of our AKE. The KEM is secure against the adversary who is allowed to choose not only part of the challenge public key but also query strong decryption oracle which will decrypt the ciphertext under several public keys rather than only the challenge public key.

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 $h : \{0,1\}^* \rightarrow \{0,1\}^{2\lambda}$, $g : \{0,1\}^* \rightarrow \{0,1\}^*$, and $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ be hash functions, where λ is the security parameter. The KEM is shown in Figure 4.

$\mathsf{KeyGen}(\lambda)$. Choose a random element $a_1 \leftarrow Z/l_1^{e_1}Z$ and compute the isogeny $\phi_{A_1} : E_0 \rightarrow E_{A_1}$ 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 $x \leftarrow Z/l_1^{e_1}Z$ and compute the isogeny $\phi_x : E_0 \rightarrow E_X$ 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 $\phi_Y : E_0 \rightarrow E_Y$ 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 $\phi_{YA_1} : E_{A_1} \rightarrow E_{YA_1}$ and $\phi_{YX} : E_X \rightarrow E_{YX}$ 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 $y_2 = h(j(E_{YX})) \oplus m_2$. The session key is $K = H(X, m_1, m_2, C)$.

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

Remark 5: In this scheme, these points in the public key are chosen from subgroups of order $l_2^{e_2}$, and these points in the first part of ciphertext are chosen from subgroups of order $l_1^{e_1}$. We can change the subgroups by replacing A with B and X with Y. And now the underlying assumption is altered from 1-Oracle B-SIDH to 1-Oracle A-SIDH as in Remark 3.

$\mathsf{KeyGen}(\lambda)$	
01 $a_1 \leftarrow Z/l_1^{e_1}Z$	05 $x \leftarrow Z/l_1^{e_1}Z$
02 $E_{A_1} = E_0/\langle P_1 + [a_1]Q_1\rangle$	06 $E_X = E_0/\langle P_1 + [x]Q_1\rangle$
03 $\phi_{A_1}(P_2), \phi_{A_1}(Q_2)$	07 $\phi_X(P_2), \phi_X(Q_2)$
04 $A_1 = \big(E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2)\big)$	08 $X = \big(E_X, \phi_X(P_2), \phi_X(Q_2)\big)$
	09 $pk = (A_1, X), sk = (a_1, x)$

$\mathsf{Encaps}(pk)$	$\mathsf{Dec}(sk, C)$
01 $m_1, m_2 \leftarrow \{0,1\}^\lambda$	01 $(c_1, c_2, c_3) \leftarrow C$
02 $y = g(m_1, m_2)$	02 $E_{A_1Y} = E_Y/\langle\phi_Y(P_1 + [a_1]Q_2)\rangle$
03 $E_Y = E_0/(P_2 + [y]Q_2)$	03 $E_{XY} = E_Y/\langle\phi_Y(P_1 + [x]Q_2)\rangle$
04 $Y = \big(E_Y, \phi_Y(P_1), \phi_Y(Q_1)\big)$	04 $m_1' = y_1 \oplus h(j(E_{A_1Y}))$
05 $E_{YA_1} = E_{A_1}/\langle\phi_{A_1}(P_2) + [y]\phi_{A_1}(Q_2)\rangle$	05 $m_2' = y_2 \oplus h(j(E_{XY}))$
06 $E_{YX} = E_X/\langle\phi_X(P_2) + [y]\phi_X(Q_2)\rangle$	06 $y' = g(m_1', m_2')$
07 $y_1 = h(j(E_{YA_1})) \oplus m_1$	07 if $E_Y = E_0/\langle P_2 + [y']Q_2\rangle$
08 $y_2 = h(j(E_{YX})) \oplus m_2$	08 return $H(X, m_1', m_2', C)$
09 $C = (Y, y_1, y_2), K = H(X, m_1, m_2, C)$	09 else $\perp$

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

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 strong decryption oracle. Note that $\mathcal{A}$ has the capability to choose X^* and query strong decryption oracle that will decrypt the ciphertext under general public keys generated by the challenger.

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 $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{CPCCA}}_{\mathcal{A}}$ is negligible.

