An Efficient and Generic Construction for Signal's Handshake (X3DH): Post-Quantum, State Leakage Secure, and Deniable

Keitaro Hashimoto[1,2], Shuichi Katsumata[2], Kris Kwiatkowski[3], Thomas Prest[3]

[1]Tokyo Institute of Technology, Japan
hashimoto.k.au@m.titech.ac.jp
[2]AIST, Japan
shuichi.katsumata@aist.go.jp
[3]PQShield, U.K.
{kris.kwiatkowski,thomas.prest}@pqshield.com

May 12, 2021

Abstract

The Signal protocol is a secure instant messaging protocol that underlies the security of numerous applications such as WhatsApp, Skype, Facebook Messenger among many others. The Signal protocol consists of two sub-protocols known as the X3DH protocol and the double ratchet protocol, where the latter has recently gained much attention. For instance, Alwen, Coretti, and Dodis (Eurocrypt'19) provided a concrete security model along with a generic construction based on simple building blocks that are instantiable from versatile assumptions, including post-quantum ones. In contrast, as far as we are aware, works focusing on the X3DH protocol seem limited.

In this work, we cast the X3DH protocol as a specific type of authenticated key exchange (AKE) protocol, which we call a *Signal-conforming AKE* protocol, and formally define its security model based on the vast prior work on AKE protocols. We then provide the first efficient generic construction of a Signal-conforming AKE protocol based on standard cryptographic primitives such as key encapsulation mechanisms (KEM) and signature schemes. Specifically, this results in the first post-quantum secure replacement of the X3DH protocol on well-established assumptions. Similar to the X3DH protocol, our Signal-conforming AKE protocol offers a strong (or stronger) flavor of security, where the exchanged key remains secure even when all the non-trivial combinations of the long-term secrets and session-specific secrets are compromised. Moreover, our protocol has a weak flavor of deniability and we further show how to strengthen it using ring signatures. Finally, we provide a full-fledged, generic C implementation of our (weakly deniable) protocol. We instantiate it with several Round 3 candidates (finalists and alternates) to the NIST post-quantum standardization process and compare the resulting bandwidth and computation performances. Our implementation is publicly available.

1 Introduction

Secure instant messaging (SIM) ensures privacy and security by making sure that only the person you are sending the message to can read the message, a.k.a. end-to-end encryption. With the ever-growing awareness against mass-surveillance of communications, people have become more privacy-aware and the demand for SIM has been steadily increasing. While there have been a range of SIM protocols, the Signal protocol [SIG] is widely regarded as the gold standard. Not only is it used by the Signal app[1], the Signal protocol is also

[1] The name Signal is used to point to the app *and* the protocol.

used by WhatsApp, Skype, Facebook Messenger among many others, where the number of active users is well over 2 billions. One of the reasons for such popularity is due to the simplicity and the strong security properties it provides, such as forward secrecy and post-compromise secrecy, while simultaneously allowing for the same user experience as any (non-cryptographically secure) instant messaging app.

The Signal protocol consists of two sub-protocols: the X3DH protocol [MP16b] and the double ratchet protocol [MP16a]. The former protocol can be viewed as a type of key exchange protocol allowing two parties to exchange a secure initial/session key. The latter protocol is executed after the X3DH protocol and it allows two parties to perform a secure back-and-forth message delivery. Below, we briefly recall the current affair of these two protocols.

The Double Ratchet Protocol. The first attempt at a full security analysis of the Signal protocol was made by Cohn-Gordon et al. [CGCD$^+$17, CGCD$^+$20]. They considered the Signal protocol as one large protocol and analyzed the security guarantees in its entirety. Since the double ratchet protocol was understood to be the root of the complexity, many subsequent works aimed at further abstracting and formalizing (and in some cases enhancing) the security of the double ratchet protocol by viewing it as a stand-alone protocol [BSJ$^+$17, PR18, ACD19, DV19, JMM19a, JMM19b]. Under these works, our understanding of the double ratchet protocol has much matured. Notably, Alwen et al. [ACD19] fully abstracted the complex Diffie-Hellman based double ratchet protocol used by Signal and provided a concrete security model along with a generic construction based on simple building blocks. Since these blocks are instantiable from versatile assumptions, including post-quantum ones, their work resulted in the first *post-quantum secure* double ratchet protocol. Here, we elucidate that all the aforementioned works analyze the double ratchet protocol as a stand-alone primitive, and hence, it is assumed that any two parties can securely share an session key, for instance, by executing a "secure" X3DH protocol.

The X3DH Protocol. In contrast, other than the white paper offered by Signal [MP16b] and those indirectly considered by Cohn-Gordon et al. [CGCD$^+$17, CGCD$^+$20], works focusing on the X3DH protocol seems to be limited. As far as we are aware, there is one recent work that studies the formalization [BFG$^+$20] and a few papers that study one of the appealing security properties, known as (off-line) *deniability*, claimed by the X3DH protocol [VGIK20, UG15, UG18].

Brendel et al. [BFG$^+$20] abstract the X3DH protocol and provides the first generic construction based on a new primitive they call a *split key encapsulation mechanism* (KEM). However, so far, instantiations of split KEMs with strong security guarantees required for the X3DH protocol are limited to Diffie-Hellman style assumptions. In fact, the recent result of Guo et al. [GKRS20] implies that it would be difficult to construct them from one of the promising post-quantum candidates: lattice-based assumptions (and presumably coded-based assumptions). On the other hand, Vatandas et al. [VGIK20] study one of the security guarantees widely assumed for the X3DH protocol called (off-line) deniability [MP16b, Section 4.4] and showed that a strong knowledge-type assumption would be necessary to formally prove it. Unger and Goldberg [UG15, UG18] construct several protocols that can be used as a drop-in replacement of the X3DH protocol that achieves a strong flavor of (on-line) deniability from standard assumptions, albeit by making a noticeable sacrifice in the security against key-compromise attacks: a type of attack that exploits leaked secret information of a party. For instance, while the X3DH protocol is secure against key-compromise impersonation (KCI) attacks [BWJM97],[2] the protocols of Unger and Goldberg are no longer secure against such attacks.[3]

Motivation. In summary, although we have a rough understanding of what the X3DH protocol offers [MP16b, CGCD$^+$17, CGCD$^+$20], the current state of affairs is unsatisfactory for the following reasons, and making progress on these issues will be the focus of this work:

- It is difficult to formally understand the security guarantees offered by the X3DH protocol or to make a meaningful comparison among different protocols achieving the same functionality as the X3DH protocol without a clearly defined security model.

[2] Although [MP16b, Section 4.6] states that the X3DH protocol is susceptible to KCI attacks, this is only because they consider the scenario where the *session-specific* secret is compromised. If we consider the standard KCI attack scenario where the long-term secret is the only information being compromised [BWJM97], then the X3DH protocol is secure.

[3] Being vulnerable against KCI attacks seems to be intrinsic to on-line deniability [UG15, UG18, MP16b].

- The X3DH protocol is so far only instantiable from Diffie-Hellman style assumptions [BFG+20] and it is unclear whether such assumptions are inherent to the Signal protocol.
- Ideally, similarly to what Alwen et al. [ACD19] did for the double ratchet protocol, we would like to abstract the X3DH protocol and have a generic construction based on simple building blocks that can be instantiated from versatile assumptions, including but not limited to post-quantum ones.
- No matter how secure the double ratchet protocol is, we cannot completely secure the Signal protocol if the initial X3DH protocol is the weakest link in the chain (e.g., insecure against state-leakage and only offering security against classical adversaries).

1.1 Our Contribution

In this work, we cast the X3DH protocol (see Figure 1) as a specific type of authenticated key exchange (AKE) protocol, which we call a *Signal-conforming AKE* protocol, and define its security model based on the vast prior work on AKE protocols. We then provide an efficient generic construction of a Signal-conforming AKE protocol based on standard cryptographic primitives: an (IND-CCA secure) KEM, a signature scheme, and a pseudorandom function (PRF). Since all of these primitives can be based on well-established post-quantum assumptions, this results in the first post-quantum secure replacement of the X3DH protocol. Similarly to the X3DH protocol, our Signal-conforming AKE protocol offers a strong flavor of key-compromise security. Borrowing terminologies from AKE-related literature, our protocol is proven secure in the strong Canetti-Krawczyk (CK) type security models [CK01, Kra05, FSXY12, LLM07], where the exchanged session key remains secure even if all the non-trivial combinations of the long-term secrets and session-specific secrets of the parties are compromised. In fact, our protocol is more secure than the X3DH protocol since it is even secure against KCI-attacks where the parties' session-specific secrets are compromised (see Footnote 2). [4] We believe the level of security offered by our Signal-conforming AKE protocol aligns with the level of security guaranteed by the double ratchet protocol where (a specific notion of) security still holds even when such secrets are compromised. Moreover, while our Signal-conforming AKE already provides a weak form of deniability, we can strengthen its deniability by using a ring signature scheme or/and a non-interactive zero-knowledge proof of knowledge. Likewise to the X3DH protocol [VGIK20] although our construction seemingly offers (off-line) deniability, the formal proof relies on a strong knowledge-type assumption. However, relying on such assumptions seems unavoidable considering that all known deniable AKE protocols secure against key-compromise attacks, including the X3DH protocol, rely on them [DGK06, YZ10, VGIK20]. We consider deniability against semi-honest and malicious adversaries and note that we only achieve classical security against the latter type of adversary.

We implemented our (weakly deniable) Signal-conforming AKE protocol in C, building on the open source libraries PQClean and LibTomCrypt. Our implementation is fully generic and can thus be instantiated with a wide range of KEMs and signature schemes. The code is available at [Kwi20]. We instantiate it with several Round 3 candidates (finalists and alternates) to the NIST post-quantum standardization process, and compare the bandwidth and computation costs that result from these choices. Our protocol performs best with "balanced" schemes, for example most lattice-based schemes. The isogeny-based scheme SIKE offers good bandwidth performance, but entails a significant computation cost. Finally, schemes with large public keys (Classic McEliece, Rainbow, etc.) do not seem to be a good match for our protocol, since these keys are transferred at each run of the protocol.

1.2 Technical Overview

We now briefly recall the X3DH protocol and abstract its required properties by viewing it through the lens of AKE protocols. We then provide an overview of how to construct a Signal-conforming AKE protocol from standard assumptions.

[4] The X3DH can be made secure against leakge of session-specific secrets by using NAXOS trick [LLM07], but it requires additional computation. Because it affects efficiency, we do not consider AKE protocols using NAXOS trick (e.g., [FSXY12, KF14, YCL18]).

Recap on the X3DH Protocol. At a high level, the X3DH protocol allows for an asynchronous key exchange where two parties, say Alice and Bob, exchange a session key without having to be online at the same time. Even more, the party, say Bob, that wishes to send a secure message to Alice can do so without Alice even knowing Bob. For instance, imagine the scenario where you send a friend request and a message at the same time before being accepted as a friend. At first glance, it seems what we require is a non-interactive key exchange (NIKE) since Bob needs to exchange a key with Alice who is offline, while Alice does not yet know that Bob is trying to communicate with her. Unfortunately, solutions based on NIKEs are undesirable since they either provide weaker guarantees than standard (interactive) AKE or exhibit inefficient constructions [Ber06, CKS08, FHKP13, PS14].

The X3DH protocol circumvents this issue by considering an *untrusted server* (e.g., the Signal server) to sit in the middle between Alice and Bob to serve as a public bulletin board. That is, the parties can store and retrieve information from the server while the server is not assumed to act honestly. A simplified description of the X3DH protocol, which still satisfies our purpose, based on the classical Diffie-Hellman (DH) key exchange is provided in Figure 1.[5] As the first step, Alice sends her DH component $g^x \in \mathbb{G}$ to the server[6] and then possibly goes offline. We point out that Alice does *not* need to know who she will be communicating with at this point. Bob, who may ad-hocly decide to communicate with Alice, then fetches Alice's first message from the server and uploads its DH component g^y to the server. As in a typical DH key exchange, Bob computes the session key $\mathsf{k_B}$ using the long-term secret exponent $b \in \mathbb{Z}_p$ and session-specific secret exponent $y \in \mathbb{Z}_p$. Since Bob can compute the session key $\mathsf{k_B}$ while Alice is offline, he can begin executing the subsequent double ratchet protocol without waiting for Alice to come online. Whenever Alice comes online, she can fetch whatever message Bob sent from the server.

Alice: ($\mathsf{lpk_A} = g^a, \mathsf{lsk_A} = a$)		Server		Bob: ($\mathsf{lpk_B} = g^b, \mathsf{lsk_B} = b$)
				Fetch (Alice, g^x)
$x \leftarrow\$ \mathbb{Z}_p$				$y \leftarrow\$ \mathbb{Z}_p$
Store x		Store		$\mathsf{k_B} := \mathsf{KDF}((g^x)^b,$
Upload g^x to server	$\xrightarrow{g^x}$	(Alice, g^x)	$\xrightarrow{g^x}$	$(g^a)^y, (g^x)^y)$
`-- go offline --`				Upload g^y to server
				Erase y
`-- come online --`		Store		
Fetch ((Alice, Bob), g^y)	$\xleftarrow{g^y}$	((Alice, Bob),	$\xleftarrow{g^y}$	
$\mathsf{k_A} := \mathsf{KDF}((g^b)^x, (g^y)^a, (g^y)^x)$		g^y)		

Figure 1: Simplified description of the X3DH Protocol. Alice and Bob have the long-term key pairs $(\mathsf{lpk_A}, \mathsf{lsk_A})$ and $(\mathsf{lpk_B}, \mathsf{lsk_B})$, respectively. Alice and Bob agree on a session key $\mathsf{k_A} = \mathsf{k_B}$, where KDF denotes a key derivation function.

Casting the X3DH Protocol as an AKE Protocol. It is not difficult to see that the X3DH protocol can be cast as a specific type of AKE protocol. In particular, we can think of the server as an adversary that tries to mount a man-in-the-middle (MIM) attack in a standard AKE protocol. Viewing the server as a malicious adversary, rather than some semi-honest entity, has two benefits: the parties do not need to put trust in the server since the protocol is supposed to be secure even against a malicious server, while the server or the company providing the app is relieved from having to "prove" that it is behaving honestly. One distinguishing feature required by the X3DH protocol when viewed as an AKE protocol is that it needs to be a two-round protocol where the initiator message is generated *independently* from the receiver. That is, Alice

[5] We assume Alice and Bob know each other's long-term key. In practice, this can be enforced by "out-of-bound" authentications (see [MP16b, Section 4.1]).

[6] In the actual protocol, Alice also signs g^x sent to the server (i.e., *signed pre-keys*). We ignore this subtlety as it does not play a crucial role in the analysis of security. See Remark 4.3 for more detail. Also, we note that in practice, Bob may initiate the double ratchet protocol using $\mathsf{k_B}$ and send his message to Alice along with g^y to the server before Alice responds.

needs to be able to store her first message to the server without knowing who she will be communicating with. In this work, we define an AKE protocol with such functionality as a *Signal-conforming* AKE protocol.

Regarding the security model for a Signal-conforming AKE protocol, we base it on the vast prior works on AKE protocols. Specifically, we build on the recent formalization of [GJ18, CCG+19] that study the tightness of efficient AKE protocols (including a slight variant of the X3DH protocol) and strengthen the model to also incorporate *state leakage* compromise; a model where an adversary can obtain session-specific information called *session-state.* Since the double ratchet protocol considers a very strong form of state leakage security, we believe it would be the most rational design choice to discuss the X3DH protocol in a security model that captures such leakage as well. Informally, we consider our Signal-conforming AKE protocol in the Canetti-Krawczyk (CK) type security model [CK01, Kra05, FSXY12, LLM07], which is a strengthening of the Bellare-Rogaway security model [BR94] considered by [GJ18, CCG+19]. A detailed discussion and comparison between ours and the numerous other security models of AKE protocols are provided in Section 3.

Lack of Signal-Conforming AKE Protocol. The main feature of a Signal-conforming AKE protocol is that the initiator's message does *not* depend on the receiver. Although this seems like a very natural feature considering DH-type AKE protocols, it turns out that they are quite unique (see Brendel et al. [BFG+20] for some discussion). For instance, as far as we are aware, the only other assumption that allows for a clean analog of the X3DH protocol is based on the *gap* CSIDH assumption recently introduced by De Kock et al. [dKGV20] and Kawashima et al. [KTAT20]. Considering the community is still in the process of assessing the concrete parameter selection for *standard* CSIDH [BS20, Pei20], it would be desirable to base the X3DH protocol on more well-established and versatile assumptions. On the other hand, when we turn our eyes to known generic construction of AKE protocols [FSXY12, FSXY13, XLL+18, HKSU20, XAY+20] that can be instantiated from versatile assumptions, including post-quantum ones, we observe that none of them is Signal-conforming. That is, they are all either non-2-round or the initiator's message depends on the public key of the receiver.

Our Construction. To this end, in this work, we provide a new practical generic construction of a Signal-conforming AKE protocol from an ($\mathsf{IND\text{-}CCA}$ secure) KEM and a signature scheme. We believe this may be of independent interest in other scenarios where we require an AKE protocol that has a flavor of "receiver obliviousness."[7] The construction is simple: The construction is simple: Let us assume Alice and Bob's long-term key consist of KEM key pairs $(\mathsf{ek_A}, \mathsf{dk_A})$ and $(\mathsf{ek_B}, \mathsf{dk_B})$ and signature key pairs $(\mathsf{vk_A}, \mathsf{sk_A})$ and $(\mathsf{vk_B}, \mathsf{sk_B})$, respectively. The Signal-conforming AKE protocol then starts by Alice (i.e., the initiator) generating a session-specific KEM key $(\mathsf{ek}_T, \mathsf{dk}_T)$ and sending ek_T to Bob (i.e., the receiver).[8] Here, observe that Alice's message does not depend on who she will be communicating with. Bob then constructs two ciphertexts: one using Alice's long-term key $(\mathsf{K_A}, \mathsf{C_A}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek_A})$ and another using the session-specific key $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_T)$. It then signs these ciphertext $\mathsf{M} := (\mathsf{C_A}, \mathsf{C}_T)$ as $\sigma_\mathsf{B} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk_B}, \mathsf{M})$, where we include other session-specific components in M in the actual construction. Since sending σ_B in the clear may serve as public evidence that Bob communicated with Alice, Bob will hide this. To this end, he derives two keys, a session key $\mathsf{k_{AKE}}$ and a one-time pad key $\mathsf{k_{OTP}}$, by running a key derivation function on input the random KEM keys $(\mathsf{K_A}, \mathsf{K}_T)$. Bob then sends $(\mathsf{C_A}, \mathsf{C}_T, \mathsf{c} := \sigma_\mathsf{B} \oplus \mathsf{k_{OTP}})$ to Alice and sets the session key as $\mathsf{k_{AKE}}$. Once Alice receives the message from Bob, she decrypts the ciphertexts $(\mathsf{C_A}, \mathsf{C}_T)$, derives the two keys $(\mathsf{k_{AKE}}, \mathsf{k_{OPT}})$, and checks if $\sigma := \mathsf{c} \oplus \mathsf{k_{OTP}}$ is a valid signature of Bob's. If so, she sets the session key as $\mathsf{k_{AKE}}$. At a high level, Alice (explicitly) authenticates Bob through verifying Bob's signature and Bob (implicitly) authenticates Alice since Alice is the only party that can decrypt *both* ciphertexts $(\mathsf{C_A}, \mathsf{C}_T)$. We turn this intuition into a formal proof and show that our scheme satisfies a strong flavor of security where the shared session key remains pseudorandom even to an adversary that can obtain any non-trivial combinations of the long-term private keys (i.e., $\mathsf{dk_A}, \mathsf{dk_B}, \mathsf{sk_A}, \mathsf{sk_B}$) and session-specific secret keys (i.e., dk_T). Notably, our protocol satisfies a stronger notion of security compared to the X3DH protocol since it prevents an adversary to impersonate Alice even if her session-specific secret key is compromised [MP16b, Section 4.6].

[7] This property has also been called as *post-specified peers* [CK02] in the context of Internet Key Exchange (IKE) protocols.

[8] As we briefly commented in Footnote 6, Alice can sign her message ek_T as in the X3DH protocol. This will only make our protocol more secure. See Remark 4.3 for more detail.

Finally, our Signal-conforming AKE protocol already satisfies a limited form of deniability where the publicly exchanged messages do not directly leak the participant of the protocol. However, if Alice at a later point gets compromised or turns malicious, she can publicize the signature σ_{B} sent from Bob to cryptographically prove that Bob was communicating with Alice. This is in contrast to the X3DH protocol that does not allow such a deniability attack. We, therefore, show that we can protect Bob from such attacks by replacing the signature scheme with a *ring* signature scheme. In particular, Alice now further sends a session-specific ring signature verification key vk_T, and Bob signs to the ring $\{\mathsf{vk}_T, \mathsf{vk}_{\mathsf{B}}\}$. Effectively, when Alice outputs a signature from Bob $\sigma_{\mathsf{B},T}$, she cannot fully convince a third party whether it originates from Bob since she could have signed $\sigma_{\mathsf{B},T}$ using her signing key sk_T corresponding to vk_T. Although the intuition is clear, it turns out that turning this into a formal proof is quite difficult. Similar to all previous works on AKE protocols satisfying a strong flavor of key-compromise security [DGK06, YZ10] (including the X3DH protocol [VGIK20]), the proof of deniability must rely on a strong knowledge-type assumption. We leave it as future work to investigate the deniability of our Signal-conforming AKE protocols from more standard assumptions.

2 Preliminaries

In this section, we review the basic notations and definitions of cryptographic primitives used in this paper.

Notation. The operator $\oplus$ denotes bit-wise "XOR", and $\|$ denotes string concatenation. For $n \in \mathbb{N}$, we write $[n]$ to denote the set $[n] := \{1, \ldots, n\}$. For $j \in [n]$, we write $[n \backslash j]$ to denote the set $[n \backslash j] := \{1, \ldots, n\} \setminus \{j\}$. We denote by $x \leftarrow\!\!\$\, S$ the sampling of an element x uniformly at random from a finite set S. PPT (resp. QPT) stands for probabilistic (resp. quantum) polynomial time.

Cryptographic Primitives.