Theorem 2. *Under DSIDH assumption,* KEM_{dsidh} *is CPCCA secure in the random oracle model. Precisely, for any PPT CPCCA adversary* $\mathcal{A}$ *there exists algorithm* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{CPCCA}_{\mathcal{A},\mathsf{KEM}_{dsidh}} \leq \mathbf{Avd}^{DSIDH}_{\mathcal{B}}.$$

Game CPCCA	
01 $a_1 \leftarrow Z/l_1^{e_1}Z$	$\underline{\mathcal{O}_{\text{i-dec}}}$
02 $A_1 = \big(E_{A_1}, \phi_{A_1}(P_2), \phi_{A_1}(Q_2)\big)$	01 $L = \{-,-\}$
03 $(state, X^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{i-dec}}}(A_1)$	02 $x_i \leftarrow Z/l_1^{e_1}Z$ sends to $\mathcal{A}$
04 $b \leftarrow \{0,1\}, K_1^* \leftarrow \{0,1\}^\lambda$	03 $X_i = \{E_{X_i}, \phi_{X_i}(P_1), \phi_{X_i}(Q_1)\}$
05 $(C^*, K_0^*) \leftarrow \mathsf{Encaps}(A_1, X^*)$	04 $L = L \cup \{X_i, x_i\}$
06 $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{i-dec}}}(state, C^*, K_b^*)$	05 On receiving (X', C') from $\mathcal{A}$
07 return $b' \stackrel{?}{=} b$	06 If $\exists X_j \in L \wedge X_j = X'$
	07 $K' = \mathsf{Decaps}((a_1, x_i), C')$
	08 else $\perp$

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

Proof of Sketch: By the classical FO transformation [11], the KEM_{dsidh} scheme is obviously classical one-way CCA secure under 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 challenge public key X^*; 2) it can query decryption oracle with the ciphertext under several public keys inspired by Okamoto [33]. To overcome this obstacle, when generating encapsulated key, we put 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 $\mathcal{A}$, and it will help $\mathcal{A}$ to determine m_2^*, the event that it queries H with m_1^* is bounded by solving DSIDH problem. Precisely, for public parameters $(E_0, P_1, Q_1, P_2, Q_2)$, given 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 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 $(Y^*, j(E_C) \oplus m_1^*, y_2^*)$ 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 solving DSIDH problem.

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 $h(j(E_X/\langle R_2^* + [y]S_2^*\rangle))$ is leaked. Precisely, for any PPT CPCCA adversary $\mathcal{A}$ who also gets $h(j(E_X/\langle R_2^* + [y]S_2^*\rangle))$, there exists algorithm $\mathcal{B}$ such that*

$$\mathbf{Adv}^{CPCCA}_{\mathcal{A},\mathsf{KEM}_{dsidh}} \leq \mathbf{Avd}^{1\text{-}OSIDH}_{\mathcal{B}}.$$

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

4.2 Three-pass AKE from 1-Oracle SIDH Assumption

Equipped with strong CPCCA secure KEM under 1-Oracle SIDH assumption as a core building block, we propose a 3-pass AKE 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 [12] that one party can play both the roles of initiator and responder in a multi-user setting. Therefore, we should distribute two pairs of static keys for the participant 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 key lies 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 $Z/l_1^{e_1}Z$. Then U_A revokes 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 $\{0,1\}^\lambda$, spares $G(r_2, b_2)$ 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 $y_2 = h(j(E_{YX})) \oplus m_2$. Then U_B forwards the identity messages of U_A and U_B with (Y, y_1, y_2) to U_A.

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 $\big(h(j(E_{A_1Y})),\ h(j(E_{XY}))\big)$ with the secret keys a_1 and x. Then U_A could know m_1' and m_2' by $h(j(E_{A_1Y})) \oplus y_1$ and $h(j(E_{XY})) \oplus y_2$. 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 $x_2 = h(j(E_{XY})) \oplus n_2$ to U_B.

The key $K_A = H(Y, n_1, n_2, X, x_1, x_2)$. 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 $h(j(E_{YX})) \oplus x_2$. 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 $sid = \big(U_A, U_B, ek_{A_1}, ek_{B_2}, X, x_1, x_2, Y, y_1, y_2\big)$ 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 ephemeral secret key r_1, decapsulated key K_B' and encapsulated key K_A. The session state of sid owned by U_B consists of ephemeral secrete key r_2, encapsulated key K_B and decapsulated key K_A'.

Correctness. To show that both U_A and U_B can agree on the same session key, we present an in-depth 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.