2.1 Key Encapsulation Mechanisms

Definition 2.1 (KEM Schemes). *A key encapsulation mechanism (KEM) scheme with session key space $\mathcal{KS}$ consists of the following four PPT algorithms $\Pi_{\mathsf{KEM}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$:*

$\mathsf{Setup}(1^\kappa) \to \mathsf{pp}$**:** *The setup algorithm takes the security parameter 1^κ as input and outputs a public parameter pp. In the following, we assume pp is provided to all the algorithms and may omit it for simplicity.*

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{ek}, \mathsf{dk})$**:** *The key generation algorithm takes a public parameter pp as input and outputs a pair of keys $(\mathsf{ek}, \mathsf{dk})$.*

$\mathsf{Encap}(\mathsf{ek}) \to (\mathsf{K}, \mathsf{C})$**:** *The encapsulation algorithm takes an encapsulation key ek as input and outputs a session key $\mathsf{K} \in \mathcal{KS}$ and a ciphertext C.*

$\mathsf{Decap}(\mathsf{dk}, \mathsf{C}) \to \mathsf{K}$**:** *The decapsulation algorithm takes a decapsulation key dk and a ciphertext C as input and outputs a session key $\mathsf{K} \in \mathcal{KS}$.*

Definition 2.2 ($(1-\delta)$-Correctness). *We say a KEM scheme Π_{KEM} is $(1-\delta)$-correct if for all $\kappa \in \mathbb{N}$ and $\mathsf{pp} \in \mathsf{Setup}(1^\kappa)$,*

$$(1-\delta) \leq \Pr\left[\mathsf{Decap}(\mathsf{dk}, \mathsf{C}) = \mathsf{K} : \begin{array}{c} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); \\ (\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{Encap}(\mathsf{ek}) \end{array}\right].$$

Definition 2.3 (IND-CPA and IND-CCA Security). *Let κ be a security parameter, $\Pi_{\mathsf{KEM}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ be a KEM scheme and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, we define the* *advantage of $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}ATK}}(\mathcal{A}) := \left| \Pr\left[b = b' : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa); \\ (\mathsf{ek}^*, \mathsf{dk}^*) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); \\ \mathsf{state} \leftarrow \mathcal{A}_1^{\mathsf{O}_{\mathsf{ATK}}}(\mathsf{pp}, \mathsf{ek}^*); \\ b \leftarrow\!\!\$ \{0,1\}; \\ (\mathsf{K}_0^*, \mathsf{C}_0^*) \leftarrow \mathsf{Encap}(\mathsf{ek}^*); \\ \mathsf{K}_1^* \leftarrow\!\!\$ \mathcal{KS}; \\ b' \leftarrow \mathcal{A}_2^{\mathsf{O}_{\mathsf{ATK}}}(\mathsf{pp}, \mathsf{ek}^*, (\mathsf{K}_b^*, \mathsf{C}_0^*), \mathsf{state}) \end{array} \right] - \frac{1}{2} \right|,$$

where

$$\mathsf{O}_{\mathsf{ATK}} = \begin{cases} \perp & \mathsf{ATK} = \mathsf{CPA} \\ \mathsf{O}_{\mathsf{Decap}}(\mathsf{dk}^*, \cdot) & \mathsf{ATK} = \mathsf{CCA} \end{cases}.$$

When $\mathsf{ATK} = \mathsf{CCA}$*,* $\mathcal{A}_2$ *is not allowed to make an oracle query containing the challenge ciphertext* C_0^**. We say* Π_{KEM} *is* $\mathsf{IND\text{-}ATK}$ *secure for security parameter* κ *if the advantage* $\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}ATK}}(\mathcal{A})$ *is negligible for any* QPT *adversary* $\mathcal{A}$*.*

Definition 2.4 (Min-Entropy of KEM Encapsulation Key). *We say a KEM scheme* Π_{KEM} *has* ν*-high encapsulation key min-entropy if for all* $\kappa \in \mathbb{N}$ *and* $\mathsf{pp} \in \mathsf{Setup}(1^\kappa)$*,*

$$\nu \leq -\log_2 \left(\max_{\mathsf{ek}^*} \Pr\left[\mathsf{ek} = \mathsf{ek}^* : (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})\right] \right).$$

Definition 2.5 (Min-Entropy of KEM Ciphertext). *We say a KEM scheme* Π_{KEM} *has* χ*-high* ciphertext *min-entropy if for all* $\kappa \in \mathbb{N}$ *and* $\mathsf{pp} \in \mathsf{Setup}(1^\kappa)$*,*

$$\chi \leq -\log_2 \left(\mathbb{E}\left[\max_{\mathsf{C}^*} \Pr\left[\mathsf{C} = \mathsf{C}^* : (\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{Encap}(\mathsf{ek})\right] \right] \right),$$

where the expectation is taken over the randomness used to sample $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*.*

2.2 Digital Signatures

Definition 2.6 (Signature Schemes). *A signature scheme with message space* $\mathcal{M}$ *consists of the following four PPT algorithms* $\Pi_{\mathsf{SIG}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

$\mathsf{Setup}(1^\kappa) \rightarrow \mathsf{pp}$**:** *The setup algorithm takes a security parameter* 1^κ *as input and outputs a public parameter* pp*. In the following, we assume* pp *is provided to all the algorithms and may omit it for simplicity.*

$\mathsf{KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{vk}, \mathsf{sk})$**:** *The key generation algorithm takes a public parameter* pp *as input and outputs a pair of keys* $(\mathsf{vk}, \mathsf{sk})$*.*

$\mathsf{Sign}(\mathsf{sk}, \mathsf{M}) \rightarrow \sigma$**:** *The signing algorithm takes a signing key* sk *and a message* $\mathsf{M} \in \mathcal{M}$ *as input and outputs a signature* σ*.*

$\mathsf{Verify}(\mathsf{vk}, \mathsf{M}, \sigma) \rightarrow 1/0$**:** *The verification algorithm takes a verification key* vk*, a message* M *and a signature* σ *as input and outputs* 1 *or* 0*.*

Definition 2.7 ($(1-\delta)$-Correctness). *We say a signature scheme* Π_{SIG} *is* $(1-\delta)$*-correct if for all* $\kappa \in \mathbb{N}$*, all messages* $\mathsf{M} \in \mathcal{M}$ *and all* $\mathsf{pp} \in \mathsf{Setup}(1^\kappa)$*,*

$$(1-\delta) \leq \Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{M}, \sigma) = 1 : (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{M})\right].$$

Definition 2.8 (EUF-CMA Security). *Let κ be a security parameter, $\Pi_{\mathsf{SIG}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme and $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{A}) := \Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{vk}^*, \mathsf{M}^*, \sigma^*) = 1 \\ \wedge \mathsf{M}^* \notin \mathcal{M}^* \end{array} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa); \\ (\mathsf{vk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); \\ (\mathsf{M}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{Sign}}(\mathsf{sk}^*, \cdot)}(\mathsf{pp}, \mathsf{vk}^*) \end{array}\right]$$

where $\mathsf{O}_{\mathsf{Sign}}$ is the signing oracle and $\mathcal{M}^*$ is the set of messages that $\mathcal{A}$ submitted to the signing oracle. We say Π_{SIG} is EUF-CMA *secure for security parameter κ if the advantage $\mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{A})$ is negligible for any* QPT *adversary $\mathcal{A}$.*

2.3 Pseudo-Random Functions

Let $\mathsf{F} : \mathcal{FK} \times \mathcal{D} \to \mathcal{R}$ be a function family with key space $\mathcal{FK}$, domain $\mathcal{D}$ and finite range $\mathcal{R}$. We define a pseudo-random function as follows. Below, we note that the adversary $\mathcal{A}$ is only allowed to make classical queries to the oracles.

Definition 2.9 (Pseudo-Random Function Family). *Let $\mathcal{A}$ be an adversary that is given oracle access to either $\mathsf{F}_{\mathsf{K}}(\cdot) := \mathsf{F}(\mathsf{K}, \cdot)$ for $\mathsf{K} \leftarrow\$ \mathcal{FK}$ or a truly random function $\mathsf{RF} : \mathcal{D} \to \mathcal{R}$. We define the advantage of $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{F}}^{\mathsf{PRF}}(\mathcal{A}) := \left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{F}_{\mathsf{K}}(\cdot)}(1^\kappa)\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{RF}(\cdot)}(1^\kappa)\right]\right|.$$

We say F is a pseudo-random function (PRF) family if $\mathsf{Adv}_{\mathsf{F}}^{\mathsf{PRF}}(\mathcal{A})$ is negligible for any QPT *adversary $\mathcal{A}$.*

2.4 Strong Randomness Extractors

The statistical distance between random variables X, Y over a finite domain S is defined by

$$\mathsf{SD}(X, Y) := \frac{1}{2}\sum_{s \in S} \left|\Pr[X = s] - \Pr[Y = s]\right|.$$

Definition 2.10 (Strong Randomness Extractors). *Let $\mathsf{Ext} : \mathcal{S} \times \mathcal{D} \to \mathcal{R}$ be a family of efficiently computable functions with set $\mathcal{S}$, domain $\mathcal{D}$ and range $\mathcal{R}$, all with finite size. A function family Ext is a strong $(\lambda, \varepsilon_{\mathsf{Ext}})$-extractor if for any random variable X over $\mathcal{D}$ with $\Pr[X = x] \leq 2^{-\lambda}$ (i.e., X has min-entropy at least λ), if s and R are chosen uniformly at random from $\mathcal{S}$ and $\mathcal{R}$, respectively, the two distributions $(s, \mathsf{Ext}_s(X))$ and (s, R) are within statistical distance $\varepsilon_{\mathsf{Ext}}$, that is*

$$\mathsf{SD}((s, \mathsf{Ext}_s(X)), (s, R)) \leq \varepsilon_{\mathsf{Ext}}.$$

3 Security Model for Signal-Conforming AKE Protocols

In this section, we define a security model for a *Signal-conforming* authenticated key exchange (AKE) protocol; AKE protocols that can be used as a drop-in replacement of the X3DH protocol. We first provide in Sections 3.1 to 3.3 a game-based security model building on the recent formalization of [GJ18, CCG+19] targeting general AKE protocols. We then discuss in Section 3.4 the modifications needed to make it Signal-conforming. A detailed comparison and discussion between ours and other various security models for AKE protocols are provided in Section 3.5.

3.1 Execution Environment

We consider a system of μ parties $P_1, \ldots, P_\mu$. Each party P_i is represented by a set of ℓ oracles $\{\pi_i^1, \ldots, \pi_i^\ell\}$, where each oracle corresponds to a single execution of a protocol, and $\ell \in \mathbb{N}$ is the maximum number of protocol sessions per party. Each oracle is equipped with fixed randomness but is otherwise deterministic. Each oracle π_i^s has access to the long-term key pair $(\mathsf{lpk}_i, \mathsf{lsk}_i)$ of P_i and the public keys of all other parties, and maintains a list of the following local variables:

- rand_i^s is the randomness hard-wired to π_i^s;
- sid_i^s ("session identifier") stores the identity of the session as specified by the protocol;
- Pid_i^s ("peer id") stores the identity of the intended communication partner;
- $\Psi_i^s \in \{\bot, \texttt{accept}, \texttt{reject}\}$ indicates whether oracle π_i^s has successfully completed the protocol execution and "accepted" the resulting key;
- k_i^s stores the session key computed by π_i^s;
- state_i^s holds the (secret) session-state values and intermediary results required by the session;
- $\mathsf{role}_i^s \in \{\bot, \texttt{init}, \texttt{resp}\}$ indicates π_i^s's role during the protocol execution.

For each oracle π_i^s, these variables, except the randomness, are initialized to $\bot$. An AKE protocol is executed interactively between two oracles. An oracle that first sends a message is called an *initiator* ($\mathsf{role} = \texttt{init}$) and a party that first receives a message is called a *responder* ($\mathsf{role} = \texttt{resp}$). The computed session key is assigned to the variable k_i^s if and only if π_i^s reaches the $\texttt{accept}$ state, that is, $\mathsf{k}_i^s \neq \bot \iff \Psi_i^s = \texttt{accept}$.

Partnering. To exclude trivial attacks in the security model, we need to define a notion of "partnering" of two oracles. Intuitively, this dictates which oracles can be corrupted without trivializing the security game. We define the notion of partnering via session-identifiers following the work of [CK01, dFW20]. Discussions on other possible choices of the definition for partnering is provide in Section 3.5.

Definition 3.1 (Partner Oracles). *For any $(i, j, s, t) \in [\mu]^2 \times [\ell]^2$ with $i \neq j$, we say that oracles π_i^s and π_j^t are* partners *if (1) $\mathsf{Pid}_i^s = j$ and $\mathsf{Pid}_j^t = i$; (2) $\mathsf{role}_i^s \neq \mathsf{role}_j^t$; and (3) $\mathsf{sid}_i^s = \mathsf{sid}_j^t$.*

For correctness, we require that two oracles executing the AKE protocol faithfully (i.e., without adversarial interaction) derive identical session-identifiers. We also require that two such oracles reach the $\texttt{accept}$ state and derive identical session keys except with all but a negligible probability. We call a set $S \subseteq ([\mu] \times [\ell])^2$ to have a *valid pairing* if the following properties hold:

- For all $((i, s), (j, t)) \in S$, we have $i \leq j$.
- For all $(i, s) \in [\mu] \times [\ell]$, there exists a unique $(j, t) \in [\mu] \times [\ell]$ such that $i \neq j$ and either $((i, s), (j, t)) \in S$ or $((j, t), (i, s)) \in S$.

In other words, a set with a valid pairing S partners off each oracle π_i^s and π_j^t in a way that the pairing is unique and no oracle is left out without a pair. We define correctness of an AKE protocol as follows.

Definition 3.2 ($(1-\delta)$-Correctness). *An AKE protocol Π_{AKE} is $(1-\delta)$-correct if for any set with a valid pairing $S \subseteq ([\mu] \times [\ell])^2$, when we execute the AKE protocol faithfully between all the oracle pairs included in S, it holds that*

$$(1-\delta) \leq \Pr\left[\begin{array}{l} \pi_i^s \text{ and } \pi_j^t \text{ are partners} \wedge \Psi_i^s = \Psi_j^t = \texttt{accept} \\ \wedge \mathsf{k}_i^s = \mathsf{k}_j^t \neq \bot \text{ for all } ((i,s),(j,t)) \in S \end{array}\right],$$

where the probability is taken over the randomness used in the oracles.

3.2 Security Game

We define security of an AKE protocol via the following game, denoted by $G_{\Pi_{\mathsf{AKE}}}(\mu, \ell)$, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The security game is parameterized by two integers μ (the number of honest parties) and ℓ (the maximum number of protocol executions per party), and is run as follows:

Setup: $\mathcal{C}$ first chooses a secret bit $b \leftarrow\!\!\$ \{0, 1\}$. Then $\mathcal{C}$ generates the public parameter of Π_{AKE} and μ long-term key pair $\{(\mathsf{lpk}_i, \mathsf{lsk}_i) \mid i \in [\mu]\}$, and initializes the collection of oracles $\{\pi_i^s \mid i \in [\mu], s \in [\ell]\}$. $\mathcal{C}$ runs $\mathcal{A}$ providing the public parameter and all the long-term public keys $\{\mathsf{lpk}_i \mid i \in [\mu]\}$ as input.

Phase 1: $\mathcal{A}$ adaptively issues the following queries any number of times in an arbitrary order:

- $\mathsf{Send}(i, s, m)$: This query allows $\mathcal{A}$ to send an arbitrary message m to oracle π_i^s. The oracle will respond according to the protocol specification and its current internal state. To start a new oracle, the message m takes a special form:
  $\langle \mathtt{START} : \mathsf{role}, j \rangle$; $\mathcal{C}$ initializes π_i^s in the role role, having party P_j as its peer, that is, $\mathcal{C}$ sets $\mathsf{Pid}_i^s := j$ and $\mathsf{role}_i^s := \mathsf{role}$. If π_i^s is an initiator (i.e., $\mathsf{role} = \mathtt{init}$), then $\mathcal{C}$ returns the first message of the protocol.[9]
- $\mathsf{RevLTK}(i)$: For $i \in [\mu]$, this query allows $\mathcal{A}$ to learn the long-term secret key lsk_i of party P_i. After this query, P_i is said to be *corrupted.*
- $\mathsf{RegisterLTK}(i, \mathsf{lpk}_i)$: For $i \in \mathbb{N} \setminus [\mu]$, this query allows $\mathcal{A}$ to register a new party P_i with public key lpk_i. We do not require that the adversary knows the corresponding secret key. After the query, the pair (i, lpk_i) is distributed to all other oracles. Parties registered by $\mathsf{RegisterLTK}$ are corrupted by definition.
- $\mathsf{RevState}(i, s)$: This query allows $\mathcal{A}$ to learn the session-state state_i^s of oracle π_i^s. After this query, state_i^s is said to be *revealed.*
- $\mathsf{RevSessKey}(i, s)$: This query allows $\mathcal{A}$ to learn the session key k_i^s of oracle π_i^s.

Test: Once $\mathcal{A}$ decides that Phase 1 is over, it issues the following special Test-query which returns a real or a random key depending on the secret bit b.

- $\mathsf{Test}(i, s)$: If $(i, s) \notin [\mu] \times [\ell]$ or $\Psi_i^s \neq \mathtt{accept}$, $\mathcal{C}$ returns $\perp$. Else, $\mathcal{C}$ returns k_b, where $k_0 := \mathsf{k}_i^s$ and $k_1 \leftarrow \$ \mathcal{K}$ (where $\mathcal{K}$ is the session key space).

After this query, π_i^s is said to be *tested.*

Phase 2: $\mathcal{A}$ adaptively issues queries as in Phase 1.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. At this point, the tested oracle must be *fresh.* Here, an oracle π_i^s with $\mathsf{Pid}_i^s = j$[10] is *fresh* if all the following conditions hold:

1. $\mathsf{RevSessKey}(i, s)$ has not been issued;
2. if π_i^s has a partner π_j^t for some $t \in [\ell]$, then $\mathsf{RevSessKey}(j, t)$ has not been issued;
3. P_i is not corrupted or state_i^s is not revealed;
4. if π_i^s has a partner π_j^t for some $t \in [\ell]$, then P_j is not corrupted or state_j^t is not revealed;
5. if π_i^s has no partner oracle, then P_j is not corrupted.

If the tested oracle is not fresh, $\mathcal{C}$ aborts the game and outputs a random bit b' on behalf of $\mathcal{A}$. Otherwise, we say $\mathcal{A}$ wins the game if $b = b'$.

The advantage of $\mathcal{A}$ in the security game $G_{\Pi_{\mathsf{AKE}}}(\mu, \ell)$ is defined as

$$\mathsf{Adv}_{\Pi_{\mathsf{AKE}}}^{\mathsf{AKE}}(\mathcal{A}) := \left| \Pr\left[b = b'\right] - \frac{1}{2} \right|.$$

Definition 3.3 (Security of AKE Protocol). *An AKE protocol Π_{AKE} is secure if $\mathsf{Adv}_{\Pi_{\mathsf{AKE}}}^{\mathsf{AKE}}(\mathcal{A})$ is negligible for any* QPT *adversary $\mathcal{A}$.*

[9] Looking ahead, when the first message is independent of party P_j (i.e., $\mathcal{C}$ can first create the first message without knowledge of P_j and then set $\mathsf{Pid}_i^s := j$), we call the scheme *receiver oblivious.* See Section 3.4 for more details.

[10] Note that by definition, the peer id Pid_i^s of a tested oracle π_i^s is always defined.

3.3 Security Properties

In this section, we explain the security properties captured by our security model. Comparison between other protocols is differed to Section 3.5.

The freshness clauses Items 1 and 2 imply that we only exclude the reveal of session keys for the tested oracle and its partner oracles. This captures *key independence*; if the revealed keys are different from the tested oracle's key, then such keys must not enable computing the session key. Note that key independence implies resilience to "no-match attacks" presented by Li and Schäge [LS17]. This is because revealed keys have no information on the tested oracle's key. Moreover, the two items capture *implicit authentication* between the involved parties. This is because an oracle π that computes the same session key as the tested oracle but disagrees on the peer would not be a partner of the tested oracle, and hence, an adversary can obtain the tested oracle's key by querying the session key computed by π. Specifically, our model captures resistance to *unknown key-share* (UKS) attacks [BWM99]; a successful UKS attack is a specific type of attack that breaks implicit authentication where two parties compute the same session key but have different views on whom they are communicating with.

The freshness clauses Items 3 to 5 indicate that the game allows the adversary to reveal any subset of the four secret information — the long-term secret keys and the session-states of the two parties (where one party being the party defined by the tested oracle and the other its peer) — except for the combination where both the long-term secret key and session-state of one of the party is revealed. These clauses capture *weak forward secrecy* [Kra05]: the adversary can obtain the long-term secret keys of both parties if it has been passive in the protocol run of the two oracles. Another property captured by our model is resistance to *key-compromise impersonation* (KCI) attacks [BWJM97]. Recall that KCI attacks are those where the adversary uses a party P_i's long-term secret key to impersonate other parties towards P_i. This is captured by our model because the adversary can learn the long-term secret key of a tested oracle without any restrictions. Most importantly, our model captures resistance to *state leakage* [CK01, Kra05, LLM07, FSXY12] where an adversary is allowed to obtain session-states of both parties. We point out that our security model is strictly stronger than the recent models [GJ18, CCG$^+$19] that do not allow the adversary to learn sessions-states. More discussion on state leakage is provided in Section 3.5.

3.4 Property for Signal-Conforming AKE: Receiver Obliviousness

In this work, we care for a specific type of (two-round) AKE protocol that is compatible with the X3DH protocol [MP16b] used by the Signal protocol [SIG]. As explained in Section 1.2, the X3DH protocol can be viewed as a special type of AKE protocol where the Signal server acts as an (untrusted) bulletin board, where parties can store and retrieve information from. More specifically, the Signal server can be viewed as an adversary for an AKE protocol that controls the communication channel between the parties. When casting the X3DH protocol as an AKE protocol, one crucial property is that the first message of the initiator is generated *independently* of the communication partner. This is because, in secure messaging, parties are often *offline* during the initial key agreement so if the first message depended on the communication partner, then we must wait until they become online to complete the initial key agreement. Since we cannot send messages without agreeing on an initial key, such an AKE protocol where the first message depends on the communication partner cannot be used as a substitute for the X3DH protocol.

We abstract this crucial yet implicit property achieved by the X3DH protocol as *receiver obliviousness*.[11]

Definition 3.4 (Receiver Obliviousness / Signal-Conforming). *An AKE protocol is* receiver oblivious *(or* Signal-conforming*) if it is two-rounds and the initiator can compute the first-message without knowledge of the peer id and long-term public key of the communication peer.*

Many Diffie-Hellman type AKE protocols (e.g., the X3DH protocol used in Signal and some CSIDH-based AKE protocols [dKGV20, KTAT20]) can be checked to be receiver oblivious. In contrast, known generic AKE protocols such as [FSXY12, FSXY13, XLL$^+$18, HKSU20, XAY$^+$20] are not receiver oblivious since the first message requires the knowledge of the receiver's long-term public key.

[11] This property has also been called as *post-specified peers* [CK02] in the context of Internet Key Exchange (IKE) protocols.

3.5 Relation to Other Security Models

In the literature of AKE protocols, many security models have been proposed: the Bellare-Rogaway (BR) model [BR94], the Canetti-Krawczyk (CK) model [CK01], the CK+ model [Kra05, FSXY12], the extended CK (eCK) model [LLM07], and variants therein [CF12, BHJ+15, GJ18, CCG+19, HKSU20, JKRS20]. Although many of these security models are built based on similar motivations, there are subtle differences. We point out the notable similarities and differences between our model and the models listed above.