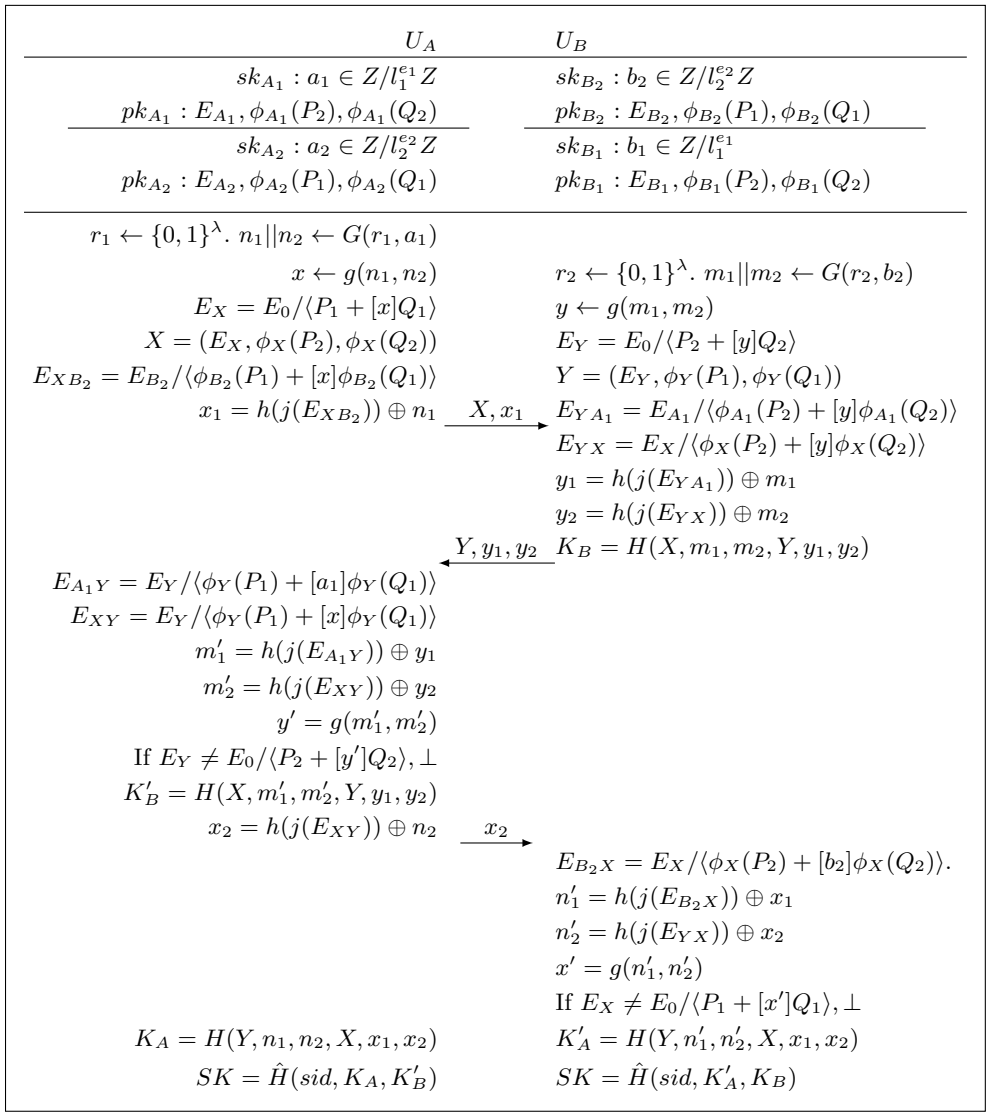

Fig. 6. A 3-Pass AKE $\mathsf{AKE}_{\mathsf{SIDH\text{-}3}}$ Based on 1-Oracle SIDH. Here *sid* is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x_1, Y, y_1, y_2, x_2\big)$.

$$
\begin{aligned}
E_{XB_2} &= E_{B_2}/\langle \phi_{B_2}(P_1) + [a_1']\phi_{B_2}(Q_1)\rangle = (E_0/\langle R_{B_2}\rangle)/(\langle R_X, R_{B_2}\rangle/\langle R_{B_2}\rangle) \\
&= E_0/\langle R_X, R_{B_2}\rangle \\
E_{B_2X} &= E_X/\langle \phi_X(P_2) + [b_2]\phi_X(Q_2)\rangle = (E_0/\langle R_X\rangle)/(\langle R_X, R_{B_2}\rangle/\langle R_X\rangle) \\
&= E_0/\langle R_X, R_{B_2}\rangle
\end{aligned}
$$

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* $\mathsf{AKE}_{\mathit{SIDH\text{-}3}}$ *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_G *times of* G *oracle queries,* q_H *(*q_g*) times of* H *(*g*) oracle queries and* q *times of CK*$^+$ *queries, there exists* $\mathcal{S}$ *s.t.*

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

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,

$$
\begin{aligned}
\Pr[\mathsf{Succ}] &= \Pr[\mathsf{Succ} \wedge \overline{\mathsf{AskH}}] + \Pr[\mathsf{Succ} \wedge \mathsf{AskH}] &(1)\\
&\leq \Pr[\mathsf{Succ} \wedge \overline{\mathsf{AskH}}] + \Pr[\mathsf{AskH}], &(2)
\end{aligned}
$$

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 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 CK$^+$ model. Followed by Lemma 2, we achieve the security of AKE in 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 event* AskH *defined above is negligible. Precisely,*

$$\Pr[\mathit{AskH}] \leq N^2 lq \cdot \mathbf{Avd}^{\mathit{1\text{-}OSIDH}}_{\mathcal{S}}$$

We give the sketch of proof here, for formal proof please refer to Appendix A. 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 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 $h(j(E_X^*/\langle R_2^* + [y]S_2^* \rangle))$. 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 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 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 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 $h(j(E_X^*/\langle R_2^* + [y]S_2^* \rangle))$ 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. □

5 A Two-Pass AKE from Supersingular Isogeny

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 soundness of AKE, we propose a 2-pass AKE with a little more communication (which is still competitive with existing schemes) based on DSIDH assumption.

Intuition of 2-Pass AKE. The reason 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 avoid the 1-Oracle SIDH assumption. Besides, it is also necessary to delete x_2 and set (X, x_1) as the ciphertext under 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.

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.

Step 1. U_A randomly selects two elements $r_1 \in \{0,1\}^\lambda$ and $r_{X_0} \in Z/l_1^{e_1}Z$. 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 $G(r_2, b_2)$ to two bit strings $m_1||m_2$ of 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 $E_0/\langle P_1 + [x']Q_1\rangle$. If it is not equal to E_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 $E_0/\langle P_2 + [y']Q_2\rangle$. If it is not equal to E_Y, 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 ephemeral secret key r_1, r_{X_0}, decapsulated key K_B' and encapsulated key K_A; The session state of sid owned by U_B consists of ephemeral secrete key r_2 and encapsulated key K_B, but does not include 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 groups of curves are not exactly the same. We only have