Long-Term Key Reveal. We first compare the models with respect to the secret information the adversary is allowed to obtain. All models including ours allow the adversary to obtain the party's long-term secret key $\{\mathsf{lsk}_i \mid i \in [\mu]\}$. In some models such as the BR model [BR94] and it's variants (e.g., [BHJ+15, GJ18, CCG+19])[12], this will be the only information given to an adversary. Although this may be a restricted model, it often serves as an initial step in proving the security of an AKE protocol.

Session-State Reveal. We can also consider a stronger and more realistic security model where the adversary is allowed to obtain the *secret session-states* of the parties. Unlike a party's long-term secret key where the definition is clear from context, the notion of secret session-states is rather unclear, and this is one of the main reasons for the various incomparable security models. In the original CK model [CK01], the session-state can depend arbitrary on the long-term secret and the randomness used by the party. More formally, using the terminology from Section 3.1, an adversary can query an oracle π_i^s for a secret session-state $f(\mathsf{lsk}_i, \mathsf{rand}_i^s)$ for an arbitrary function f, where rand_i^s is the randomness hardwired to the oracle π_i^s, and we say the AKE protocol is secure with respect to the session-state defined by f.[13] The eCK model [LLM07] and the CK+ model [Kra05, FSXY12] made the CK model more accessible by only considering a specific but natural set of functions.[14] The eCK model defines the secret session-state as the randomness used by the oracle (i.e., $f(\mathsf{lsk}_i, \mathsf{rand}_i^s) := \mathsf{rand}_i^s$). On the other hand, the CK+ model defines the session-state to be what we called session-state in Section 3.1. More specifically, the model allows the adversary to obtain the session-state state_i^s (defined at the implementation level) for all oracles except for the tested oracle and allows the adversary to only obtain the randomness $\mathsf{rand}_{i^*}^{s^*}$ of the tested oracle. As Cremers [Cre11, Cre09]) points out, depending on how we define the function f, state_i^s, and rand_i^s, these notions provide incomparable security guarantees. For instance, we can always artificially modify the scheme so that $\mathsf{state}_i^s := \mathsf{rand}_i^s$ but this usually results in an unnatural and less efficient implementation. Recent works [HKSU20, JKRS20] consider an arguably more simple and natural definition compared to the CK+ model where the adversary can obtain all the session-state state_i^s *including* the tested oracle. This seems to be in align with the type of state leakage considered by the double ratchet protocol and we choose to follow this formalization in our work.

Partnering. Another point of difference is how to define the partnering of two oracles, where recall that this was used to capture attacks that trivialize the security game. One popular method to define partnering of two oracles is by the so-called *matching conversations* used for instance by [BR94, Kra05, FSXY12, LLM07, CF12, BHJ+15, CCG+19, HKSU20, JKRS20]. As the name indicates, two oracles are partnered when the input-output (i.e., the conversation between the two oracles) matches. One benefit of using matching conversations is that they are simple to handle; given a particular instantiation of an AKE protocol, a matching conversation is uniquely defined. However, it was recently observed by Li and Schäge [LS17] that some protocols using matching conversations are vulnerable against *no-match attacks*, where two oracles compute the same session key but do not have matching conversations. A protocol with a no-match attack allows the adversary to trivially win the security game since it can query the oracle that is not a partner of the tested oracle but computes the same session key as the tested oracle. It was noted by Li and Schäge that this is only a hypothetical attack that takes advantage of the security model and has no meaningful consequence in the real-world. Therefore, in this work, we chose to use a more robust definition based

[12] We note that the subsequent variants differ from the original BR model [BR94] as they also model forward secrecy and KCI attacks.

[13] Note that the meaning of the session-state is different from those we defined in Section 3.1 (i.e., state_i^s). In the CK model, a "session-state" is only defined in the security model and does not capture the state_i^s specified by the implementation.

[14] These variants also strengthen the CK model by allowing the adversary to obtain the session-state of the tested oracle and further modeling KCI attacks.

on session-identifiers [CK01, dFW20]. Unlike matching conversations, session-identifiers must be explicitly defined for each AKE protocol and we note that if a session-identifier is defined to be the concatenation of sent and received messages, then defining partnering via session-identifiers and matching conversations become equivalent. Finally, we note that Li and Schäge [LS17] proposed another method to define partnering called *original-key partnering*. This has been used in [GJ18]. The original-key of two oracles is defined as the session key that is computed when the oracles are executed faithfully. Then, in the security game (i.e., in the presence of an adversary), if two oracles compute their original-key, they are said to be partners. The original-key partnering is conceptually cleaner but arguably harder to handle since we need to consider two session keys for each oracle: the original-key and the actual key, in the security game. Therefore, in this work, we use partnering based on session-identifiers.

Number of Test-query. Finally, we allow the adversary to issue only one Test-query in the security game. This *single-challenge* setting has been widely used in the literature. However, recently, in order to evaluate the tightness of the security proof, [BHJ+15, GJ18, CCG+19, JKRS20] consider the *multi-challenge* setting, where an adversary is allowed to make multiple Test-queries.

Remark 3.5 (Key Indistinguishability and Authenticity). As standard in the AKE literature, we capture both key indistinguishability and authenticity of the session keys in a single security game. In contrast, the recent work by de Saint Guilhem et al. [dFW20] consider these two properties separately and defined two security games: one for key indistinguishability and the other for authenticity.

Remark 3.6 (Implicit and Explicit Authentication). Our model captures *implicit authentication*, where each party is assured that no other party aside from the intended peer can gain access to the session key. Here, note that implicit authentication does *not* guarantee that the intended peer holds the same key. What it guarantees is that although your intended peer may be computing a different key, that peer is the only possible party that can have information on your computed session key. On the other hand, the property that also guarantees that the intended peer has computed the same session key is called *explicit authentication*. In (mutual) explicit authentication protocols, if both parties reach the `accept` state, then they are guaranteed to share the same session key. In practice, the distinction between implicit and explicit authentication is a minor issue since we can always add a key confirmation step to enhance an implicit authentication AKE protocol into an explicit one [Yan14, CCG+19, dFW20]. For instance, we can send encrypted messages or MACs under the established session key to check if the peer computed the same key without compromising security.

4 Generic Construction of Signal-Conforming AKE $\Pi_{\mathsf{SC\text{-}AKE}}$

In this section, we propose a Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ that can be used as a drop-in replacement for the X3DH protocol. Unlike the X3DH protocol, our protocol can be instantiated from post-quantum assumptions, and moreover, it also provides stronger security against state leakage. The protocol description is presented in Figure 2. Details follow.

Building Blocks. Our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ consists of the following building blocks.

- $\Pi_{\mathsf{KEM}} = (\mathsf{KEM.Setup}, \mathsf{KEM.KeyGen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ is a KEM scheme that is $\mathsf{IND\text{-}CCA}$ secure and assume we have $(1 - \delta_{\mathsf{KEM}})$-correctness, ν_{KEM}-high encapsulation key min-entropy and χ_{KEM}-high ciphertext min-entropy.

- $\Pi_{\mathsf{wKEM}} = (\mathsf{wKEM.Setup}, \mathsf{wKEM.KeyGen}, \mathsf{wKEM.Encap}, \mathsf{wKEM.Decap})$ is a KEM schemes that is $\mathsf{IND\text{-}CPA}$ secure (and not $\mathsf{IND\text{-}CCA}$ secure) and assume we have $(1-\delta_{\mathsf{wKEM}})$-correctness, ν_{wKEM}-high encapsulation key min-entropy, and χ_{wKEM}-high ciphertext min-entropy. In the following, for simplicity of presentation and without loss of generality, we assume $\delta_{\mathsf{wKEM}} = \delta_{\mathsf{KEM}}$, $\nu_{\mathsf{wKEM}} = \nu_{\mathsf{KEM}}$, $\chi_{\mathsf{wKEM}} = \chi_{\mathsf{KEM}}$.

- $\Pi_{\mathsf{SIG}} = (\mathsf{SIG.Setup}, \mathsf{SIG.KeyGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ is a signature scheme that is $\mathsf{EUF\text{-}CMA}$ secure and $(1 - \delta_{\mathsf{SIG}})$-correctness. We denote d as the bit length of the signature generated by $\mathsf{SIG.Sign}$.

Common public parameters: $(s, \mathsf{pp}_{\mathsf{KEM}}, \mathsf{pp}_{\mathsf{wKEM}}, \mathsf{pp}_{\mathsf{SIG}})$

Initiator P_i		Responder P_j
$\mathsf{lpk}_i = (\mathsf{ek}_i, \mathsf{vk}_i), \mathsf{lsk}_i = (\mathsf{dk}_i, \mathsf{sk}_i)$		$\mathsf{lpk}_j = (\mathsf{ek}_j, \mathsf{vk}_j), \mathsf{lsk}_j = (\mathsf{dk}_j, \mathsf{sk}_j)$
$(\mathsf{ek}_T, \mathsf{dk}_T) \leftarrow \mathsf{wKEM.KeyGen}(\mathsf{pp}_{\mathsf{wKEM}})$		
$\mathsf{state}_i := \mathsf{dk}_T$		
		$(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_i)$
		$(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{wKEM.Encap}(\mathsf{ek}_T)$
$\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{dk}_i, \mathsf{C})$		$\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K}); \mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$
$\mathsf{K}_T \leftarrow \mathsf{wKEM.Decap}(\mathsf{dk}_T, \mathsf{C}_T)$	$\xrightarrow{\mathsf{ek}_T}$	$\mathsf{sid}_j := P_i \Vert P_j \Vert \mathsf{lpk}_i \Vert \mathsf{lpk}_j \Vert \mathsf{ek}_T \Vert \mathsf{C} \Vert \mathsf{C}_T$
$\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K}); \mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$		$\mathsf{k}_j \Vert \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_j) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_j)$
$\mathsf{sid}_i := P_i \Vert P_j \Vert \mathsf{lpk}_i \Vert \mathsf{lpk}_j \Vert \mathsf{ek}_T \Vert \mathsf{C} \Vert \mathsf{C}_T$	$\xleftarrow{\mathsf{C}, \mathsf{C}_T, \mathsf{c}}$	$\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_j, \mathsf{sid}_j)$
$\mathsf{k}_i \Vert \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_i) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_i)$		$\mathsf{c} \leftarrow \sigma \oplus \tilde{k}$
$\sigma \leftarrow \mathsf{c} \oplus \tilde{k}$		Output the session key k_j
$\mathsf{SIG.Verify}(\mathsf{vk}_j, \mathsf{sid}_i, \sigma) \overset{?}{=} 1$		
Output the session key k_i		

Figure 2: Our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$.

- $\mathsf{F} : \mathcal{FK} \times \{0,1\}^* \rightarrow \{0,1\}^{\kappa+d}$ is a pseudo-random function family with key space $\mathcal{FK}$.
- $\mathsf{Ext} : \mathcal{S} \times \mathcal{KS} \rightarrow \mathcal{FK}$ is a strong $(\gamma_{\mathsf{KEM}}, \varepsilon_{\mathsf{Ext}})$-extractor.

Public Parameters. All the parties in the system are provided with the following public parameters as input: $(s, \mathsf{pp}_{\mathsf{KEM}}, \mathsf{pp}_{\mathsf{wKEM}}, \mathsf{pp}_{\mathsf{SIG}})$. Here, s is a random seed chosen uniformly from $\mathcal{S}$, and pp_{X} for $\mathsf{X} \in \{\mathsf{KEM}, \mathsf{wKEM}, \mathsf{SIG}\}$ are public parameters generated by $\mathsf{X.Setup}$.

Long-Term Public and Secret Keys. Each party P_i runs $(\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{KEM.KeyGen}(\mathsf{pp}_{\mathsf{KEM}})$ and $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{SIG.KeyGen}(\mathsf{pp}_{\mathsf{SIG}})$. Party P_i's long-term public key and secret key are set as $\mathsf{lpk}_i = (\mathsf{ek}_i, \mathsf{vk}_i)$ and $\mathsf{lsk}_i = (\mathsf{dk}_i, \mathsf{sk}_i)$, respectively.

Construction. A key exchange between an initiator P_i in the s-th session (i.e., π_i^s) and responder P_j in the t-th session (i.e., π_j^t) is executed as in Figure 2. More formally, we have the following.

1. Party P_i sets $\mathsf{Pid}_i^s := j$ and $\mathsf{role}_i^s := \mathtt{init}$. P_i computes $(\mathsf{dk}_T, \mathsf{ek}_T) \leftarrow \mathsf{wKEM.KeyGen}(\mathsf{pp}_{\mathsf{wKEM}})$ and sends ek_T to party P_j. P_i stores the ephemeral decapsulation key dk_T as the session-state i.e., $\mathsf{state}_i^s := \mathsf{dk}_T$.[15]

2. Party P_j sets $\mathsf{Pid}_j^t := i$ and $\mathsf{role}_j^t := \mathtt{resp}$. Upon receiving ek_T, P_j first computes $(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_i)$ and $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{wKEM.Encap}(\mathsf{ek}_T)$. Then P_j derives two PRF keys $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K})$, $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$. It then defines the session-identifier as $\mathsf{sid}_j^t := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{C} \| \mathsf{C}_T$ and computes $\mathsf{k}_j \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_j) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_j)$, where $\mathsf{k}_j \in \{0,1\}^\kappa$ and $\tilde{k} \in \{0,1\}^d$, and sets the session key as $\mathsf{k}_j^t := \mathsf{k}_j$. P_j then signs $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_j, \mathsf{sid}_j^t)$ and encrypts it as $\mathsf{c} \leftarrow \sigma \oplus \tilde{k}$. Finally, it sends $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ to P_i and sets $\Psi_j := \mathtt{accept}$. Here, note that P_j does not require to store any session-state, i.e., $\mathsf{state}_j^t = \perp$.

3. Upon receiving $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$, P_i first decrypts $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{dk}_i, \mathsf{C})$ and $\mathsf{K}_T \leftarrow \mathsf{wKEM.Decap}(\mathsf{dk}_T, \mathsf{C}_T)$, and derives two PRF keys $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K})$ and $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$. It then sets the session-identifier as $\mathsf{sid}_i^s := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{C} \| \mathsf{C}_T$ and computes $\mathsf{k}_i \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_i) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_i)$, where $\mathsf{k}_j \in \{0,1\}^\kappa$ and $\tilde{k} \in \{0,1\}^d$. P_i then decrypts $\sigma \leftarrow \mathsf{c} \oplus \tilde{k}$ and checks whether $\mathsf{SIG.Verify}(\mathsf{vk}_j, \mathsf{sid}_i^s, \sigma) = 1$ holds. If not, P_i sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{reject}, \perp, \perp)$ and stops. Otherwise, it sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{accept}, \mathsf{k}_i, \perp)$. Here, note that P_i deletes the session-state $\mathsf{state}_i^s = \mathsf{dk}_T$ at the end of the key exchange.

[15] Notice the protocol is receiver oblivious since the first message is computed independently of the receiver.

Remark 4.1 (A Note on Session-State). The session-state of the initiator P_i contains the ephemeral decryption key dk_T and P_i must store it until the peer responds. Any other information that is computed after receiving the message from the peer is immediately erased when the session key is established. In contrast, the responder P_j has no session-state because the responder directly computes the session key after receiving the initiator's message and does not have to store any session-specific information. That is, all states can be erased as soon as a session key is computed.

Remark 4.2 (Acquiring Long Term Keys). We assume without loss of generality that the users know each others long-term public keys required to initiate the Signal-conforming AKE protocol. In the Signal protocol, the initiator sends its long-term public key with the first message and the responder sends its long-term key with the second message. Then, these public keys are authenticated by means outside of the Signal protocol such as relying on "out-of-bound" authentications. We assume the same procedure is performed for our Signal-conforming AKE protocol as well. See the white paper for the X3DH protocol [MP16b, Section 4.1] for a brief discussion.

Remark 4.3 (Signed Prekeys). In the X3DH protocol, the initiator sends the first message with a signature attached called *signed prekey*. Informally, this allows Bob to *explicitly* authenticate Alice, while otherwise without the signature, Bob can only *implicitly* authenticate Alice. Moreover, this signature enhances the X3DH protocol to be *perfect* forward secret rather than being only *weak* forward secret, where the former allows the adversary to be active in the protocol run of the two oracles. Indeed, according to [MP16b], the X3DH is considered to have perfect forward secrecy. We observe that adding such signature in our protocol has the same effect as long as the added signature is not included in the session-identifier. This is due to Li and Schäge [LS17, Appendix D], who showed that adding new messages to an already secure protocol cannot lower the security as long as the derived session keys and the session-identifiers remain the same as the original protocol. Here, note the latter implies that the partnering relation remains the same. Similarly, Cremers and Feltz [CF12] show that adding a signature to the exchanged messages can enhance weak forward secrecy to perfect forward secrecy for natural classes of AKE protocols.

Security. The following theorems establish the correctness and security of our protocol $\Pi_{\mathsf{SC\text{-}AKE}}$.

Theorem 4.4 (Correctness of $\Pi_{\mathsf{SC\text{-}AKE}}$). *Assume* Π_{KEM} *and* Π_{wKEM} *are* $(1-\delta_{\mathsf{KEM}})$*-correct and* Π_{SIG} *is* $(1-\delta_{\mathsf{SIG}})$*-correct. Then,* $\Pi_{\mathsf{SC\text{-}AKE}}$ *is* $(1-\mu\ell(\delta_{\mathsf{SIG}}+2\delta_{\mathsf{KEM}})/2)$*-correct.*

Proof. It is clear that an initiator oracle and a responder oracle become partners when they execute the protocol faithfully. Moreover, if no correctness error occurs in the underlying KEM and signature schemes, the partner oracles compute an identical session key. Since each oracle is assigned to uniform randomness, the probability that a correctness error occurs in one of the underlying schemes is bounded by $\delta_{\mathsf{SIG}}+2\delta_{\mathsf{KEM}}$. Since there are at most $\mu\ell/2$ responder oracles, the AKE protocol is correct except with probability $\mu\ell(\delta_{\mathsf{SIG}}+2\delta_{\mathsf{KEM}})/2$. □

Theorem 4.5 (Security of $\Pi_{\mathsf{SC\text{-}AKE}}$). *For any* QPT *adversary* $\mathcal{A}$ *against the security of* $\Pi_{\mathsf{SC\text{-}AKE}}$ *with* μ *parties that establishes at most* ℓ *sessions per party, there exist* QPT *algorithms* $\mathcal{B}_1$ *breaking the* IND-CPA *security of* Π_{wKEM}*,* $\mathcal{B}_2$ *and* $\mathcal{B}_4$ *breaking the* IND-CCA *security of* Π_{KEM}*,* $\mathcal{B}_3$ *breaking the* EUF-CMA *security of* Π_{SIG}*, and* $\mathcal{D}_1,\ldots,\mathcal{D}_3$ *breaking the security of PRF* F *such that*

$$\mathsf{Adv}^{\mathsf{AKE}}_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mathcal{A}) \le \max\left\{\begin{array}{l} \mu^2\ell^2\cdot(\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{wKEM}}(\mathcal{B}_1)+\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_1)+\varepsilon_{\mathsf{Ext}}),\\ \mu^2\ell\cdot(\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{wKEM}}(\mathcal{B}_2)+\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_2)+\varepsilon_{\mathsf{Ext}})+\mu\ell^2\cdot\left(\frac{1}{2^{2\chi_{\mathsf{KEM}}}}+\frac{1}{2^{\nu_{\mathsf{KEM}}}}\right),\\ \mu\cdot\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B}_3),\\ \mu^2\ell\cdot\left(\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}_4)+\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_3)+\varepsilon_{\mathsf{Ext}}\right)+\mu\ell^2\cdot\frac{1}{2^{\chi_{\mathsf{KEM}}}} \end{array}\right\} + \frac{\mu\ell}{2}\cdot(\delta_{\mathsf{SIG}}+2\delta_{\mathsf{KEM}}),$$

where ν_{KEM} *(resp.* χ_{KEM}*) is the encapsulation key (resp. ciphertext) min-entropy of* Π_{wKEM} *and* Π_{KEM}*. The running time of* $\mathcal{B}_1,\ldots,\mathcal{B}_4$ *and* $\mathcal{D}_1,\ldots,\mathcal{D}_3$ *are about that of* $\mathcal{A}$*.*

The full proof of Theorem 4.5 can be found in Appendix B. Here, we provide an overview of the proof.

Strategy	Role of tested oracle	Partner oracle	$\mathsf{lsk}_{\mathtt{init}}$	$\mathsf{state}_{\mathtt{init}}$	$\mathsf{lsk}_{\mathtt{resp}}$	$\mathsf{state}_{\mathtt{resp}}$
Type-1	`init` or `resp`	Yes	✓	✗	✓	✗
Type-2	`init` or `resp`	Yes	✓	✗	✗	✓
Type-3	`init` or `resp`	Yes	✗	✓	✓	✗
Type-4	`init` or `resp`	Yes	✗	✓	✗	✓
Type-5	`init`	No	✓	✗	✗	-
Type-6	`init`	No	✗	✓	✗	-
Type-7	`resp`	No	✗	-	✓	✗
Type-8	`resp`	No	✗	-	✗	✓

Table 1: The strategy taken by the adversary in the security game when the tested oracle is fresh. "Yes" means the tested oracle has some (possibly non-unique) partner oracles and"No" means it has none. "✓" means the secret-key/session-state is revealed to the adversary, "✗" means the secret-key/session-state is not revealed. "-" means the session-state is not defined.

Proof sketch. Let $\mathcal{A}$ be an adversary that plays the security game $G_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mu, \ell)$. We distinguish between all possible strategies that can be taken by $\mathcal{A}$. Specifically, $\mathcal{A}$'s strategy can be divided into the eight types of strategies listed in Table 1. Here, each strategy is mutually independent and covers all possible (non-trivial) strategies. We point out that for our specific AKE construction we have $\mathsf{state}_{\mathtt{resp}} := \perp$ since the responder does not maintain any states (see Remark 4.1). Therefore, the Type-1 (resp. Type-3, Type-7) strategy is strictly stronger than the Type-2 (resp. Type-4, Type-8) strategy. Concretely, for our proof, we only need to consider the following four cases and to show that $\mathcal{A}$ has no advantage in each cases: (a) $\mathcal{A}$ uses the Type-1 strategy; (b) $\mathcal{A}$ uses the Type-3 strategy; (c) $\mathcal{A}$ uses the Type-5 or Type-6 strategy; (d) $\mathcal{A}$ uses the Type-7 strategy.

In cases (a), (b) and (d), the session key is informally protected by the security properties of KEM, PRF, and randomness extractor. In case (a), since the ephemeral decapsulation key dk_T is not revealed, K_T is indistinguishable from a random key due to the $\mathsf{IND\text{-}CPA}$ security of Π_{wKEM}. On the other hand, in case (b) and (d), since the initiator's decapsulation key $\mathsf{dk}_{\mathtt{init}}$ is not revealed, K is indistinguishable from a random key due to the $\mathsf{IND\text{-}CCA}$ security of Π_{KEM}. Here, we require $\mathsf{IND\text{-}CCA}$ security because there are initiator oracles other than the tested oracle that uses $\mathsf{dk}_{\mathtt{init}}$, which the reduction algorithm needs to simulate. This is in contrast to case (a) where dk_T is only used by the tested oracle. Then, in all cases, since either K_T or K has sufficient high min-entropy from the view of the adversary, Ext on input K_T or K outputs a uniformly random PRF key. Finally, we can invoke the pseudo-randomness of the PRF and argue that the session key in the tested oracle is indistinguishable from a random key.

In case (c), the session key is informally protected by the security property of the signature scheme. More concretely, in case (c), the tested oracle is an initiator and the signing key $\mathsf{sk}_{\mathtt{resp}}$ included in the long-term public key of its peer is not revealed. Then, due to the $\mathsf{EUF\text{-}CMA}$ security of Π_{SIG}, $\mathcal{A}$ cannot forge the signature for the session-identifier of the tested oracle $\mathsf{sid}_{\mathsf{test}}$. In addition, since the tested oracle has no partner oracles, no responder oracle ever signs $\mathsf{sid}_{\mathsf{test}}$. Therefore, combining these two, we conclude that the tested oracle cannot be in the `accept` state unless $\mathcal{A}$ breaks the signature scheme. In other words, when $\mathcal{A}$ queries Test, the tested oracle always returns $\perp$. Thus the session key of the tested oracle is hidden from $\mathcal{A}$. □

5 Instantiating Post-Quantum Signal-Conforming AKE $\Pi_{\mathsf{SC\text{-}AKE}}$

In this section, we present the implementation details of our post-quantum Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$. We take existing implementations of post-quantum KEMs and signature schemes submitted for the NIST PQC standardization. To instantiate our Signal-conforming AKE we pair variants of KEMs and signature schemes corresponding to the same security level. We consider security levels 1, 3 and 5 as defined by NIST for the PQC standardization. With more than 30 variants of KEM and 13 variants of signature schemes, we can create at least 128 different instantiations of post-quantum Signal-conforming AKE protocols. The provided implementation simulates post-quantum, weakly deniable authenticated key exchange between two entities. We study the efficiency of our instantiations through two metrics — the total amount of data exchanged between parties and run-time performance. Our implementation is available at [Kwi20].

5.1 Instantiation details

Our implementation is instantiated with the following building blocks:

- s: (pseudo)-randomly generated 32 bytes of data calculated at session initialization phase,
- Ext_s: uses HMAC-SHA256 as a strong randomness extractor. As an input message we use a key K_T prepended with byte `0x02` which works as a domain separator (since we also use HMAC-SHA256 as a PRF). Security of using HMAC as a strong randomness extractor is studied in [FPZ08],
- PRF: uses HMAC-SHA256 as a PRF. The session-specific sid is used as an input message to HMAC, prepended with byte `0x01`. An output from Ext_s is used as a key. Security of using HMAC as a PRF is studied in [Bel06],
- b: depends on the security level of the underlying post-quantum KEM scheme, where $b \in \{128, 192, 256\}$,
- d: depends on the byte length of the signature generated by the post-quantum signature scheme Π_{SIG},
- Π_{KEM}, Π_{wKEM}, Π_{SIG}: to instantiate $\Pi_{\mathsf{SC\text{-}AKE}}$, implementation uses pairs of KEM and signature schemes. List of the schemes used can be found in the table below. We always use the same KEM scheme for Π_{KEM} and Π_{wKEM}.

NIST security level	KEM	Signature
1	SABER, CLASSIC-MCELIECE, KYBER, NTRU HQC, SIKE, FRODOKEM, BIKE	RAINBOW, FALCON, DILITHIUM SPHINCS, PICNIC
3	SABER, NTRU, CLASSIC-MCELIECE, KYBER, SIKE, HQC, BIKE, FRODOKEM	DILITHIUM, RAINBOW PICNIC, SPHINCS
5	SABER, CLASSIC-MCELIECE, NTRU, KYBER FRODOKEM, SIKE, HQC	FALCON, RAINBOW PICNIC, SPHINCS

Table 2: Considered KEM and signature schemes under NIST security level 1, 3, and 5.

At a high level, the implementation is split into 3 main parts. The initiator's ephemeral KEM key generation (`offer` function), the recipient's session key generation (`accept` function), and initiator's session key generation (`finalize` function). Additionally there is an initialization part which performs the generation and exchange of the long-term public keys as well as dynamic initialization of memory. To evaluate the computational cost of $\Pi_{\mathsf{SC\text{-}AKE}}$, we instantiate it with concrete parameters as described above. The implementation runs 3 main functions in a loop for a fixed amount of time. We do not include the time spent in the initialization phase, hence the cost of key generation and memory initialization has no impact on the results.

Finally, we use an implementation of post-quantum algorithms that can be found in libOQS[16]. We also use LibTomCrypt[17] which provides an implementation of the building blocks HMAC, HKDF and SHA-256.

[16] https://github.com/open-quantum-safe/liboqs

[17] https://github.com/libtom/libtomcrypt

5.2 Efficiency Analysis

In this subsection, we provide an assessment of the costs related to running the concrete instantiation of $\Pi_{\mathsf{SC\text{-}AKE}}$. We provide two metrics:

- Communication cost: the amount of data exchanged between two parties trying to establish a session key
- Computational cost: number of CPU cycles spent in computation during session establishment by both parties

The computational cost of the protocol depends on the performance of the cryptographic primitives used. More precisely, the most expensive operations are those done by the post-quantum schemes. $\Pi_{\mathsf{SC\text{-}AKE}}$ performs 7 such operations during a session agreement: the initiator runs a KEM key generation, two KEM decapsulations and one signature verification, and the recipient performs two KEM encapsulations and one signing.

For benchmarking, we modeled a scenario in which two parties try to establish a session key. Alice generates and makes her long-term public key lpk_A and ephemeral KEM key ek_T publicly available. Bob retrieves the pair $(\mathsf{lpk}_\mathsf{A}, \mathsf{ek}_T)$ and uses it to perform his part of the session establishment. Namely, Bob generates the triple $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ and sends it to Alice along with its long-term public key lpk_B. Upon receipt, Alice finalizes the process by computing the session key on her side. We note that in the case of the Signal protocol, both parties communicate with a server (e.g., the Signal server), and not directly. For simplicity, we abstract this fact out of our scenario. Further note that in the Signal protocol, the long-term public keys lpk must be fetched from the server as the parties do not store the keys lpk corresponding to those that they have not communicated with before.[18]

Table 3 provides the results for Round 3 candidates of the NIST PQC standardization process.[19] The **CPU cycles** column is related to the computational cost. It is the number of cycles needed on both the initiator and responder side to run the protocol for a given instantiation. We run benchmarking on the Intel Xeon E3-1220v3 @3.1GhZ with Turbo Boost disabled. The last four columns relate to communication cost. They contain the byte size of the data exchanged during session key establishment. In particular, the lpk column contains the size of the long-term public key. The ek_T column contains the size of the ephemeral KEM key. The $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ column is the size of the triple generated by Bob. Here, the amount of data transferred from Alice to Bob is the sum of lpk and ek_T, while the amount of data transferred from Bob to Alice is the sum of lpk and $\mathsf{C}, \mathsf{C}_T, \mathsf{c}$. Finally, the column **Total** contains the total size of data exchanged between Alice and Bob.

In a scenario as described above, instantiations with Falcon, Dilithium, Saber and Kyber schemes seem to be the most promising when it comes to computational cost. The communication cost can be minimized by using the SIKE scheme as Π_{KEM} and Π_{wKEM}, but this significantly increases the computational cost.

We note that the computational cost is far less absolute as it depends on the concrete implementation of the post-quantum schemes. Our implementation is biased by the fact that it uses unoptimized, portable `C` code. There are two reasons for such a choice. First, our goal was to show the expected results on a broad number of platforms. Second, the libOQS library that we used does not provide hardware-assisted optimizations for all schemes, hence enabling those optimizations only for some algorithms would provide biased results.

Our implementation is based on open-source libraries, which makes it possible to perform fine-tuning and further analysis. For example, one could imagine a scenario for IoT devices that knows in advance which devices it may communicate with. Then, the long term keys of the devices can be exchanged prior to the session key establishment. In such a scenario, schemes with larger public keys may become more attractive since transferring long-term public keys could be done ahead of time.

[18] The X3DH protocol assumes the parties authenticate the long-term public keys through some authenticated channel [MP16b, Section 4.1].

[19] The results for all 128 instantiations can be found at [Kwi20]

Scheme	**CPU cycles**	lpk	ek_T	$(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$	**Total**
NIST security level 1					
Dilithium2/Saber Light	2770622	1856	672	3516	7900
Dilithium2/Kyber512	3059898	1984	800	3516	8284
Falcon512/NTRU hps2048509	28830055	1596	699	2088	5979
SPHINCS-SHAKE256-128f-s/Saber Light	269464814	704	672	18448	20528
NIST security level 3					
Dilithium4/Saber	4204171	2752	992	5542	12038
Dilithium4/NTRU hps2048677	24513381	2690	930	5226	11536
SPHINCS-SHAKE256-192f-s/Kyber768	337783175	1232	1184	37840	41488
Dilithium4/SIKE p610	790625496	2222	462	4338	9244
NIST security level 5					
Falcon1024/Saber Fire	37423092	3105	1312	4274	11796
Falcon1024/Kyber1024	37875710	3361	1568	4466	12756
Falcon1024/SIKE p751	356918904	2357	564	2522	7800
SPHINCS-SHAKE256-256f-s/SIKE p751	1041010995	628	564	50408	52228

Table 3: Computational and communication cost of running $\Pi_{\mathsf{SC\text{-}AKE}}$ instantiated with various post-quantum schemes.

Note on Low Quality Network Links. We anticipate $\Pi_{\mathsf{SC\text{-}AKE}}$ to be used with handheld devices and areas with a poor quality network connection. In such cases, larger key, ciphertext and signature sizes generated may negatively impact the quality of the connection. Network packet loss is an additional factor which should be considered when choosing schemes for concrete instantiation.

Data on the network is exchanged in packets. The maximum transmission unit (MTU) defines the maximal size of a single packet, usually set to 1500 bytes. Ideally, the size of data sent between participants in a single pass is less than MTU. Network quality is characterized by a packet loss rate. When a packet is lost, the TCP protocol ensures that it is retransmitted, where each retransmission causes a delay. A typical data loss on a high-quality network link is below 1%, while data loss on a mobile network depends on the strength of the network signal.

Depending on the scheme used, increased packet loss may negatively impact session establishment time (see [PST19]). For example, a scheme instantiated with `Falcon512/NTRU hps2048509` requires exchange of $npacks = 7$ packets over the network, where instantiation with `SPHINCS-SHAKE256-128f-simple/Saber Light` requires 16. Assuming increased packet rate loss of 5%, the probability of losing a packet in the former case is $1 - (1 - rate)^{npacks} = 30\%$, where in the latter it is 56%. In the latter case, at the median, every other session key establishment will experience packet retransmission and hence a delay.

6 Adding Deniability to Our Signal-Conforming AKE $\Pi_{\mathsf{SC\text{-}AKE}}$

In this section, we provide a theory-oriented discussion on the deniability aspect of our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$. In the following, we first informally show that $\Pi_{\mathsf{SC\text{-}AKE}}$ already has a very weak form of deniability that may be acceptable in some applications. We then show that we can slightly modify $\Pi_{\mathsf{SC\text{-}AKE}}$ to satisfy a more stronger notion of deniability. As it is common with all deniable AKE protocols secure against key-compromise attacks [DGK06, YZ10, VGIK20], we prove deniability by relying on strong knowledge-type assumptions, including a variant of the *plaintext-awareness* (PA) for the KEM scheme [BR95, BDPR98, BP04].

Weak Deniability of $\Pi_{\mathsf{SC\text{-}AKE}}$. Our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ already satisfies a weak notion of deniability, where the communication transcript does not leave a trace of the two parties if both parties honestly executed the AKE protocol. Namely, an adversary that is passively collecting the communication transcript cannot convince a third party that communication between two parties took place. Informally, this can be observed by checking that all the contents in the transcript can be simulated by the adversary on its own. This notion of weak deniability may suffice for some particular applications when the two engaging parties fully trust each other for the correct execution of the protocol, and if they are fine by assuming that corruption will not occur. However, in other cases, we may want to guarantee deniability even in the case the communicating peer may be compromised, or even worse, acting maliciously. We discuss this stronger notion of deniability next.

6.1 Definition of Deniability and Tool Preparation

We follow a simplified definition of deniability for AKE protocols introduced by Di Raimondo et al. [DGK06]. Discussion on the simplification is provided in Remark 6.3. Let Π be an AKE protocol and KeyGen be the key generation algorithm. That is, for any integer $\mu = \mu(\kappa)$ representing the number of parties in the system, define $\mathsf{KeyGen}(1^\kappa, \mu) \to (\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$, where pp is the public parameter used by the system and $\overrightarrow{\mathsf{lpk}} := \{\mathsf{lpk}_i \mid i \in [\mu]\}$ and $\overrightarrow{\mathsf{lsk}} := \{\mathsf{lsk}_i \mid i \in [\mu]\}$ are the corresponding long-term public and secret keys of the μ parties, respectively.

Let $\mathcal{M}$ denote an adversary that engages in an AKE protocol with μ-honest parties in the system with long-term public keys $\overrightarrow{\mathsf{lpk}}$, acting as either an initiator or a responder. $\mathcal{M}$ may run individual sessions against an honest party in a concurrent manner and may deviate from the AKE protocol in an arbitrary fashion. The goal of $\mathcal{M}$ is not to impersonate someone to an honest party P but to collect (cryptographic) evidence that an honest party P interacted with $\mathcal{M}$. Therefore, when $\mathcal{M}$ interacts with P, it can use a long-term public key $\mathsf{lpk}_{\mathcal{M}}$ that can be either associated to or not to $\mathcal{M}$'s identity (that may possibly be generated maliciously). We then define the *view* of the adversary $\mathcal{M}$ as the entire sets of input and output of $\mathcal{M}$ and the *session keys* computed in all the protocols in which $\mathcal{M}$ participated with an honest party. Here, we assume in case the session is not completed by $\mathcal{M}$, the session key is defined as $\perp$. We denote this view as $\mathsf{View}_{\mathcal{M}}(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$.

In order to define deniability, we consider a simulator SIM that simulates the view of honest parties (both initiator and responder) to the adversary $\mathcal{M}$ *without* knowledge of the corresponding long-term secret keys $\overrightarrow{\mathsf{lsk}}$ of the honest parties. Specifically, SIM takes as input all the input given to the adversary $\mathcal{M}$ (along with the description of $\mathcal{M}$) and simulates the view of $\mathcal{M}$ with the real AKE protocol Π. We denote this simulated view as $\mathsf{SIM}_{\mathcal{M}}(\mathsf{pp}, \overrightarrow{\mathsf{lpk}})$. Roughly, if the view simulated by $\mathsf{SIM}_{\mathcal{M}}$ is indistinguishable from those generated by $\mathsf{View}_{\mathcal{M}}$, then we say the AKE protocol is deniable since $\mathcal{M}$ could have run $\mathsf{SIM}_{\mathcal{M}}$ (which does not take any secret information as input) to generate its view in the real protocol. More formally, we have the following.

Definition 6.1 (Deniability). *We say an AKE protocol* Π *with key generation algorithm* KeyGen *is* deniable*, if for any integer* $\mu = \mathsf{poly}(\kappa)$ *and PPT adversary* $\mathcal{M}$*, there exist a PPT simulator* $\mathsf{SIM}_{\mathcal{M}}$ *such that the following two distributions are (computationally) indistinguishable for any PPT distinguisher* $\mathcal{D}$*:*

$$\mathcal{F}_{\mathsf{Real}} := \{\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \mathsf{View}_{\mathcal{M}}(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}}) : (\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}}) \leftarrow \mathsf{KeyGen}(1^\kappa, \mu)\},$$
$$\mathcal{F}_{\mathsf{Sim}} := \{\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \mathsf{SIM}_{\mathcal{M}}(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}) : (\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}}) \leftarrow \mathsf{KeyGen}(1^\kappa, \mu)\}.$$

When $\mathcal{M}$ *is semi-honest (i.e., it follows the prescribed protocol), we say* Π *is* deniable against semi-honest adversaries*. When* $\mathcal{M}$ *is malicious (i.e., it takes any efficient strategy), we say* Π *is* deniable against malicious adversaries*.*

Remark 6.2 (Including Public Information and Session Keys). It is crucial that the two distributions $\mathcal{F}_{\mathsf{Real}}$ and $\mathcal{F}_{\mathsf{Sim}}$ include the public information $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}})$. Otherwise, $\mathsf{SIM}_{\mathcal{M}}$ can simply create its own set of $(\mathsf{pp}', \overrightarrow{\mathsf{lpk}}', \overrightarrow{\mathsf{lsk}}')$ and simulate the view to $\mathcal{M}$. However, this does not correctly capture deniability in the real-world since $\mathcal{M}$ would not be able to convince anybody with such a view using public information that it cooked up on its own. In addition, it is essential that the value of the session key is part of the output of $\mathsf{SIM}_{\mathcal{M}}$. This guarantees that the contents of the sessions authenticated by the session key can also be denied.

Remark 6.3 (Comparison between Prior Definition). Our definition is weaker than the deniability notion originally proposed by Di Raimondo et al. [DGK06]. In their definition, an adversary $\mathcal{M}$ (and therefore the simulator $\mathsf{SIM}_{\mathcal{M}}$) is also provided as input some auxiliary information aux that can depend non-trivially on $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$.[20] For instance, this allows to capture information that $\mathcal{M}$ may have obtained by eavesdropping conversations between honest parties (which is not modeled by $\mathsf{View}_{\mathcal{M}}$). Since our goal is to provide a minimal presentation on the deniability of our protocol, we only focus on the weaker definition where $\mathcal{M}$ does not obtain such auxiliary information. We leave it as future work to prove our protocol deniable in the sense of Di Raimondo et al. [DGK06]. We also note that stronger forms of deniability are known and formalized in the universally composable (UC) model [DKSW09, UG15, UG18], however, AKE protocols satisfying such a strong deniability notion are known to achieve weaker security guarantees. For instance, as noted in [UG18], an AKE protocol cannot be on-line deniable while also being secure against KCI attacks.

Remark 6.4 (Extending to Malicious Quantum Adversaries). We only consider classical deniability above. Although we can show deniability for semi-honest quantum adversaries, we were not able to do so for malicious quantum adversaries. This is mainly due to the fact that to prove deniability against malicious classical adversaries, we require a strong knowledge type assumption (i.e., plaintext-awareness for KEM) that assumes an extractor can invoke the adversary multiple of times on the *same* randomness. We leave it as an interesting problem to formally define a set of tools that allow to show deniability even against malicious quantum adversaries.

Required Tools. To argue deniability in the following section we rely on the following tools: ring signature, plaintext-aware ($\mathsf{PA\text{-}1}$) secure KEM scheme, and a non-interactive zero-knowledge (NIZK) argument.We use standard notions of ring signatures and NIZK arguments and we provide the formal definitions in Appendices A.1 and A.3. On the other hand, we use a slightly stronger variant of $\mathsf{PA\text{-}1}$ secure KEM schemes than those originally defined in [BR95, BDPR98, BP04]. Informally, a KEM scheme is $\mathsf{PA\text{-}1}$ secure if for any adversary $\mathcal{M}$ that outputs a valid ciphertext C, there is an extractor $\mathsf{Ext}_{\mathcal{M}}$ that outputs the matching session key K. In our work, we require $\mathsf{PA\text{-}1}$ security to hold even when $\mathcal{M}$ is given multiple public keys rather than a single public key [MSs12]. We note that although Di Raimondo et al. [DGK06] considered the standard notion of $\mathsf{PA\text{-}1}$ security, we observe that their proof only works in the case where multiple public keys are considered. Finally, we further require the extractor $\mathsf{Ext}_{\mathcal{M}}$ to be efficiently computable given $\mathcal{M}$. The formal definition is provided in Appendix A.2.

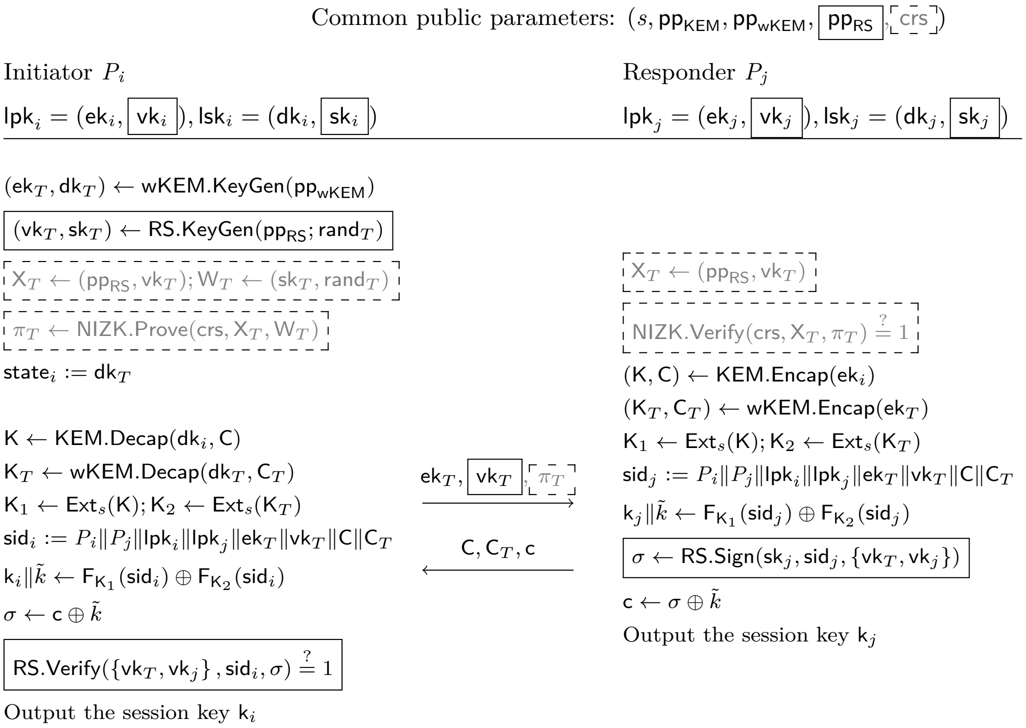
6.2 Deniable Signal-Conforming AKE $\Pi_{\mathsf{SC\text{-}DAKE}}$ against Semi-Honest Adversaries

We first provide a Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ that is deniable against semi-honest adversaries. The construction of $\Pi_{\mathsf{SC\text{-}DAKE}}$ is a simple modification of $\Pi_{\mathsf{SC\text{-}AKE}}$ where a standard signature is replaced by a ring signature. In the subsequent section, we show how to modify $\Pi_{\mathsf{SC\text{-}DAKE}}$ to a protocol that is deniable against malicious adversaries by relying on further assumptions. The high-level idea presented in this section naturally extends to the malicious setting. An overview of $\Pi_{\mathsf{SC\text{-}DAKE}}$ and $\Pi'_{\mathsf{SC\text{-}DAKE}}$ is provided in Figure 3.