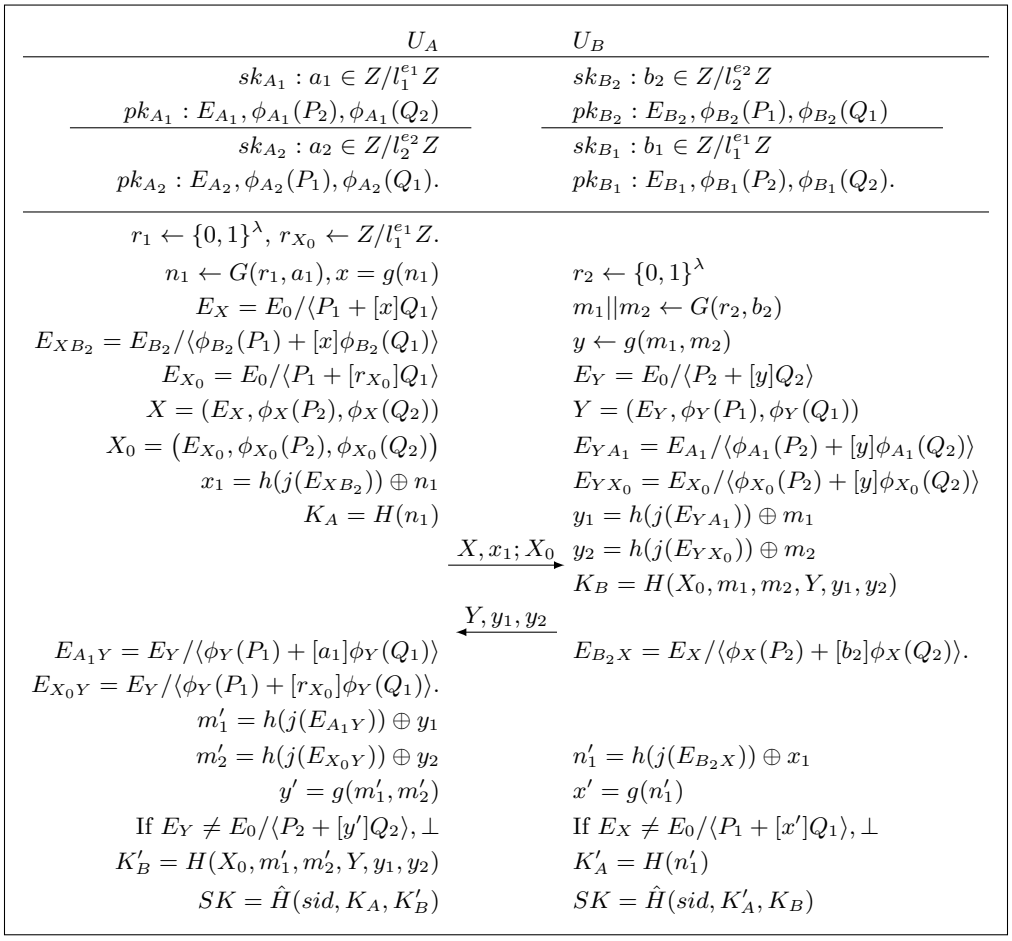

Fig. 7. A Compact 2-pass AKE Based on SIDH. Here *sid* is $\big(U_A, U_B, pk_{A_1}, pk_{B_2}, X, x, X_0, Y, y_1, y_2\big)$.

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_G *times of* G *oracle queries,* $q_H\,(q_g)$ *times of* $H\,(g)$ *oracle queries and* q *times of* CK^+ *queries, there exists* $\mathcal{S}$ *s.t.*

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

Proof of 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 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 assumption, DSIDH assumption.

6 Parameters, Implementation and Comparison

If we demand λ bits of quantum security and adopt the parameters chosen in [7] which are considered to be the most efficient choices, then the prime is of bit-length 6λ. Each field element needs 12λ since the curve is defined over $\mathbb{F}_{p^2}$. Then 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 7 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 it requires one more pass to interact with parties. However, the cost of interaction could be neglected, comparing with the cost of one such isogeny in our scheme. 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 [5], 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 [17]. 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.

In terms of implementation, the hash functions used in the authenticated key exchange are all instantiated with the SHA-3 function cSHAKE256 [26]. 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 cuts the bandwidth almost to half of both FSXY [10] and BCNP-Lon [3, 28].

In Table 4, we present the performance of our protocols comparing with the FSXY scheme [10] and the BCNP-Lon scheme [3, 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 with 1.2 times faster than FSXY and 1.4 times faster than BCNP-Lon.

Scheme	$A \to B$	$B \to A$	$A \to B$	**total(byte)**
FSXY [10]	1160	1160	-	2320
BCNP-Lon [3, 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 [17], taking into account the efficiency. "-" stands for no messages to be transmitted. Only our first scheme of this paper among the four schemes listed 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 [10]	6,238	14,779	10,124		31,141
BCNP-Lon [3, 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.

7 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.

We have proved the security in random oracle, but not considered the standard model. It is non-trial 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 Cryptographers Track at the RSA Conference (pp. 143-158). Springer, Berlin, Heidelberg (2001).
2. Azarderakhsh, R., Jao, D., Kalach, K., Koziel, B., Leonardi, C. : Key compression for isogeny-based cryptosystems. In Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography (pp. 1-10). ACM (2016).
3. Boyd, C., Cliff, Y., Nieto, J. M. G., Paterson, K. G.: One-round key exchange in the standard model. International Journal of Applied Cryptography, 1(3), 181-199 (2009).
4. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In Annual international cryptology conference (pp. 232-249). Springer, Berlin, Heidelberg (1993).
5. Costello, C., Jao, D., Longa, P., Naehrig, M., Renes, J., Urbanik, D.: Efficient compression of SIDH public keys. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 679-706). Springer, Cham (2017).
6. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In TCC 2010, pp. 453-474.
7. Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for supersingular isogeny Diffie-Hellman. In CRYPTO 2016, pp. 572-601.
8. Faz-Hernndez, A., Lpez, J., Ochoa-Jimnez, E., Rodrguez-Henrquez, F.: A faster software implementation of the supersingular isogeny diffie-hellman key exchange protocol. IEEE Transactions on Computers (2017).
9. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In PKC 2012, pp. 467-484.
10. 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.
11. Fujisaki, E., Okamoto, T. : Secure integration of asymmetric and symmetric encryption schemes. In CRYPTO 1999, pp. 537-554.
12. Galbraith, S. D.: Authenticated key exchange for SIDH. IACR Cryptology ePrint Archive 2018: 266 (2018).
13. Galbraith, S. D., Vercauteren, F.: Computational problems in supersingular elliptic curve isogenies. IACR Cryptology ePrint Archive 2017/774.
14. Galbraith, S. D., Petit, C., Silva, J.: Signature schemes based on supersingular isogeny problems. Cryptology ePrint Archive, Report 2016/1154 (2016).
15. Galbraith, S. D., Petit, C., Shani, B., Ti, Y. B.: On the security of supersingular isogeny cryptosystems. In ASIACRYPT 2016, pp. 63-91.
16. Guilhem, C. D. S., Smart, N. P., Warinschi, B.: Generic Forward-Secure Key Agreement Without Signatures. In International Conference on Information Security (pp. 114-133). Springer, Cham (2017).
17. Jao, D., Azarderakhsh, R., Campagna, M., et al: Supersingular Isogeny Key Encapsulation. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions.
18. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In International Workshop on Post-Quantum Cryptography, 19-34. Springer Berlin Heidelberg (2011).
19. 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).