Building Blocks. Our deniable Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ against semi-honest adversaries consists of the following building blocks.

- $\Pi_{\mathsf{KEM}} = (\mathsf{KEM.Setup}, \mathsf{KEM.KeyGen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ is a KEM scheme that is $\mathsf{IND\text{-}CCA}$ secure and assume we have $(1 - \delta_{\mathsf{KEM}})$-correctness, ν_{KEM}-high encapsulation key min-entropy and χ_{KEM}-high ciphertext min-entropy.
- $\Pi_{\mathsf{wKEM}} = (\mathsf{wKEM.Setup}, \mathsf{wKEM.KeyGen}, \mathsf{wKEM.Encap}, \mathsf{wKEM.Decap})$ is a KEM schemes that is $\mathsf{IND\text{-}CPA}$ secure (and not $\mathsf{IND\text{-}CCA}$ secure) and assume we have $(1 - \delta_{\mathsf{wKEM}})$-correctness, ν_{wKEM}-high encapsulation key min-entropy, and χ_{wKEM}-high ciphertext min-entropy. In the following, for simplicity of presentation and without loss of generality, we assume $\delta_{\mathsf{wKEM}} = \delta_{\mathsf{KEM}}$, $\nu_{\mathsf{wKEM}} = \nu_{\mathsf{KEM}}$, $\chi_{\mathsf{wKEM}} = \chi_{\mathsf{KEM}}$.

[20] Although in [DGK06, Definition 2], aux is defined as fixed information that $\mathcal{M}$ cannot adaptively choose, we observe that in their proof they implicitly assume that aux is sampled adaptively from some distribution dependent on $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$. Such a definition of aux is necessary to invoke $\mathsf{PA\text{-}2}$ security of the underlying encryption scheme.



Common public parameters: $(s, \mathsf{pp}_{\mathsf{KEM}}, \mathsf{pp}_{\mathsf{wKEM}}, \boxed{\mathsf{pp}_{\mathsf{RS}}}, \boxed{\color{gray}\mathsf{crs}})$

Initiator P_i

$\mathsf{lpk}_i = (\mathsf{ek}_i, \boxed{\mathsf{vk}_i}), \mathsf{lsk}_i = (\mathsf{dk}_i, \boxed{\mathsf{sk}_i})$

Responder P_j

$\mathsf{lpk}_j = (\mathsf{ek}_j, \boxed{\mathsf{vk}_j}), \mathsf{lsk}_j = (\mathsf{dk}_j, \boxed{\mathsf{sk}_j})$

$(\mathsf{ek}_T, \mathsf{dk}_T) \leftarrow \mathsf{wKEM.KeyGen}(\mathsf{pp}_{\mathsf{wKEM}})$

$\boxed{(\mathsf{vk}_T, \mathsf{sk}_T) \leftarrow \mathsf{RS.KeyGen}(\mathsf{pp}_{\mathsf{RS}}; \mathsf{rand}_T)}$

$\boxed{\color{gray}\mathsf{X}_T \leftarrow (\mathsf{pp}_{\mathsf{RS}}, \mathsf{vk}_T); \mathsf{W}_T \leftarrow (\mathsf{sk}_T, \mathsf{rand}_T)}$

$\boxed{\color{gray}\pi_T \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{X}_T, \mathsf{W}_T)}$

$\mathsf{state}_i := \mathsf{dk}_T$

$$\xrightarrow{\mathsf{ek}_T, \boxed{\mathsf{vk}_T}, \boxed{\color{gray}\pi_T}}$$

$\boxed{\color{gray}\mathsf{X}_T \leftarrow (\mathsf{pp}_{\mathsf{RS}}, \mathsf{vk}_T)}$

$\boxed{\color{gray}\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{X}_T, \pi_T) \overset{?}{=} 1}$

$(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_i)$

$(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{wKEM.Encap}(\mathsf{ek}_T)$

$\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K}); \mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$

$\mathsf{sid}_j := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{vk}_T \| \mathsf{C} \| \mathsf{C}_T$

$\mathsf{k}_j \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_j) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_j)$

$\boxed{\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{sk}_j, \mathsf{sid}_j, \{\mathsf{vk}_T, \mathsf{vk}_j\})}$

$\mathsf{c} \leftarrow \sigma \oplus \tilde{k}$

Output the session key k_j

$$\xleftarrow{\mathsf{C}, \mathsf{C}_T, \mathsf{c}}$$

$\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{dk}_i, \mathsf{C})$

$\mathsf{K}_T \leftarrow \mathsf{wKEM.Decap}(\mathsf{dk}_T, \mathsf{C}_T)$

$\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K}); \mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$

$\mathsf{sid}_i := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{vk}_T \| \mathsf{C} \| \mathsf{C}_T$

$\mathsf{k}_i \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_i) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_i)$

$\sigma \leftarrow \mathsf{c} \oplus \tilde{k}$

$\boxed{\mathsf{RS.Verify}(\{\mathsf{vk}_T, \mathsf{vk}_j\}, \mathsf{sid}_i, \sigma) \overset{?}{=} 1}$

Output the session key k_i

Figure 3: Deniable Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ and $\Pi'_{\mathsf{SC\text{-}DAKE}}$. The components that differ from the non-deniable protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ is indicated by a box. The protocol with (resp. without) the gray and dotted-box component satisfies deniability against malicious (resp. semi-honest) adversaries.

- $\Pi_{\mathsf{RS}} = (\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Verify})$ is a ring signature scheme that is anonymous and unforgeable. We denote d as the bit length of the signature generated by $\mathsf{RS.Sign}$.
- $\mathsf{F} : \mathcal{FK} \times \{0,1\}^* \to \{0,1\}^{\kappa+d}$ is a pseudo-random function family with key space $\mathcal{FK}$.
- $\mathsf{Ext} : \mathcal{S} \times \mathcal{KS} \to \mathcal{FK}$ is a strong $(\gamma_{\mathsf{KEM}}, \varepsilon_{\mathsf{Ext}})$-extractor.

Public Parameters. All the parties in the system are provided the following public parameters as input: $(s, \mathsf{pp}_{\mathsf{KEM}}, \mathsf{pp}_{\mathsf{wKEM}}, \mathsf{pp}_{\mathsf{RS}})$. Here, s is a random seed chosen uniformly from $\mathcal{S}$, and $\mathsf{params}_{\mathsf{X}}$ for $\mathsf{X} \in \{\mathsf{KEM}, \mathsf{wKEM}, \mathsf{RS}\}$ are public parameters generated by $\mathsf{X.Setup}$.

Long-Term Public and Secret Keys. Each party P_i runs $(\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{KEM.KeyGen}(\mathsf{pp}_{\mathsf{KEM}})$ and $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{RS.KeyGen}(\mathsf{pp}_{\mathsf{RS}})$. Party P_i's long-term public key and secret key are set as $\mathsf{lpk}_i = (\mathsf{ek}_i, \mathsf{vk}_i)$ and $\mathsf{lsk}_i = (\mathsf{dk}_i, \mathsf{sk}_i)$, respectively.

Construction. A key exchange between an initiator P_i in the s-th session (i.e., π_i^s) and responder P_j in the t-th session (i.e., π_j^t) is executed as in Figure 2. More formally, we have the following.

1. Party P_i sets $\mathsf{Pid}_i^s := j$ and $\mathsf{role}_i^s := \mathtt{init}$. P_i computes $(\mathsf{dk}_T, \mathsf{ek}_T) \leftarrow \mathsf{wKEM.KeyGen}(\mathsf{pp}_{\mathsf{wKEM}})$ and $(\mathsf{vk}_T, \mathsf{sk}_T) \leftarrow \mathsf{RS.KeyGen}(\mathsf{pp}_{\mathsf{RS}})$, and sends $(\mathsf{ek}_T, \mathsf{vk}_T)$ to party P_j. P_i erases the signing key sk_T and stores the ephemeral decapsulation key dk_T as the session-state i.e., $\mathsf{state}_i^s := \mathsf{dk}_T$.[21]
2. Party P_j sets $\mathsf{Pid}_j^t := i$ and $\mathsf{role}_j^t := \mathtt{resp}$. Upon receiving $(\mathsf{ek}_T, \mathsf{vk}_T)$, P_j first computes $(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_i)$ and $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{wKEM.Encap}(\mathsf{ek}_T)$ and derives two PRF keys $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K})$, $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$. It then defines the session-identifier as $\mathsf{sid}_j^t := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{vk}_T \| \mathsf{C} \| \mathsf{C}_T$ and computes $\mathsf{k}_j \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_j) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_j)$, where $\mathsf{k}_j \in \{0,1\}^\kappa$ and $\tilde{k} \in \{0,1\}^d$. P_j sets the session key as $\mathsf{k}_j^t := \mathsf{k}_j$. P_j then signs $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{sk}_j, \mathsf{sid}_j^t, \{\mathsf{vk}_T, \mathsf{vk}_j\})$ and encrypts it as $\mathsf{c} \leftarrow \sigma \oplus \tilde{k}$. Finally, it sends $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ to P_i and sets $\Psi_j := \mathtt{accept}$. Here, note that P_j does not require to store any session-state, i.e., $\mathsf{state}_j^t = \perp$.
3. Upon receiving $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$, P_i first decrypts $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{dk}_i, \mathsf{C})$ and $\mathsf{K}_T \leftarrow \mathsf{wKEM.Decap}(\mathsf{dk}_T, \mathsf{C}_T)$, and derives two PRF keys $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K})$ and $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$. It then sets the session-identifier as $\mathsf{sid}_i^s := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{vk}_T \| \mathsf{C} \| \mathsf{C}_T$ and computes $\mathsf{k}_i \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_i) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_i)$, where $\mathsf{k}_i \in \{0,1\}^\kappa$ and $\tilde{k} \in \{0,1\}^d$. P_i then decrypts $\sigma \leftarrow \mathsf{c} \oplus \tilde{k}$ and checks whether $\mathsf{RS.Verify}(\{\mathsf{vk}_T, \mathsf{vk}_j\}, \mathsf{sid}_i^s, \sigma) = 1$ holds. If not, P_i sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{reject}, \perp, \perp)$ and stops. Otherwise, P_i sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{accept}, \mathsf{k}_i, \perp)$. Here, note that P_i deletes the session-state $\mathsf{state}_i^s = \mathsf{dk}_T$ at the end of the key exchange.

Security. We first check that $\Pi_{\mathsf{SC\text{-}DAKE}}$ is correct and secure as a standard AKE protocol. Since the proof is similar in most parts to the non-deniable protocol $\Pi_{\mathsf{SC\text{-}AKE}}$, we defer the details to Appendix C. The main difference from the security proof of $\Pi_{\mathsf{SC\text{-}AKE}}$ is that we have to make sure that using a ring signature instead of a standard signature does not allow the adversary to mount a key-compromise impersonation (KCI) attack (see Section 3.3 for the explanation on KCI attacks).

Theorem 6.5 (Correctness of $\Pi_{\mathsf{SC\text{-}DAKE}}$). *Assume Π_{KEM} and Π_{wKEM} are $(1-\delta_{\mathsf{KEM}})$-correct and Π_{RS} is $(1-\delta_{\mathsf{RS}})$-correct. Then, the Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ is $(1-\mu\ell(\delta_{\mathsf{RS}}+2\delta_{\mathsf{KEM}})/2)$-correct.*

Theorem 6.6 (Security of $\Pi_{\mathsf{SC\text{-}DAKE}}$). *For any* QPT *adversary $\mathcal{A}$ against the security of $\Pi_{\mathsf{SC\text{-}DAKE}}$ with μ parties that establishes at most ℓ sessions per party, there exist* QPT *algorithms $\mathcal{B}_1$ breaking the* IND-CPA *security of Π_{wKEM}, $\mathcal{B}_2$ and $\mathcal{B}_4$ breaking the* IND-CCA *security of Π_{KEM}, $\mathcal{B}_3$ breaking the unforgeability of Π_{RS}, and $\mathcal{D}_1, \ldots, \mathcal{D}_3$ breaking the security of PRF* F *such that*

$$\mathsf{Adv}^{\mathsf{AKE}}_{\Pi_{\mathsf{SC\text{-}DAKE}}}(\mathcal{A}) \leq \max \left\{ \begin{array}{l} \mu^2\ell^2 \cdot (\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{wKEM}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_1) + \varepsilon_{\mathsf{Ext}}), \\ \mu^2\ell \cdot (\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{wKEM}}(\mathcal{B}_2) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_2) + \varepsilon_{\mathsf{Ext}}) + \mu\ell^2 \cdot \left(\frac{1}{2^{2\chi_{\mathsf{KEM}}}} + \frac{1}{2^{\nu_{\mathsf{KEM}}}}\right), \\ \mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{RS}}(\mathcal{B}_3), \\ \mu^2\ell \cdot \left(\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}_4) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_3) + \varepsilon_{\mathsf{Ext}}\right) + \mu\ell^2 \cdot \frac{1}{2^{\chi_{\mathsf{KEM}}}}. \end{array} \right\} + \frac{\mu\ell}{2} \cdot (\delta_{\mathsf{RS}} + 2\delta_{\mathsf{KEM}}),$$

[21] Notice the protocol is receiver oblivious since the first message is computed independently of the receiver.

where ν_{KEM} is the encapsulation key min-entropy of Π_{wKEM} and Π_{KEM}, and χ_{KEM} is the ciphertext min-entropy of Π_{wKEM} and Π_{KEM}. The running time of $\mathcal{B}_1, \ldots, \mathcal{B}_4$ and $\mathcal{D}_1, \ldots, \mathcal{D}_3$ are about that of $\mathcal{A}$.

The following guarantees deniability of our protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ against semi-honest adversaries.

Theorem 6.7 (Deniability of $\Pi_{\mathsf{SC\text{-}DAKE}}$ against Semi-Honest Adversaries). *Assume Π_{RS} is anonymous. Then, the Signal-conforming protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ is deniable against semi-honest adversaries.*

Proof. Let $\mathcal{M}$ be any PPT semi-honest adversary. We explain the behavior of the simulator $\mathsf{SIM}_{\mathcal{M}}$ by considering three cases: (a) $\mathcal{M}$ initializes an initiator P_i, (b) $\mathcal{M}$ queries the initiator P_i on message $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$, and (c) $\mathcal{M}$ queries the responder P_j on message $(\mathsf{ek}_T, \mathsf{vk}_T)$. In case (a), $\mathsf{SIM}_{\mathcal{M}}$ runs the honest initiator algorithm and returns $(\mathsf{ek}_T, \mathsf{vk}_T)$ as specified by the protocol. In case (b), since $\mathcal{M}$ is semi-honest, we are guaranteed that it runs the honest responder algorithm to generate $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$. In particular, since $\mathcal{M}$ is run on randomness sampled by $\mathsf{SIM}_{\mathcal{M}}$, $\mathsf{SIM}_{\mathcal{M}}$ gets to learn the key K that was generated along with C. Therefore, $\mathsf{SIM}_{\mathcal{M}}$ runs the real initiator algorithm except that it uses K extracted from $\mathcal{M}$ rather than computing $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{dk}_i, \mathsf{C})$. Here, note that $\mathsf{SIM}_{\mathcal{M}}$ cannot run the latter since it does not know the corresponding dk_i held by an honest initiator party P_i. In case (c), similarly to case (b), $\mathsf{SIM}_{\mathcal{M}}$ learns dk_T and sk_T used by $\mathcal{M}$ to generate ek_T and vk_T. Therefore, $\mathsf{SIM}_{\mathcal{M}}$ runs the honest responder algorithm except that it runs $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{sk}_T, \mathsf{sid}_j, \{\mathsf{vk}_T, \mathsf{vk}_j\})$ instead of running $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{sk}_j, \mathsf{sid}_j, \{\mathsf{vk}_T, \mathsf{vk}_j\})$ as in the real protocol. Here, note that $\mathsf{SIM}_{\mathcal{M}}$ cannot run the latter since it does not know the corresponding sk_j held by an honest responder party P_j.

Let us analyze $\mathsf{SIM}_{\mathcal{M}}$. First, for case (a), the output by $\mathsf{SIM}_{\mathcal{M}}$ is distributed exactly as in the real transcript. Next, for case (b), the only difference between the real distribution and $\mathsf{SIM}_{\mathcal{M}}$'s output distribution (which is the derived session key k) is that $\mathsf{SIM}_{\mathcal{M}}$ uses the KEM key K output by $\mathsf{KEM.Encap}$ to compute the session key rather than using the KEM key decrypted using $\mathsf{KEM.Decap}$ with the initiator party P_i's decryption key dk_i. However, by $(1 - \delta_{\mathsf{KEM}})$-correctness of Π_{KEM}, these two KEM keys are identical with probability at least $(1 - \delta_{\mathsf{KEM}})$. Hence, the output distribution of $\mathsf{SIM}_{\mathcal{M}}$ and the real view are indistinguishable. Finally, for case (c), the only difference between the real distribution and $\mathsf{SIM}_{\mathcal{M}}$'s output distribution (which is the derived session key and the message sent $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$) is how the ring signature is generated. While the real protocol uses the signing key sk_j of the responder party P_j, the simulator $\mathsf{SIM}_{\mathcal{M}}$ uses sk_T. However, the signatures outputted by these two distributions are computationally indistinguishable assuming the anonymity of Π_{RS}. Hence, the output distribution of $\mathsf{SIM}_{\mathcal{M}}$ and the real view are indistinguishable.

Combining everything together, we conclude the proof. □

6.3 Deniable Signal-Conforming AKE $\Pi'_{\mathsf{SC\text{-}DAKE}}$ against Malicious Adversaries

We discuss security of our Signal-conforming AKE protocol $\Pi'_{\mathsf{SC\text{-}DAKE}}$ against malicious adversaries. As depicted in Figure 3, to achieve deniability against malicious adversaries, we modify the protocol so that the initiator party adds a NIZK proof attesting to the fact that it constructed the verification key of the ring signature vk_T honestly. Formally, we require the following additional building blocks.

Building Blocks. Our deniable Signal-conforming AKE protocol $\Pi'_{\mathsf{SC\text{-}DAKE}}$ against malicious adversaries requires the following primitives in addition to those required by $\Pi_{\mathsf{SC\text{-}DAKE}}$ in the previous section.

- $\Pi_{\mathsf{KEM}} = (\mathsf{KEM.Setup}, \mathsf{KEM.KeyGen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ is an $\mathsf{IND\text{-}CCA}$ secure KEM scheme as in the previous section that additionally satisfies $\mathsf{PA}_\mu\text{-}1$ security with an efficiently constructible extractor, where μ is the number of parties in the system.

- $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ is a NIZK argument system for the relation $\mathcal{R}_{\mathsf{RS}}$ where $(\mathsf{X}, \mathsf{W}) \in \mathcal{R}_{\mathsf{RS}}$ if and only if the statement $\mathsf{X} = (\mathsf{pp}, \mathsf{vk})$ and witness $\mathsf{W} = (\mathsf{sk}, \mathsf{rand})$ satisfy $(\mathsf{vk}, \mathsf{sk}) = \mathsf{RS.KeyGen}(\mathsf{pp}; \mathsf{rand})$.

Additional Assumption. We require a knowledge-type assumption to prove deniability against malicious adversaries. Considering that all of the previous AKE protocols satisfying a strong form of security and deniability require such knowledge-type assumptions [DGK06, YZ10, VGIK20], this seems unavoidable. On the other hand, there are protocols achieving a strong form of deniability from standard assumptions [DKSW09, UG15, UG18], however, they make a significant compromise in the security such as being vulnerable to KCI attacks and state leakages.

The following knowledge assumption is defined similarly in spirit to those of Di Raimondo et al. [DGK06] that assumed that for any adversary $\mathcal{M}$ that outputs a valid MAC, then there exists an extractor algorithm Ext that extracts the corresponding MAC key. Despite it being a strong knowledge-type assumption in the standard model, we believe it holds in the random oracle model if we further assume the NIZK comes with an *online* knowledge extractor[22] like those provided by Fischlin's NIZK [Fis05]. We leave it to future works to investigate the credibility of the following assumption and those required to prove deniability of the X3DH protocol [VGIK20].

Assumption 6.8 (Key-Awareness Assumption for $\Pi'_{\mathsf{SC\text{-}DAKE}}$)**.** *We say that* $\Pi'_{\mathsf{SC\text{-}DAKE}}$ *has the* key-awareness property *if for all PPT adversaries* $\mathcal{M}$ *interacting with a real protocol execution in the deniability game as in Definition 6.1, there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{M}}$ *such that for any choice of* $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}}) \in \mathsf{KeyGen}(1^\kappa, \mu)$*, whenever* $\mathcal{M}$ *outputs a ring signature verification key* vk *and a* NIZK *proof* π *for the language* $\mathcal{L}_{\mathsf{RS}}$*, then* $\mathsf{Ext}_{\mathcal{M}}$ *taking input the same input as* $\mathcal{M}$ *(including its randomness) outputs a signing key* sk *such that* $(\mathsf{vk}, \mathsf{sk}) \in \mathsf{RS.KeyGen}(\mathsf{pp}_{\mathsf{RS}})$ *for any* $\mathsf{pp}_{\mathsf{RS}} \in \mathsf{RS.Setup}(1^\kappa)$*.*

With the added building blocks along with the key-awareness assumption, we prove the following theorem. The high-level approach is similar to the previous proof against semi-honest adversaries, however, the concrete proof is rather involved. The main technicality is when invoking the PA_μ-1 security: if we do the reduction naively, the extractor needs the randomness used to sample the ring signature key pairs of the honest party but the simulator of the deniability game does not know such randomness. We circumvent this issue by hard-wiring the verification key of the ring signature used by the adversary and by considering PA_μ-1 security against a non-uniform adversary.

Theorem 6.9 (Deniability of $\Pi'_{\mathsf{SC\text{-}DAKE}}$ against Malicious Adversaries). *Assume* Π_{KEM} *is* PA_μ-1 *secure with an efficiently constructible extractor,* Π_{RS} *is anonymous,* Π_{NIZK} *is sound,*[23] *and the key-awareness assumption in Assumption 6.8 holds. Then, the Signal-conforming protocol* $\Pi'_{\mathsf{SC\text{-}DAKE}}$ *with* μ *parties is deniable against malicious adversaries.*

Proof. The high-level idea of the proof is similar to those of Theorem 6.7. Below, we consider a sequence of simulators $\mathsf{SIM}_{\mathcal{M},i}$ where the first and last simulators $\mathsf{SIM}_{\mathcal{M},0}$ and $\mathsf{SIM}_{\mathcal{M},3}$ simulate the real and simulated protocols, respectively. That is, $\mathsf{SIM}_{\mathcal{M},3}$ is the desired simulator $\mathsf{SIM}_{\mathcal{M}}$. We define $\mathcal{F}_i$ to be the distribution of $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}})$ along with the output of $\mathsf{SIM}_{\mathcal{M},i}$. Our goal is to prove that $\mathcal{F}_0$ and $\mathcal{F}_3$ are indistinguishable.

$\mathsf{SIM}_{\mathcal{M},0}$: It is given $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$ as input and simulates the interaction with the adversary $\mathcal{M}$ following the protocol description of the real-world. Here, note that $\mathcal{M}$ is invoked by $\mathsf{SIM}_{\mathcal{M},0}$ on input $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}})$ with uniform randomness. By definition $\mathcal{F}_{\mathsf{Real}} := \mathcal{F}_0$.

$\mathsf{SIM}_{\mathcal{M},1}$: This is the same as $\mathsf{SIM}_{\mathcal{M},0}$ except that whenever $\mathcal{M}$ queries an honest responder party P_j on input $(\mathsf{ek}_T, \mathsf{vk}_T, \pi_T)$, $\mathsf{SIM}_{\mathcal{M},1}$ extracts the corresponding secret ring signature signing key sk_T. More formally, due to the key-awareness assumption of $\Pi'_{\mathsf{SC\text{-}DAKE}}$, for any PPT $\mathcal{M}$, there exists a PPT extractor $\mathsf{Ext}_{\mathcal{M}}$ such that whenever $\mathcal{M}$ outputs a ring signature verification key vk_T and a NIZK proof π_T for the language $\mathcal{L}_{\mathsf{RS}}$, then $\mathsf{Ext}_{\mathcal{M}}$ taking input the same input as $\mathcal{M}$ (including its randomness) outputs a signing key sk_T such that $(\mathsf{vk}_T, \mathsf{sk}_T) \in \mathsf{RS.KeyGen}(\mathsf{pp}_{\mathsf{RS}})$. Since $\mathsf{SIM}_{\mathcal{M},1}$ knows all the input and randomness fed to $\mathcal{M}$, it can run $\mathsf{Ext}_{\mathcal{M}}$. Namely, whenever $\mathcal{M}$ makes the above query, $\mathsf{SIM}_{\mathcal{M},1}$ invokes $\mathsf{Ext}_{\mathcal{M}}$ on input fed to $\mathcal{M}$ until that point along with its initial randomness and extracts sk_T. Since

[22] This roughly guarantees that the witness from a proof can be extracted without rewinding the adversary.

[23] We note that this is redundant since it is implicitly implied by the key-awareness assumption. We only include it for clarity.

the output of $\mathsf{SIM}_{\mathcal{M},1}$ is unaltered, the distribution $\mathcal{F}_1$ is identical to the previous game. Below, for simplicity, we assume that $\mathcal{M}$ always outputs sk_T whenever it queries an honest responder party P_j on input $(\mathsf{ek}_T, \mathsf{vk}_T, \pi_T)$. This is without loss of generality since we can combine $\mathcal{M}$ and $\mathsf{Ext}_{\mathcal{M}}$ and view it as another adversary against the deniability game.

$\mathsf{SIM}_{\mathcal{M},2}$: This is the same as $\mathsf{SIM}_{\mathcal{M},1}$ except that when $\mathcal{M}$ queries an honest responder party P_j on input $(\mathsf{ek}_T, \mathsf{vk}_T, \mathsf{sk}_T, \pi_T)$, $\mathsf{SIM}_{\mathcal{M},1}$ responds as in the real protocol except that it runs $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{sk}_T, \mathsf{sid}_j, \{\mathsf{vk}_T, \mathsf{vk}_j\})$ instead of running $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{sk}_j, \mathsf{sid}_j, \{\mathsf{vk}_T, \mathsf{vk}_j\})$. Due to the anonymity of the ring signature Π_{RS}, the distributions $\mathcal{F}_1$ and $\mathcal{F}_2$ are indistinguishable.

Before explaining the next simulator, notice that we can view the combined algorithm $(\mathsf{SIM}_{\mathcal{M},2}, \mathcal{M})$ as a ciphertext creator $\mathcal{C}$ for the PA_μ-1 security of the KEM scheme Π_{KEM}. Formally, we decompose $\mathsf{SIM}_{\mathcal{M},2}$ into two algorithms: $\mathsf{SIM}'_{\mathcal{M},2}$ and $\mathsf{O}_{\mathsf{dec}}$, where $\mathsf{SIM}'_{\mathcal{M},2}$ is identical to $\mathsf{SIM}_{\mathcal{M},2}$ except that it outsources the decapsulation of ciphertexts corresponding to those of honest initiator parties to $\mathsf{O}_{\mathsf{dec}}$. That is, $\mathsf{SIM}'_{\mathcal{M},2}$ proceeds as $\mathsf{SIM}_{\mathcal{M},2}$ except that when $\mathcal{M}$ queries the honest initiator P_i on message $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$, it queries (i, C) to $\mathsf{O}_{\mathsf{dec}}$ to receive the corresponding KEM key K. Since $\mathsf{SIM}'_{\mathcal{M},2}$ no longer requires the secret KEM keys $\{\mathsf{dk}_i \mid i \in [\mu]\}$ of the honest initiator parties, we can assume that $\mathsf{SIM}'_{\mathcal{M},2}$ only takes as input $(\mathsf{pp}, \{\mathsf{ek}_i \mid i \in [\mu]\})$. Here, we also assume it has μ-ring signature verification keys $\{\mathsf{vk}_i \mid i \in [\mu]\}$ hard-wired rather than $\mathsf{SIM}'_{\mathcal{M},2}$ generating it on its own. At this point, it is clear that the combined algorithm $(\mathsf{SIM}'_{\mathcal{M},2}, \mathcal{M})$ can be viewed as a valid ciphertext creator $\mathcal{C}$ that outputs the view of $\mathcal{M}$ as the string v, where $\mathsf{O}_{\mathsf{dec}}$ corresponds to the decapsulation oracle $\mathsf{KEM.Decap}$ run by the challenger in $\mathsf{Exp}^{\mathsf{dec}}_{\mathcal{C},\mathcal{D}}$. Then, by the PA_μ-1 security, there must exist an extractor $\mathcal{E}_{\mathcal{C}}$ that simulates $\mathsf{O}_{\mathsf{dec}}$ that only takes as input $(\mathsf{pp}, (\mathsf{ek}_i)_{i\in[\mu]}, \mathsf{rand}_{\mathcal{C}})$, where $\mathsf{rand}_{\mathcal{C}}$ is the randomness used by $\mathcal{C}$ (i.e., by $(\mathsf{SIM}'_{\mathcal{M},2}, \mathcal{M})$). Moreover, such an extractor $\mathcal{E}_{\mathcal{C}}$ is efficiently constructible given the description of $\mathcal{C}$. Here, note that $\mathsf{rand}_{\mathcal{C}}$ does *not* include the randomness used to generate the μ-ring signature verification keys since we hard-wire these to the description of $\mathsf{SIM}'_{\mathcal{M},2}$. In particular, $\mathcal{E}_{\mathcal{C}}$ does not require randomness used to generate $\overrightarrow{\mathsf{lpk}}$ to be executed. We are now ready to define the next simulator.

$\mathsf{SIM}_{\mathcal{M},3} := \mathsf{SIM}_{\mathcal{M}}$: This is the same as $\mathsf{SIM}_{\mathcal{M},2}$ except that it constructs the extractor $\mathcal{E}_{\mathcal{C}}$ and when $\mathcal{M}$ queries the honest initiator P_i on message $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ it runs $\mathcal{E}_{\mathcal{C}}(i, \mathsf{C})$ instead of $\mathsf{O}_{\mathsf{dec}}(\mathsf{dk}_i, \mathsf{C})$. Notice that $\mathsf{SIM}_{\mathcal{M},3}$ no longer requires any long-term secret key $\overrightarrow{\mathsf{lsk}}$ to simulate $\mathcal{M}$. Due to the PA_μ-1 security of the KEM scheme Π_{KEM}, the two distributions $\mathcal{F}_2$ and $\mathcal{F}_3 := \mathcal{F}_{\mathsf{Sim}}$ are indistinguishable.

This completes the proof. □

Finally, it remains to show that the $\Pi'_{\mathsf{SC\text{-}DAKE}}$ is correct and secure as a standard Signal-conforming AKE protocol. Due to the correctness of Π_{NIZK}, the correctness of $\Pi'_{\mathsf{SC\text{-}DAKE}}$ follows from Theorem 6.5. Moreover, the security of $\Pi'_{\mathsf{SC\text{-}DAKE}}$ follows almost immediately from the proof of Theorem 6.6. The only difference is that in the proof of Lemma C.1 (which is a sub-lemma used to prove Theorem 6.6), the reduction algorithm that does not know the corresponding signing key sk_T of the verification key vk_T invokes the zero-knowledge simulator to simulate the proof π_T. The rest of the proof is identical.

Acknowledgement. The second author was supported by JST CREST Grant Number JPMJCR19F6. The third and fourth authors were supported by the Innovate UK Research Grant 104423 (PQ Cybersecurity).

References

[ACD19] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 129–158. Springer, Heidelberg, May 2019.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer, Heidelberg, August 1998.

[Bel06] Mihir Bellare. New proofs for nmac and hmac: Security without collision-resistance. Cryptology ePrint Archive, Report 2006/043, 2006.

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer, Heidelberg, April 2006.

[BFG$^+$20] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. Towards post-quantum security for signal's x3dh handshake. In *SAC 2020*, 2020. https://eprint.iacr.org/2019/1356.

[BHJ$^+$15] Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 629–658. Springer, Heidelberg, March 2015.

[BP04] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 48–62. Springer, Heidelberg, December 2004.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995.

[BS20] Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 493–522. Springer, Heidelberg, May 2020.

[BSJ$^+$17] Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 619–650. Springer, Heidelberg, August 2017.

[BWJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer, Heidelberg, December 1997.

[BWM99] Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 154–170. Springer, Heidelberg, March 1999.

[CCG$^+$19] Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. Highly efficient key exchange protocols with optimal tightness. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 767–797. Springer, Heidelberg, August 2019.

[CF12] Cas J. F. Cremers and Michele Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS 2012*, volume 7459 of *LNCS*, pages 734–751. Springer, Heidelberg, September 2012.

[CGCD$^+$17] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 451–466, 2017.

[CGCD$^+$20] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. *Journal of Cryptology*, pages 1–70, 2020.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, Heidelberg, May 2001.

[CK02] Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 143–161. Springer, Heidelberg, August 2002. http://eprint.iacr.org/2002/120/.

[CKS08] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 127–145. Springer, Heidelberg, April 2008.

[Cre09] Cas J. F. Cremers. Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS authenticated key exchange protocol. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 20–33. Springer, Heidelberg, June 2009.

[Cre11] Cas Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *ASIACCS 11*, pages 80–91. ACM Press, March 2011.

[dFW20] C. D. de Saint Guilhem, M. Fischlin, and B. Warinschi. Authentication in key-exchange: Definitions, relations and composition. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pages 288–303, 2020.

[DGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 400–409. ACM Press, October / November 2006.

[dKGV20] Bor de Kock, Kristian Gjøsteen, and Mattia Veroni. Practical isogeny-based key-exchange with optimal tightness. In *SAC 2020*, 2020. https://eprint.iacr.org/2020/1165.

[DKSW09] Yevgeniy Dodis, Jonathan Katz, Adam Smith, and Shabsi Walfish. Composability and on-line deniability of authentication. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 146–162. Springer, Heidelberg, March 2009.

[DV19] F. Betül Durak and Serge Vaudenay. Bidirectional asynchronous ratcheted key agreement with linear complexity. In Nuttapong Attrapadung and Takeshi Yagi, editors, *IWSEC 19*, volume 11689 of *LNCS*, pages 343–362. Springer, Heidelberg, August 2019.

[FHKP13] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Heidelberg, February / March 2013.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, August 2005.

[FPZ08] Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. HMAC is a randomness extractor and applications to TLS. In Masayuki Abe and Virgil Gligor, editors, *ASIACCS 08*, pages 21–32. ACM Press, March 2008.

[FSXY12] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 467–484. Springer, Heidelberg, May 2012.

[FSXY13] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS 13*, pages 83–94. ACM Press, May 2013.

[GJ18] Kristian Gjøsteen and Tibor Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 95–125. Springer, Heidelberg, August 2018.

[GKRS20] Siyao Guo, Pritish Kamath, Alon Rosen, and Katerina Sotiraki. Limits on the efficiency of (ring) LWE based non-interactive key exchange. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 374–395. Springer, Heidelberg, May 2020.

[HKSU20] Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. Generic authenticated key exchange in the quantum random oracle model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 389–422. Springer, Heidelberg, May 2020.

[JKRS20] Tibor Jager, Eike Kiltz, Doreen Riepel, and Sven Schäge. Tightly-secure authenticated key exchange, revisited. Cryptology ePrint Archive, Report 2020/1279, 2020.

[JMM19a] Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 159–188. Springer, Heidelberg, May 2019.

[JMM19b] Daniel Jost, Ueli Maurer, and Marta Mularczyk. A unified and composable take on ratcheting. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 180–210. Springer, Heidelberg, December 2019.

[KF14] Kaoru Kurosawa and Jun Furukawa. 2-pass key exchange protocols from CPA-secure KEM. In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 385–401. Springer, Heidelberg, February 2014.

[Kra05] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, Heidelberg, August 2005.

[KTAT20] Tomoki Kawashima, Katsuyuki Takashima, Yusuke Aikawa, and Tsuyoshi Takagi. An efficient authenticated key exchange from random self-reducibility on csidh, 2020. to appear in *ICISC 2020*.

[Kwi20] Kris Kwiatkowski. Signal-conforming ake protocol implementation, 2020. https://github.com/post-quantum-cryptography/post-quantum-state-leakage-secure-ake.

[LLM07] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 1–16. Springer, Heidelberg, November 2007.

[LS17] Yong Li and Sven Schäge. No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1343–1360. ACM Press, October / November 2017.

[MP16a] Moxie Marlinspike and Trevor Perrin. The double ratchet algorithm, November 2016. https://signal.org/docs/specifications/doubleratchet/.

[MP16b] Moxie Marlinspike and Trevor Perrin. The x3dh key agreement protocol, November 2016. https://signal.org/docs/specifications/x3dh/.

[MSs12] Steven Myers, Mona Sergi, and abhi shelat. Blackbox construction of a more than non-malleable CCA1 encryption scheme from plaintext awareness. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 149–165. Springer, Heidelberg, September 2012.

[Pei20] Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 463–492. Springer, Heidelberg, May 2020.

[PR18] Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 3–32. Springer, Heidelberg, August 2018.

[PS14] David Pointcheval and Olivier Sanders. Forward secure non-interactive key exchange. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 21–39. Springer, Heidelberg, September 2014.

[PST19] Christian Paquin, Douglas Stebila, and Goutam Tamvada. Benchmarking post-quantum cryptography in tls. Cryptology ePrint Archive, Report 2019/1447, 2019.

[SIG] Signal protocol: Technical documentation. https://signal.org/docs/.

[UG15] Nik Unger and Ian Goldberg. Deniable key exchanges for secure messaging. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 1211–1223. ACM Press, October 2015.

[UG18] Nik Unger and Ian Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *PoPETs*, 2018(1):21–66, January 2018.

[VGIK20] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. On the cryptographic deniability of the Signal protocol. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20, Part II*, volume 12147 of *LNCS*, pages 188–209. Springer, Heidelberg, October 2020.

[XAY$^+$20] Haiyang Xue, Man Ho Au, Rupeng Yang, Bei Liang, and Haodong Jiang. Compact authenticated key exchange in the quantum random oracle model. Cryptology ePrint Archive, Report 2020/1282, 2020.

[XLL$^+$18] Haiyang Xue, Xianhui Lu, Bao Li, Bei Liang, and Jingnan He. Understanding and constructing AKE via double-key key encapsulation mechanism. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 158–189. Springer, Heidelberg, December 2018.

[Yan14] Zheng Yang. Modelling simultaneous mutual authentication for authenticated key exchange. In Jean Luc Danger, Mourad Debbabi, Jean-Yves Marion, Joaquin Garcia-Alfaro, and Nur Zincir Heywood, editors, *Foundations and Practice of Security*, pages 46–62, Cham, 2014. Springer International Publishing.

[YCL18] Zheng Yang, Yu Chen, and Song Luo. Two-message key exchange with strong security from ideal lattices. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 98–115. Springer, Heidelberg, April 2018.

[YZ10] Andrew Chi-Chih Yao and Yunlei Zhao. Deniable internet key exchange. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 329–348. Springer, Heidelberg, June 2010.

A Omitted Preliminaries

In this section, we provide the definitions of standard cryptographic primitives used throughout the main body.

A.1 Ring Signatures

Definition A.1 (Ring Signature Schemes). *A ring signature scheme consists of four PPT algorithms* $\Pi_{\mathsf{RS}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

$\mathsf{Setup}(1^\kappa) \to \mathsf{pp}$: *The setup algorithm takes as input a security parameter* 1^κ *and outputs a public parameters* pp *used by the scheme.*

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$: *The key generation algorithm on input the public parameters* pp *outputs a pair of public and secret keys* $(\mathsf{vk}, \mathsf{sk})$*.*

$\mathsf{Sign}(\mathsf{sk}, \mathsf{M}, \mathsf{R}) \to \sigma$: *The signing algorithm on input a secret key* sk*, a message* M*, and a list of public keys, i.e., a* ring*,* $\mathsf{R} = \{\mathsf{vk}_1, \dots, \mathsf{vk}_N\}$*, outputs a signature* σ*.*

$\mathsf{Verify}(\mathsf{R}, \mathsf{M}, \sigma) \to 1/0$: *The verification algorithm on input a ring* $\mathsf{R} = \{\mathsf{vk}_1, \dots, \mathsf{vk}_N\}$*, a message* M*, and a signature* σ*, outputs either* 1 *or* 0*.*

Definition A.2 ($(1-\delta)$-Correctness). *We say a ring signature scheme* Π_{RS} *is* $(1-\delta)$*-correct if for all* $\kappa \in \mathbb{N}$*,* $N = \mathsf{poly}(\kappa)$*,* $j \in [N]$*, and every message* M*,*

$$(1-\delta) \le \Pr\left[\ \mathsf{Verify}(\mathsf{R}, \mathsf{M}, \sigma) = 1 \ \middle|\ \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa); \\ (\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})\ \forall i \in [N]; \\ \mathsf{R} := (\mathsf{vk}_1, \cdots, \mathsf{vk}_N); \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_j, \mathsf{M}, \mathsf{R}). \end{array} \right].$$

Definition A.3 (Anonymity). *We say a ring signature scheme* Π_{RS} *is* anonymous *if, for any* $\kappa \in \mathbb{N}$*,* $\mathsf{pp} \in \mathsf{Setup}(1^\kappa)$*,* $(\mathsf{vk}_0, \mathsf{sk}_0), (\mathsf{vk}_1, \mathsf{sk}_1) \in \mathsf{KeyGen}(\mathsf{pp})$*, and message* M*, and any PPT distinguisher* $\mathcal{A}$*, the two distributions* $D_b := \{\sigma : \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_b, \mathsf{M}, \{\mathsf{vk}_0, \mathsf{vk}_1\})\}$ *for* $b \in \{0, 1\}$ *are indistinguishable.*

Definition A.4 (Unforgeability). *We say a ring signature scheme* Π_{RS} *is* unforgeable *if, for all* $\kappa \in \mathbb{N}$ *and* $N = \mathsf{poly}(\kappa)$*, any PPT adversary* $\mathcal{A}$ *has at most negligible advantage in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$ *and generates key pairs* $(\mathsf{vk}_i, \mathsf{sk}_i) = \mathsf{KeyGen}(\mathsf{pp}; r_i)$ *for all* $i \in [N]$ *using random coins* r_i*. It sets* $\mathsf{VK} := \{\mathsf{vk}_i \mid i \in [N]\}$ *and initializes two empty sets* SL *and* CL*.*

(ii) The challenger provides pp *and* VK *to* $\mathcal{A}$*;*

(iii) $\mathcal{A}$ *can make signing and corruption queries an arbitrary polynomial number of times:*

- $(\mathsf{sign}, i, \mathsf{M}, \mathsf{R})$*: The challenger checks if* $\mathsf{vk}_i \in \mathsf{R}$ *and if so it computes the signature* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \mathsf{M}, \mathsf{R})$*. The challenger provides* σ *to* $\mathcal{A}$ *and adds* $(i, \mathsf{M}, \mathsf{R})$ *to* SL*;*

- $(\mathsf{corrupt}, i)$: *The challenger adds* vk_i *to* CL *and returns* r_i *to* $\mathcal{A}$.

(iv) $\mathcal{A}$ *outputs* $(\mathsf{R}^*, \mathsf{M}^*, \sigma^*)$. *If* $\mathsf{R}^* \subset \mathsf{VK} \backslash \mathsf{CL}$, $(\cdot, \mathsf{M}^*, \mathsf{R}^*) \notin \mathsf{SL}$, *and* $\mathsf{Verify}(\mathsf{R}^*, \mathsf{M}^*, \sigma^*) = 1$, *then we say the adversary* $\mathcal{A}$ *wins.*

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{RS}}(\mathcal{A}) = \Pr[\mathcal{A}\ wins]$.

A.2 Plaintext-Awareness

We define plaintext-awareness (PA) for KEM schemes [BR95, BP04] where multiple keys are considered [MSs12]. We observe that the standard PA security defined for a single key does not immediately imply a multi-key variant and that the original proof of deniability by Di Raimondo et al. [DGK06, Theorem 2 and 3] crucially relies on the multi-key variant. Furthermore, below we consider strengthening of the (already strong) PA security where the efficient extractor $\mathcal{E}_\mathcal{C}$ for the ciphertext creator $\mathcal{C}$ can be constructed efficiently given the description of $\mathcal{C}$. This is required in the deniability proof as the simulator must construct such $\mathcal{E}_\mathcal{C}$ given the description of the adversary $\mathcal{M}$.

Definition A.5 (Plaintext-Awareness). *Let* $t = t(\kappa)$ *be an integer. We say a KEM scheme* Π_{KEM} *is* plaintext-aware $(\mathsf{PA_t}\text{-}1)$ *secure if for all* $\kappa \in \mathbb{N}$ *and (non-uniform) PPT ciphertext creator* $\mathcal{C}$, *there exists a PPT extractor* $\mathcal{E}_\mathcal{C}$ *such that for any PPT distinguisher* $\mathcal{D}$, *the following two experiments* $\mathsf{Exp}^{\mathsf{dec}}_{\mathcal{C},\mathcal{D}}$ *and* $\mathsf{Exp}^{\mathsf{ext}}_{\mathcal{C},\mathcal{E}_\mathcal{C},\mathcal{D}}$ *are indistinguishable:*

$\underline{\mathsf{Exp}^{\mathsf{dec}}_{\mathcal{C},\mathcal{D}}(1^\kappa)}$:

(i) *The challenger runs* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$ *and* $(\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *for* $i \in [t]$. *It then runs* $\mathcal{C}$ *on input* $(\mathsf{pp}, (\mathsf{ek}_i)_{i \in [t]})$ *with uniform randomness* $\mathsf{rand}_\mathcal{C}$.