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 International Conference on Selected Areas in Cryptography (pp. 64-81). Springer, Cham (2017).
23. Koziel, B., Azarderakhsh, R., Mozaffari-Kermani, M.: A High-Performance and Scalable Hardware Architecture for Isogeny-Based Cryptography. IEEE Transactions on Computers (2018).
24. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In CRYPTO 2005, pp. 546-566.
25. 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).
26. Kelsey, J.: SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash. NIST Special Publication, 800, 185 (2016).
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 (2018).
29. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In ProvSec 2007, pp. 1-16.
30. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient Protocol for Authenticated Key Agreement. Designs, Codes and Cryptography, 28(2), 119-134 (2003).
31. Menezes, A., Qu, M., Vanstone, S.: Some new key agreement protocols providing mutual implicit authentication. Selected Areas in Cryptography (1995).
32. Matsumoto, T., Takashima, Y., Imai, H.: On seeking smart public-key-distribution systems. IEICE TRANSACTIONS (1976-1990), 69(2), 99-106 (1986).
33. Okamoto, T.: Authenticated Key Exchange and Key Encapsulation Without Random Oracles. Cryptology ePrint Archive, Report 2007/473 (2007).
34. Sun, X., Tian, H., Wang, Y.: Toward quantum-resistant strong designated verifier signature from isogenies. In Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on. IEEE, 292-296 (2012).
35. Urbanik, D., Jao, D.: SoK: The problem landscape of SIDH. IACR Cryptology ePrint Archive 2018: 336 (2018).
36. Yoo, Y., Azarderakhsh, R., Jalali, A., Jao, D., Soukharev, V.: A post-quantum digital signature scheme based on supersingular isogenies. In International Conference on Financial Cryptography and Data Security (pp. 163-181). Springer, Cham (2017).
37. Yao, A. C. C., Zhao, Y. : OAKE: A new family of implicitly authenticated Diffie-Hellman protocols. In CCS 2013, pp. 1113-1128.
38. Zanon, G. H., Simplicio, M. A., Pereira, G. C., Doliskani, J., Barreto, P. S.: Faster isogeny-based compressed key agreement. In International Conference on Post-Quantum Cryptography (pp. 248-268). Springer, Cham (2018).
39. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644654 (1976)
40. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, EUROCRYPT 2000, volume 1807 of LNCS, pages 139155. Springer, Heidelberg, May 2000.

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 $1/N^2l$. $\mathcal{S}$ 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 $h(j(E_X^*/\langle R_2^* + [y]S_2^* \rangle))$. 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 $h(j(E_X^*/\langle R_2^* + [y]S_2^* \rangle))$ 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 $(\Pi, I, U_A, U_B, X, x_1, Y, y_1, y_2, x_2)$ 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 $h(j(E_X^*/\langle R_2^* + [y]S_2^*\rangle))$ 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 $h(j(E_X/\langle R_2 + [y]S_2\rangle)$. 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 $\Pr[\mathsf{AskH} \wedge E_3] \leq N^2 l \cdot \mathbf{Adv}^{\mathrm{CPCCA}}_{\mathsf{KEM}_{dsidh}}(\mathcal{S}) \leq N^2 l \cdot \mathbf{Avd}^{\text{1-OSIDH}}_{\mathcal{B}}$.

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 $\mathbf{Avd}_{\mathcal{B}}^{\text{1-OSIDH}}$.

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.