(ii) *When* $\mathcal{C}$ *queries an index-ciphertext pair* (i, C) *to the challenger, the challenger returns* $\mathsf{KEM.Decap}(\mathsf{dk}_i, \mathsf{C})$. *Here,* $\mathcal{C}$ *can query the challenger polynomially many times in an arbitrary manner.*

(iii) $\mathcal{C}$ *finally outputs a string* v.

(iv) *The experiment outputs* $\mathcal{D}(v) \to b \in \{0,1\}$.

$\underline{\mathsf{Exp}^{\mathsf{ext}}_{\mathcal{C},\mathcal{E}_\mathcal{C},\mathcal{D}}(1^\kappa)}$:

(i) *The challenger runs* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$ *and* $(\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *for* $i \in [t]$. *It then runs* $\mathcal{C}$ *on input* $(\mathsf{pp}, (\mathsf{ek}_i)_{i \in [t]})$ *with uniform randomness* $\mathsf{rand}_\mathcal{C}$, *and runs* $\mathcal{E}_\mathcal{C}$ *on input* $(\mathsf{pp}, (\mathsf{ek}_i)_{i \in [t]}, \mathsf{rand}_\mathcal{C})$.

(ii) $\mathcal{C}$ *can adaptively query an index-ciphertext pair* C *polynomially many times to the challenger. When the challenger receives* (i, C), *it returns* $\mathcal{E}_\mathcal{C}(\mathtt{query}, (i, \mathsf{C}), \mathsf{rand}_\mathcal{C})$.[24]

(iii) $\mathcal{C}$ *finally outputs a string* v.

(iv) *The experiment outputs* $\mathcal{D}(v) \to b \in \{0,1\}$.

Moreover, we say the extractor $\mathcal{E}_\mathcal{C}$ *is* efficiently constructible *if the description of* $\mathcal{E}_\mathcal{C}$ *can be efficiently computed from the description of* $\mathcal{C}$.

A.3 Non-Interactive Zero-Knowledge

Let $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ be a polynomial time recognizable binary relation. For $(x, w) \in \mathcal{R}$, we call x as the statement and w as the witness. Let $\mathcal{L}$ be the corresponding **NP** language $\mathcal{L} = \{x \mid \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$. Below, we define non-interactive zero-knowledge arguments for **NP** languages.

[24] We assume algorithms $\mathcal{C}$ and $\mathcal{E}_\mathcal{C}$ are stateful.

Definition A.6 (NIZK Arguments). *A non-interactive zero-knowledge (NIZK) argument* Π_{NIZK} *for the relation* $\mathcal{R}$ *consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$*.*

$\mathsf{Setup}(1^\kappa) \rightarrow \mathsf{crs}$*: The setup algorithm takes as input the security parameter* 1^κ *and outputs a common reference string* crs*.*

$\mathsf{Prove}(\mathsf{crs}, x, w) \rightarrow \pi$*: The prover's algorithm takes as input a common reference string* crs*, a statement* x*, and a witness* w *and outputs a proof* π*.*

$\mathsf{Verify}(\mathsf{crs}, x, \pi) \rightarrow \top$ *or* $\bot$*: The verifier's algorithm takes as input a common reference string, a statement* x*, and a proof* π *and outputs* $\top$ *to indicate acceptance of the proof and* $\bot$ *otherwise.*

Definition A.7 (Correctness). *We say a NIZK argument* Π_{NIZK} *is* correct *if for all pairs* $(x, w) \in \mathcal{R}$*, if we run* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)$*, then we have*

$$\Pr[\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) : \mathsf{Verify}(\mathsf{crs}, x, \pi) = \top] = 1.$$

Definition A.8 (Soundness). *We say a* NIZK *argument* Π_{NIZK} *is* sound *if for all PPT adversaries* $\mathcal{A}$*, if we run* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)$*, then we have*

$$\Pr[(x, \pi) \leftarrow \mathcal{A}(1^\kappa, \mathsf{crs}) : x \notin \mathcal{L} \wedge \mathsf{Verify}(\mathsf{crs}, x, \pi) = \top] = \mathsf{negl}(\kappa).$$

Definition A.9 (Zero-Knowledge). *We say a* NIZK *argument* Π_{NIZK} *is* zero-knowledge *if for all PPT adversaries* $\mathcal{A}$*, there exists a PPT simulator* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that if we run* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)$ *and* $(\overline{\mathsf{crs}}, \bar{\tau}) \leftarrow \mathsf{Sim}_1(1^\kappa)$*, then we have*

$$\left|\Pr[\mathcal{A}^{\mathsf{O}_0(\mathsf{crs},\cdot,\cdot)}(1^\kappa, \mathsf{crs}) = 1] - \Pr[\mathcal{A}^{\mathsf{O}_1(\bar{\mathsf{crs}},\bar{\tau},\cdot,\cdot)}(1^\kappa, \overline{\mathsf{crs}}) = 1]\right| = \mathsf{negl}(\kappa),$$

where $\mathsf{O}_0(\mathsf{crs}, x, w)$ *outputs* $\mathsf{Prove}(\mathsf{crs}, x, w)$ *if* $(x, w) \in \mathcal{R}$ *and* $\bot$ *otherwise, and* $\mathsf{O}_1(\overline{\mathsf{crs}}, \bar{\tau}, x, w)$ *outputs* $\mathsf{Sim}_2(\overline{\mathsf{crs}}, \bar{\tau}, x)$ *if* $(x, w) \in \mathcal{R}$ *and* $\bot$ *otherwise.*

B Omitted Proofs for Signal-conforming AKE $\Pi_{\mathsf{SC\text{-}AKE}}$

We prove the security of our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$.

Proof of Theorem 4.5. Let $\mathcal{A}$ be an adversary that plays the security game $G_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mu, \ell)$ with the challenger $\mathcal{C}$ with advantage $\mathsf{Adv}^{\mathsf{AKE}}_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mathcal{A}) = \epsilon$. In order to prove Theorem 4.5, we distinguish between the strategy that can be taken by the $\mathcal{A}$. Specifically, $\mathcal{A}$'s strategy can be divided into the eight types of strategies listed in Table 1. Here, each strategy is mutually independent and covers all possible (non-trivial) strategies.[25] We point out that for our specific AKE construction we have $\mathsf{state}_{\mathsf{resp}} := \bot$ since the responder does not maintain any states (see Remark 4.1). Therefore, the Type-1 (resp. Type-3, Type-7) strategy is strictly stronger than the Type-2 (resp. Type-4, Type-8) strategy. We only include the full types of strategies in Table 1 as we believe it would be helpful when proving other AKE protocols, and note that our proof implicitly handles both strategies at the same time.

For each possible strategy taken by $\mathcal{A}$, we construct an algorithm that breaks one of the underlying assumptions by using such an adversary $\mathcal{A}$ as a subroutine. More formally, we construct seven algorithms $\mathcal{B}_1, \ldots, \mathcal{B}_4$ and $\mathcal{D}_1, \ldots, \mathcal{D}_3$ satisfying the following:

1. If $\mathcal{A}$ uses the Type-1 (or Type-2) strategy, then $\mathcal{B}_1$ succeeds in breaking the IND-CPA security of Π_{wKEM} with advantage $\approx \frac{1}{\mu^2 \ell^2}\epsilon$ or $\mathcal{D}_1$ succeeds in breaking the security of PRF F with advantage $\approx \frac{1}{\mu^2 \ell^2}\epsilon$.

[25] We note that although we can consider an adversary $\mathcal{A}$ that makes no reveal queries (i.e., all lsk and state are either 7 or "-"), we can exclude them without loss of generality since such $\mathcal{A}$ can always be modified into an adversary $\mathcal{A}'$ that follows one of the strategies listed in Table 1.

2. If $\mathcal{A}$ uses the Type-3 (or Type-4) strategy, then $\mathcal{B}_2$ succeeds in breaking the $\mathsf{IND\text{-}CCA}$ security of Π_{KEM} with advantage $\approx \frac{1}{\mu^2\ell}\epsilon$ or $\mathcal{D}_2$ succeeds in breaking the security of PRF F with advantage $\approx \frac{1}{\mu^2\ell}\epsilon$.
3. If $\mathcal{A}$ uses the Type-5 or Type-6 strategy, then $\mathcal{B}_3$ succeeds in breaking the $\mathsf{EUF\text{-}CMA}$ security of Π_{SIG} with advantage $\approx \frac{1}{\mu}\epsilon$.
4. If $\mathcal{A}$ uses the Type-7 (or Type-8) strategy, then $\mathcal{B}_4$ succeeds in breaking the $\mathsf{IND\text{-}CCA}$ security of Π_{KEM} with advantage $\approx \frac{1}{\mu^2\ell}\epsilon$ or $\mathcal{D}_3$ succeeds in breaking the security of PRF F with advantage $\approx \frac{1}{\mu^2\ell}\epsilon$.

We present a security proof structured as a sequence of games. Without loss of generality, we assume that $\mathcal{A}$ always issues a Test-query. In the following, let S_j denote the event that $b = b'$ occurs in game G_j and let $\epsilon_j := |\Pr[\mathsf{S}_j] - 1/2|$ denote the advantage of the adversary in game G_j. Regardless of the strategy taken by $\mathcal{A}$, all proofs share a common game sequence G_0-G_1 as described below.

Game G_0. This game is identical to the original security game. We thus have

$$\epsilon_0 = \epsilon.$$

Game G_1. This game is identical to G_0, except that we add an abort condition. Let $\mathsf{E}_{\mathsf{corr}}$ be the event that there exist two partner oracles π_i^s and π_j^t that do not agree on the same session key. If $\mathsf{E}_{\mathsf{corr}}$ occurs, then $\mathcal{C}$ aborts (i.e., sets $\mathcal{A}$'s output to be a random bit) at the end of the game.

There are at most $\mu\ell/2$ responder oracles and each oracle is assigned uniform randomness. From Theorem 4.4, the probability of error occurring during the security game is at most $\mu\ell(\delta_{\mathsf{SIG}} + 2\delta_{\mathsf{KEM}})/2$. Therefore, $\mathsf{E}_{\mathsf{corr}}$ occurs with probability at most $\mu\ell(\delta_{\mathsf{SIG}} + 2\delta_{\mathsf{KEM}})/2$. We thus have

$$|\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_1]| \leq \frac{\mu\ell}{2} \cdot (\delta_{\mathsf{SIG}} + 2\delta_{\mathsf{KEM}}).$$

In the following games we assume no decryption error or signature verification error occurs.

We now divide the game sequence depending on the strategy taken by the adversary $\mathcal{A}$. Regardless of $\mathcal{A}$'s strategy, we prove that ϵ_1 is negligible, which in particular implies that ϵ is also negligible. Formally, this is shown in Lemmata B.1 to B.4 provided in their respective subsections below. We first complete the proof of the theorem. Specifically, by combining all the lemmata together, we obtain the following desired bound:

$$\mathsf{Adv}^{\mathsf{AKE}}_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mathcal{A}) \leq \max \left\{ \begin{array}{l} \mu^2\ell^2 \cdot (\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{wKEM}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_1) + \varepsilon_{\mathsf{Ext}}), \\ \mu^2\ell \cdot (\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{wKEM}}(\mathcal{B}_2) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_2) + \varepsilon_{\mathsf{Ext}}) + \mu\ell^2 \cdot \left(\frac{1}{2^{2\chi_{\mathsf{KEM}}}} + \frac{1}{2^{\nu_{\mathsf{KEM}}}}\right), \\ \mu \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B}_3), \\ \mu^2\ell \cdot \left(\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}_4) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_3) + \varepsilon_{\mathsf{Ext}}\right) + \mu\ell^2 \cdot \frac{1}{2^{\chi_{\mathsf{KEM}}}} \end{array} \right\} + \frac{\mu\ell}{2} \cdot (\delta_{\mathsf{SIG}} + 2\delta_{\mathsf{KEM}})$$

Here, the running time of the algorithms $\mathcal{B}_1, \ldots, \mathcal{B}_4$ and $\mathcal{D}_1, \ldots, \mathcal{D}_3$ consist essentially the time required to simulate the security game for $\mathcal{A}$ once, plus a minor number of additional operations. □

It remains to prove Lemmata B.1 to B.4.

Proof of Lemma B.1: Against Type-1 or Type-2 Adversary.

Lemma B.1. *For any* QPT *adversary* $\mathcal{A}$ *using the Type-1 or Type-2 strategy, there exist* QPT *algorithms* $\mathcal{B}_1$ *breaking the* $\mathsf{IND\text{-}CPA}$ *security of* Π_{wKEM} *and* $\mathcal{D}_1$ *breaking the security of PRF* F *such that*

$$\epsilon_1 \leq \mu^2\ell^2 \cdot \left(\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{wKEM}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_1) + \varepsilon_{\mathsf{Ext}}\right).$$

Proof of Lemma B.1. We present the rest of the sequence of games from game G_1.

Game G_2. In this game, at the beginning of the game, $\mathcal{C}$ chooses an initiator oracle $\pi_{\hat{i}}^{\hat{s}}$ and a responder oracle $\pi_{\hat{j}}^{\hat{t}}$ uniformly at random from the $\mu\ell$ oracles. Let $\mathsf{E}_{\mathsf{testO}}$ be the event that the tested oracle is neither $\pi_{\hat{i}}^{\hat{s}}$ nor $\pi_{\hat{j}}^{\hat{t}}$, or $\pi_{\hat{i}}^{\hat{s}}$ and $\pi_{\hat{j}}^{\hat{t}}$ are not partner. Since $\mathsf{E}_{\mathsf{testO}}$ is an efficiently checkable event, $\mathcal{C}$ aborts as soon as it detects that event $\mathsf{E}_{\mathsf{testO}}$ occurs.[26] $\mathcal{C}$ guesses the choice made by $\mathcal{A}$ correctly with probability at least $1/\mu^2\ell^2$, so we have

$$\epsilon_2 \geq \frac{1}{\mu^2\ell^2}\epsilon_1.$$

Game G_3. In this game, we modify the way the initiator oracle $\pi_{\hat{\imath}}^{\hat{s}}$ responds on its second invocation. In particular, when $\pi_{\hat{\imath}}^{\hat{s}}$ is invoked (on the second time) on input $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$, it proceeds as in the previous game except that it uses the key K_T that was generated by the responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ rather than using the key obtained through decrypting C_T. Here, conditioned on $\mathsf{E}_{\mathsf{testO}}$ not occurring, we are guaranteed that the responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ generated C_T by running $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{wKEM.Encap}(\mathsf{ek}_T)$, where ek_T is the encapsulation key that $\pi_{\hat{\imath}}^{\hat{s}}$ outputs on the first invocation. This is because otherwise, the oracles $\pi_{\hat{\imath}}^{\hat{s}}$ and $\pi_{\hat{\jmath}}^{\hat{t}}$ will not be partner oracles. Conditioning on event $\mathsf{E}_{\mathsf{corr}}$ (i.e., decryption failure) not occurring, the two games G_2 and G_3 are identical. Hence,

$$\epsilon_3 = \epsilon_2.$$

Game G_4. In this game, we modify the way the responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ responds. When the responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ is invoked on input ek_T, the game samples a random key $\mathsf{K}_T \leftarrow\!\!\$\, \mathcal{KS}_{\mathsf{wKEM}}$ instead of computing $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{wKEM.Encap}(\mathsf{ek}_T)$. Note that when the initiator oracle $\pi_{\hat{\imath}}^{\hat{s}}$ is invoked (on the second time) on input $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$, it uses this random key K_T. We claim G_3 and G_4 are indistinguishable assuming the $\mathsf{IND\text{-}CPA}$ security of Π_{wKEM}. To prove this, we construct an algorithm $\mathcal{B}_1$ breaking the $\mathsf{IND\text{-}CPA}$ security as follows.

$\mathcal{B}_1$ receives a public parameter $\mathsf{pp}_{\mathsf{wKEM}}$, a public key ek^*, and a challenge $(\mathsf{K}^*, \mathsf{C}^*)$ from its challenger. $\mathcal{B}_1$ sets up the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ using $\mathsf{pp}_{\mathsf{wKEM}}$ and computes $(\mathsf{lpk}_i, \mathsf{lsk}_i)$ for all $i \in [\mu]$ by running the protocol honestly, and samples $(\hat{\imath}, \hat{\jmath}, \hat{s}, \hat{t})$ uniformly random from $[\mu]^2 \times [\ell]^2$. It then invokes $\mathcal{A}$ on the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ and $\{\mathsf{lpk}_i \mid i \in [\mu]\}$ and answers queries made by $\mathcal{A}$ as follows:

- $\mathsf{Send}(i, s, \langle \mathtt{START} : \mathsf{role}, j\rangle)$: If $(i, s, j) = (\hat{\imath}, \hat{s}, \hat{\jmath})$, then $\mathcal{B}_1$ returns ek^* to $\mathcal{A}$ and implicitly sets $\mathsf{state}_i^s := \mathsf{dk}^*$. Otherwise, $\mathcal{B}_1$ responds as in G_4.
- $\mathsf{Send}(j, t, m = \mathsf{ek}_T)$: Let $i := \mathsf{Pid}_j^t$. Depending on the values of (j, t, i), it performs the following:
  - If $(j, t) = (\hat{\jmath}, \hat{t})$ and $i \neq \hat{\imath}$, then $\pi_{\hat{\imath}}^{\hat{s}}$ and $\pi_{\hat{\jmath}}^{\hat{t}}$ cannot be partner oracles. Therefore, since event $\mathsf{E}_{\mathsf{testO}}$ is triggered $\mathcal{B}_1$ aborts.
  - If $(j, t, i) = (\hat{\jmath}, \hat{t}, \hat{\imath})$, then $\mathcal{B}_1$ checks if $\mathsf{ek}_T = \mathsf{ek}^*$. If not, event $\mathsf{E}_{\mathsf{testO}}$ is triggered so it aborts. Otherwise, it proceeds as in G_4 except that it sets $\mathsf{K}_T = \mathsf{K}^*$ and $\mathsf{C}_T = \mathsf{C}^*$ rather than sampling them on its own. It then returns the message $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$.
  - If $(j, t, i) \neq (\hat{\jmath}, \hat{t}, \hat{\imath})$, then $\mathcal{B}_1$ responds as in G_4.
- $\mathsf{Send}(i, s, m = (\mathsf{C}, \mathsf{C}_T, \mathsf{c}))$: Let $j := \mathsf{Pid}_i^s$. Depending on the values of (i, s, j), it performs the following:
  - If $(i, s) = (\hat{\imath}, \hat{s})$ and $j \neq \hat{\jmath}$, then $\pi_{\hat{\imath}}^{\hat{s}}$ and $\pi_{\hat{\jmath}}^{\hat{t}}$ cannot be partner oracles. Therefore, since event $\mathsf{E}_{\mathsf{testO}}$ is triggered $\mathcal{B}_1$ aborts.
  - If $(i, s, j) = (\hat{\imath}, \hat{s}, \hat{\jmath})$, then $\mathcal{B}_1$ checks if $\mathsf{C}_T = \mathsf{C}^*$. If not, event $\mathsf{E}_{\mathsf{testO}}$ is triggered so it aborts. Otherwise, it responds as in G_4.
  - If $(i, s, j) \neq (\hat{\imath}, \hat{s}, \hat{\jmath})$, then $\mathcal{B}_1$ responds as in G_4.
- $\mathsf{RevLTK}(i)$, $\mathsf{RegisterLTK}(i)$, $\mathsf{RevState}(i, s)$, $\mathsf{RevSessKey}(i, s)$: $\mathcal{B}_1$ proceeds as in the previous game. Here, note that since $\mathcal{A}$ follows the Type-1 or Type-2 strategy, $\mathcal{B}_1$ can answer all the $\mathsf{RevState}$-query. Namely, $\mathcal{A}$ never queries $\mathsf{RevState}(\hat{\imath}, \hat{s})$ (i.e., $\mathsf{state}_{\hat{\imath}}^{\hat{s}} := \mathsf{dk}^*$) conditioning on $\mathsf{E}_{\mathsf{testO}}$ not occurring, which is the only query that $\mathcal{B}_1$ cannot answer.

[26] For example, $\mathcal{C}$ can efficiently notice if the two oracles $\pi_{\hat{\imath}}^{\hat{s}}$ and $\pi_{\hat{\jmath}}^{\hat{t}}$ become non-partners even before $\mathcal{A}$ makes a Test-query by checking the input-output of each oracles.

- $\mathsf{Test}(i,s)$: $\mathcal{B}_1$ responds as in G_4. Here, in case $(i,s) \notin \{(\hat{\imath},\hat{s}),(\hat{\jmath},\hat{t})\}$, then event $\mathsf{E}_{\mathsf{testO}}$ is triggered so it aborts.

Finally, if $\mathcal{A}$ outputs a guess b', $\mathcal{B}_1$ outputs b'. It can be checked that $\mathcal{B}_1$ perfectly simulates game G_3 (resp. G_4) to $\mathcal{A}$ when the challenge K^* is the real key (resp. a random key). Thus we have

$$|\Pr[\mathsf{S}_3] - \Pr[\mathsf{S}_4]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{wKEM}}(\mathcal{B}_1).$$

Game G_5. In this game, we modify how the PRF key K_2 is generated by the tested oracle and its partner oracle. Instead of computing $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$, both oracles use the same randomly sampled $\mathsf{K}_2 \leftarrow\!\!\$\, \mathcal{FK}$. Due to the modification we made in the previous game, K_T is chosen uniformly at random from $\mathcal{KS}_{\mathsf{wKEM}}$ so K_T has $\log_2(|\mathcal{KS}_{\mathsf{wKEM}}|) \geq \gamma_{\mathsf{KEM}}$ min-entropy. Then, by the definition of the strong $(\gamma_{\mathsf{KEM}}, \varepsilon_{\mathsf{Ext}})$-extractor Ext, we have

$$|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5]| \leq \varepsilon_{\mathsf{Ext}}.$$

Game G_6. In this game, we modify how the session key k is generated by the tested oracle. Instead of computing $\mathsf{k}\|\tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid})$, the tested oracle (which is either $\pi_{\hat{\imath}}^{\hat{s}}$ or $\pi_{\hat{\jmath}}^{\hat{t}}$ conditioned on event $\mathsf{E}_{\mathsf{testO}}$ not occurring) computes the session key as $\mathsf{k}\|\tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}) \oplus x$, where x is chosen uniformly at random from $\{0,1\}^{\kappa+d}$. Since K_2 is chosen uniformly and hidden from the views of the adversary $\mathcal{A}$, games G_5 and G_6 are indistinguishable by the security of the PRF.[27] In particular, we can construct a PRF adversary $\mathcal{D}_1$ that uses $\mathcal{A}$ as a subroutine such that

$$|\Pr[\mathsf{S}_5] - \Pr[\mathsf{S}_6]| \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_1).$$

In G_6, the session key in the tested oracle is uniformly random. Thus, even an unbounded adversary $\mathcal{A}$ cannot have distinguishing advantages. Therefore, $\Pr[\mathsf{S}_6] = 1/2$. Combining everything together, we have

$$\epsilon_1 \leq \mu^2\ell^2 \cdot \left(\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{wKEM}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_1) + \varepsilon_{\mathsf{Ext}}\right).$$

□

Proof of Lemma B.2: Against Type-3 or Type-4 Adversary.

Lemma B.2. *For any* QPT *adversary $\mathcal{A}$ using the Type-3 or Type-4 strategy, there exist* QPT *algorithms $\mathcal{B}_2$ breaking the* IND-CCA *security of Π_{KEM} and $\mathcal{D}_2$ breaking the security of PRF* F *such that*

$$\epsilon_1 \leq \mu^2\ell \cdot \left(\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}_2) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_2) + \varepsilon_{\mathsf{Ext}}\right) + \mu\ell^2 \cdot \left(\frac{1}{2^{2\chi_{\mathsf{KEM}}}} + \frac{1}{2^{\nu_{\mathsf{KEM}}}}\right).$$

Proof of Lemma B.2. We present the rest of the sequence of games from game G_1.

Game G_2. This game is identical to G_1, except that we add another abort condition. Let $\mathsf{E}_{\mathsf{uniq}}$ be the event that there exists an oracle that has more than one partner oracles. If $\mathsf{E}_{\mathsf{uniq}}$ occurs, then $\mathcal{C}$ aborts. Since G_1 and G_2 proceed identically unless $\mathsf{E}_{\mathsf{uniq}}$ occurs, we have

$$|\epsilon_1 - \epsilon_2| \leq \Pr[\mathsf{E}_{\mathsf{uniq}}].$$

We claim

$$\Pr[\mathsf{E}_{\mathsf{uniq}}] \leq \mu\ell^2 \cdot \left(\frac{1}{2^{2\chi_{\mathsf{KEM}}}} + \frac{1}{2^{\nu_{\mathsf{KEM}}}}\right).$$

Fix $j \in [\mu]$ and consider the set of oracles $S_j = \{\pi_i^s \mid \mathsf{Pid}_i^s = j\}$. For any $\pi_i^s \in S_j$, if there exist two oracles π_j^t and $\pi_j^{t'}$ with $t \neq t' \in [\ell]$ that are partners of π_i^s, then $\mathsf{sid}_i^s = \mathsf{sid}_j^t = \mathsf{sid}_j^{t'}$ holds. We distinguish between the following cases.

[27] We note that for Lemma B.1 we do not require the full power of the PRF; a pseudorandom *generator* (PRG) would have sufficed since the key K_2 is used nowhere else in the game.

Case 1. We first consider the case π_i^s is an initiator and π_j^t and $\pi_j^{t'}$ are responders. Let ek_T be the ephemeral encapsulation key generated by π_i^s. In this case, $\mathsf{E}_{\mathsf{uniq}}$ occurs if the responder oracles π_j^t and $\pi_j^{t'}$ generate the same ciphertext with respect to ek_i and ek_T. Since ek_i and ek_T are independently and honestly generated by the game and each responder oracle is assigned uniform randomness, the probability of a ciphertext collision is upper bounded by $\ell^2/2^{2\chi_{\mathsf{KEM}}}$, where recall χ_{KEM} is the ciphertext min-entropy of Π_{wKEM} and Π_{KEM}. Taking the union bound over all $j \in [\mu]$, we conclude that Case 1 occurs with probability at most $\mu\ell^2/2^{2\chi_{\mathsf{KEM}}}$.

Case 2. We next consider the case π_i^s is a responder and π_j^t and $\pi_j^{t'}$ are initiators. In this case, $\mathsf{E}_{\mathsf{uniq}}$ occurs if the initiator oracles π_j^t and $\pi_j^{t'}$ generate the same ephemeral encapsulation key. Since each initiator oracle samples an encapsulation key independently, the probability of an encapsulation key collision is upper bounded by $\ell^2/2^{\nu_{\mathsf{KEM}}}$, where recall ν_{KEM} is the encapsulation key min-entropy of Π_{wKEM}. Taking the union bound over all $j \in [\mu]$, we conclude that Case 2 occurs with probability at most $\mu\ell^2/2^{\nu_{\mathsf{KEM}}}$.

The claim can be shown by combining the two probabilities from Case 1 and Case 2. In the following games we assume every oracle has a unique partner oracle if it exists.

Game G_3. In this game, at the beginning of the game, $\mathcal{C}$ chooses a random party $P_{\hat{\imath}}$ from the μ parties and a random responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ from the $\mu\ell$ oracles. Let $\mathsf{E}_{\mathsf{testO}}$ be the event where $\neg\mathsf{E}_{\mathsf{testO}}$ denotes the event that either the tested oracle is $\pi_{\hat{\imath}}^{\hat{s}}$ for some $s \in [\ell]$ and its partner oracle is $\pi_{\hat{\jmath}}^{\hat{t}}$, or the tested oracle is $\pi_{\hat{\jmath}}^{\hat{t}}$ and its peer is $P_{\hat{\imath}}$. Since $\mathsf{E}_{\mathsf{testO}}$ is an efficiently checkable event, $\mathcal{C}$ aborts as soon as it detects that event $\mathsf{E}_{\mathsf{testO}}$ occurs. $\mathcal{C}$ guesses the choice made by $\mathcal{A}$ correctly with probability $1/\mu^2\ell$, so we have

$$\epsilon_3 = \frac{1}{\mu^2\ell}\epsilon_2.$$

Game G_4. In this game, we modify the way the initiator oracle $\pi_{\hat{\imath}}^s$ for any $s \in [\ell]$ responds on its second invocation. Let (K, C) be the Π_{KEM} key-ciphertext pair generated by oracle $\pi_{\hat{\jmath}}^{\hat{t}}$. Then, when $\pi_{\hat{\imath}}^s$ is invoked (on the second time) on input $(\mathsf{C}', \mathsf{C}_T, \mathsf{c})$, it first checks if $\mathsf{C}' = \mathsf{C}$. If so, it proceeds as in the previous game except that it uses the key K that was generated by $\pi_{\hat{\jmath}}^{\hat{t}}$ rather than using the key obtained through decrypting C'. Otherwise, if $\mathsf{C}' \neq \mathsf{C}$, then it proceeds exactly as in the previous game. Conditioning on event $\mathsf{E}_{\mathsf{corr}}$ (i.e., decryption failure) not occurring, the two games G_3 and G_4 are identical. Hence,

$$\epsilon_4 = \epsilon_3.$$

Game G_5. In this game, we modify the way the responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ responds. When the responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ is invoked on input ek_T, it samples a random key $\mathsf{K} \leftarrow\!\!\$\, \mathcal{KS}_{\mathsf{KEM}}$ instead of computing $(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_{\hat{\imath}})$. Note that due to the modification we made in the previous game, when the initiator oracle $\pi_{\hat{\imath}}^s$ for any $s \in [\ell]$ is invoked (on the second time) on input $(\mathsf{C}', \mathsf{C}_T, \mathsf{c})$ for $\mathsf{C}' = \mathsf{C}$, it uses the random key K generated by oracle $\pi_{\hat{\jmath}}^{\hat{t}}$. We claim G_4 and G_5 are indistinguishable assuming the $\mathsf{IND\text{-}CCA}$ security of Π_{KEM}. To prove this, we construct an algorithm $\mathcal{B}_2$ breaking the $\mathsf{IND\text{-}CCA}$ security as follows.

$\mathcal{B}_2$ receives a public parameter $\mathsf{pp}_{\mathsf{KEM}}$, a public key ek^*, and a challenge $(\mathsf{K}^*, \mathsf{C}^*)$ from its challenger. $\mathcal{B}_2$ then samples a random $(\hat{\imath}, \hat{\jmath}, \hat{t}) \leftarrow\!\!\$\, [\mu]^2 \times [\ell]$, sets up the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ using $\mathsf{pp}_{\mathsf{KEM}}$, and generates the long-term key pairs as follows. For party $P_{\hat{\imath}}$, $\mathcal{B}_2$ runs $(\mathsf{vk}_{\hat{\imath}}, \mathsf{sk}_{\hat{\imath}}) \leftarrow \mathsf{SIG.KeyGen}(\mathsf{pp}_{\mathsf{SIG}})$ and sets the long-term public key as $\mathsf{lpk}_{\hat{\imath}} := (\mathsf{ek}^*, \mathsf{vk}_{\hat{\imath}})$ and implicitly sets the long-term secret key as $\mathsf{lsk}_{\hat{\imath}} := (\mathsf{dk}^*, \mathsf{sk}_{\hat{\imath}})$, where note that $\mathcal{B}_2$ does not know dk^*. For all the other parties $i \in [\mu \backslash \hat{\imath}]$, $\mathcal{B}_2$ computes the long-term key pairs $(\mathsf{lpk}_i, \mathsf{lsk}_i)$ as in G_5. Finally, $\mathcal{B}_2$ invokes $\mathcal{A}$ on input the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ and $\{\mathsf{lpk}_i \mid i \in [\mu]\}$ and answers the queries made by $\mathcal{A}$ as follows:

- $\mathsf{Send}(i, s, \langle \mathtt{START} : \mathsf{role}, j \rangle)$: $\mathcal{B}_2$ responds as in G_5.
- $\mathsf{Send}(j, t, m = \mathsf{ek}_T)$: Let $i := \mathsf{Pid}_j^t$. Depending on the values of (j, t, i), it performs the following:
    - If $(j, t, i) = (\hat{\jmath}, \hat{t}, \hat{\imath})$, then $\mathcal{B}_2$ responds as in G_5 except that it sets $(\mathsf{K}, \mathsf{C}) := (\mathsf{K}^*, \mathsf{C}^*)$ rather than generating them on its own. It then returns the message $(\mathsf{C}^*, \mathsf{C}_T, \mathsf{c})$.

  - If $(j,t,i) \neq (\hat{\jmath},\hat{t},\hat{\imath})$, then $\mathcal{B}_2$ responds as in G_5.

- $\mathsf{Send}(i,s,m=(\mathsf{C},\mathsf{C}_T,\mathsf{c}))$: Depending on the value of i, it performs the following:
  - If $i=\hat{\imath}$, then $\mathcal{B}_2$ checks if $\mathsf{C}=\mathsf{C}^*$. If so, it responds as in G_5 except that it sets $\mathsf{K}:=\mathsf{K}^*$. Otherwise, if $\mathsf{C}\neq\mathsf{C}^*$, then it queries the decapsulation oracle on C and receives back K'. $\mathcal{B}_2$ then responds as in G_5 except that it sets $\mathsf{K}:=\mathsf{K}'$.
  - If $i\neq\hat{\imath}$, then $\mathcal{B}_2$ responds as in G_5.

- $\mathsf{RevLTK}(i)$, $\mathsf{RegisterLTK}(i)$, $\mathsf{RevState}(i,s)$, $\mathsf{RevSessKey}(i,s)$: $\mathcal{B}_2$ responds as in G_5. Here, note that since $\mathcal{A}$ follows the Type-3 or Type-4 strategy, $\mathcal{B}_2$ can answer all the RevLTK-query. Namely, $\mathcal{A}$ never queries $\mathsf{RevLTK}(\hat{\imath})$ (i.e., $\mathsf{lsk}_{\hat{\imath}}:=(\mathsf{dk}^*,\mathsf{sk}_{\hat{\imath}})$) conditioning on $\mathsf{E}_{\mathsf{testO}}$ not occurring, which is the only query that $\mathcal{B}_2$ cannot answer.

- $\mathsf{Test}(i,s)$: $\mathcal{B}_2$ responds to the query as the definition. Here, in case $i\neq\hat{\imath}$ or $(i,s)\neq(\hat{\jmath},\hat{t})$, then event $\mathsf{E}_{\mathsf{testO}}$ is triggered so it aborts.

If $\mathcal{A}$ outputs a guess b', $\mathcal{B}_2$ outputs b'. It can be checked that $\mathcal{B}_2$ perfectly simulates game G_4 (resp. G_5) to $\mathcal{A}$ when the challenge K^* is the real key (resp. a random key). Thus we have

$$|\Pr[\mathsf{S}_4]-\Pr[\mathsf{S}_5]|\leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}_2).$$

Game G_6. In this game, whenever we need to derive $\mathsf{K}_1^*\leftarrow\mathsf{Ext}_s(\mathsf{K}^*)$, we instead use a uniformly and randomly chosen PRF key $\mathsf{K}_1^*\leftarrow\!\$\,\mathcal{FK}$ (fixed once and for all), where K^* is the KEM key chosen by oracle $\pi_{\hat{\jmath}}^{\hat{t}}$. Due to the modification we made in the previous game, K^* is chosen uniformly at random from $\mathcal{KS}_{\mathsf{KEM}}$ so K has $\log_2(|\mathcal{KS}_{\mathsf{KEM}}|)\geq\gamma_{\mathsf{KEM}}$ min-entropy. Then, by the definition of the strong $(\gamma_{\mathsf{KEM}},\varepsilon_{\mathsf{Ext}})$-extractor Ext, we have

$$|\Pr[\mathsf{S}_5]-\Pr[\mathsf{S}_6]|\leq\varepsilon_{\mathsf{Ext}}.$$

Game G_7. In this game, we sample a random function RF and whenever we need to compute $\mathsf{F}_{\mathsf{K}_1^*}(\mathsf{sid})$ for any sid, we instead compute $\mathsf{RF}(\mathsf{K}_1^*,\mathsf{sid})$. Due to the modification we made in the previous game, K_1^* is sampled uniformly from $\mathcal{FK}$. Therefore, the two games can be easily shown to be indistinguishable assuming the pseudo-randomness of the PRF. In particular, we can construct a PRF adversary $\mathcal{D}_2$ such that

$$|\Pr[\mathsf{S}_6]-\Pr[\mathsf{S}_7]|\leq\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_2).$$

It remains to show that the session key output by the tested oracle in the game G_7 is uniformly random regardless of the challenge bit $b\in\{0,1\}$ chosen by the game. We consider the case where $b=0$ and prove that the honestly generated session key by the tested oracle is distributed uniformly random. First conditioning on event $\mathsf{E}_{\mathsf{testO}}$ not occurring, it must be the case that the tested oracle (and its partner oracle) prepares the session key as $\mathsf{k}^*\|\tilde{k}\leftarrow\mathsf{RF}(\mathsf{K}_1^*,\mathsf{sid}^*)\oplus\mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}^*)$ for some sid^*. That is, K_1^* sampled by the responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ is used to compute the session key. Next, conditioning on event $\mathsf{E}_{\mathsf{uniq}}$ not occurring, the only oracles that share the same sid^* must be the tested oracle and its partner oracle since otherwise it would break the uniqueness of partner oracles. Therefore, we conclude that $\mathsf{RF}(\mathsf{K}_1^*,\mathsf{sid}^*)$ is only used to compute the session key of the tested oracle and its partner oracle. Since the output of RF is distributed uniformly random for different inputs, we conclude that $\Pr[\mathsf{S}_7]=1/2$. Combining all the arguments together, we obtain

$$\epsilon_1\leq\mu^2\ell\cdot\left(\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}_2)+\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_2)+\varepsilon_{\mathsf{Ext}}\right)+\mu\ell^2\cdot\left(\frac{1}{2^{2\chi_{\mathsf{KEM}}}}+\frac{1}{2^{\nu_{\mathsf{KEM}}}}\right).$$

□

Proof of Lemma B.3: Against Type-5 or Type-6 Adversary.

Lemma B.3. *For any* QPT *adversary* $\mathcal{A}$ *using the Type-5 or Type-6 strategy, there exists a* QPT *algorithm* $\mathcal{B}_3$ *breaking the* $\mathsf{EUF\text{-}CMA}$ *of* Π_{SIG} *such that*

$$\epsilon_1 \leq \mu \cdot \mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{B}_3).$$

Proof of Lemma B.3. We present the rest of the sequence of games from game G_1.

Game G_2**.** In this game, at the beginning of the game, $\mathcal{C}$ chooses a party $P_{\hat{\jmath}}$ uniformly at random from the μ parties. Let $\mathsf{E}_{\mathsf{testO}}$ be the event that the peer of the tested oracle is not $P_{\hat{\jmath}}$. If event $\mathsf{E}_{\mathsf{testO}}$ occurs, $\mathcal{C}$ aborts. Since $\mathcal{C}$ guesses the choice made by $\mathcal{A}$ correctly with probability $1/\mu$, we have

$$\epsilon_2 = \frac{1}{\mu}\epsilon_1.$$

Game G_3**.** This game is identical to G_2, except that we add an abort condition. Let S be a list of message-signature pairs that $P_{\hat{\jmath}}$ generated as being a responder oracle. That is, every time $\pi_{\hat{\jmath}}^t$ for some $t \in [\ell]$ is invoked as a responder, it updates the list S by appending the message-signature pair $(\mathsf{sid}_{\hat{\jmath}}^t, \sigma_{\hat{\jmath}}^t)$ that it generates. Then, when an initiator oracle π_i^s for any $(i, s) \in [\mu] \times [\ell]$ is invoked on input $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ from party $P_{\hat{\jmath}}$ (i.e., $\mathsf{Pid}_i^s = \hat{\jmath}$), it first computes sid_i^s and σ as in the previous game and checks if $\mathsf{SIG.Verify}(\mathsf{vk}_{\hat{\jmath}}, \mathsf{sid}_i^s, \sigma) = 1$ and $(\mathsf{sid}_i^s, \sigma) \in S$. If not, the game aborts. Otherwise, it proceeds as in the previous game. We call the event that abort occurs as $\mathsf{E}_{\mathsf{sig}}$. Since the two games are identical until abort, we have

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3]| \leq \Pr[\mathsf{E}_{\mathsf{sig}}].$$

Before, bounding $\Pr[\mathsf{E}_{\mathsf{sig}}]$, we finish the proof of the lemma. We show that no adversary $\mathcal{A}$ following the Type-5 or Type-6 strategy has winning advantage in game G_3, i.e., $\Pr[\mathsf{S}_3] = 1/2$. To see this, first let us assume $\mathcal{A}$ issued $\mathsf{Test}(i^*, s^*)$ and received a key that is not a $\perp$. That is $\pi_{i^*}^{s^*}$ is in the `accept` state. Due to the modification we made in game G_2 and by the definition of the Type-5 or Type-6 strategy, $\pi_{i^*}^{s^*}$ has no partner oracle $\pi_{\hat{\jmath}}^t$ for any $t \in [\ell]$ conditioning on $\mathsf{E}_{\mathsf{testO}}$ not occurring. On the other hand, if $\pi_{i^*}^{s^*}$ is in the `accept` state, then event $\mathsf{E}_{\mathsf{sig}}$ must have not triggered. Consequently, there exists some oracle $\pi_{\hat{\jmath}}^t$ that output $(\mathsf{sid}_{i^*}^{s^*}, \sigma^*)$. Parsing $\mathsf{sid}_{i^*}^{s^*}$ as $P_{i^*} \| P_{\hat{\jmath}} \| \mathsf{lpk}_{i^*} \| \mathsf{lpk}_{\hat{\jmath}} \| \mathsf{ek}_T^* \| \mathsf{C}^* \| \mathsf{C}_T^*$, this implies that $\pi_{\hat{\jmath}}^t$ and $\pi_{i^*}^{s^*}$ are partner oracles. Since this forms a contradiction, $\mathcal{A}$ can only receive $\perp$ when it issues $\mathsf{Test}(i^*, s^*)$. Hence, since the challenge bit b is statistically hidden from $\mathcal{A}$, we have $\Pr[\mathsf{S}_3] = 1/2$.

It remains to bound $\Pr[\mathsf{E}_{\mathsf{sig}}]$. We do this by constructing an algorithm $\mathcal{B}_3$ against the $\mathsf{EUF\text{-}CMA}$ security of Π_{SIG}. The description of $\mathcal{B}_3$ follows: $\mathcal{B}_3$ receives the public parameter $\mathsf{pp}_{\mathsf{SIG}}$ and the challenge verification key vk^*. $\mathcal{B}_3$ sets up the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ as in G_2 using $\mathsf{pp}_{\mathsf{SIG}}$. $\mathcal{B}_3$ then samples $\hat{\jmath}$ randomly from $[\mu]$, runs $(\mathsf{dk}_{\hat{\jmath}}, \mathsf{ek}_{\hat{\jmath}}) \leftarrow \mathsf{KEM.KeyGen}(\mathsf{pp}_{\mathsf{KEM}})$, and sets the long-term public key of party $P_{\hat{\jmath}}$ as $\mathsf{lpk}_{\hat{\jmath}} := (\mathsf{ek}_{\hat{\jmath}}, \mathsf{vk}^*)$. The long-term secret key is implicitly set as $\mathsf{lsk}_{\hat{\jmath}} := (\mathsf{dk}_{\hat{\jmath}}, \mathsf{sk}^*)$, where sk^* is unknown to $\mathcal{B}_3$. For the rest of the parties P_i for $i \in [\mu \backslash \hat{\jmath}]$, $\mathcal{B}_3$ generates $(\mathsf{lpk}_i, \mathsf{lsk}_i)$ as in G_2. Finally, $\mathcal{B}_3$ invokes $\mathcal{A}$ on input the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ and $\{\mathsf{lpk}_i \mid i \in [\mu]\}$ and answers the queries by $\mathcal{A}$ as follows:

- $\mathsf{Send}(i, s, \langle \mathtt{START} : \mathsf{role}, j \rangle)$: $\mathcal{B}_3$ responds as in G_2.
- $\mathsf{Send}(j, t, m = \mathsf{ek}_T)$: Depending on the value of j, it performs the following:
  - If $j = \hat{\jmath}$, then $\mathcal{B}_3$ prepares $\mathsf{sid}_{\hat{\jmath}}^t$ as in G_2, and then sends $\mathsf{sid}_{\hat{\jmath}}^t$ to its signing oracle and receives back a signature σ' for message $\mathsf{sid}_{\hat{\jmath}}^t$ under sk^*. $\mathcal{B}_3$ then responds as in G_2 except that it sets $\sigma := \sigma'$.
  - If $j \neq \hat{\jmath}$, then $\mathcal{B}_3$ responds as in G_2.
- $\mathsf{Send}(i, s, m = (\mathsf{C}, \mathsf{C}_T, \mathsf{c}))$: $\mathcal{B}_3$ responds as in G_2.
- $\mathsf{RevLTK}(i)$, $\mathsf{RegisterLTK}(i)$, $\mathsf{RevState}(i, s)$, $\mathsf{RevSessKey}(i, s)$: $\mathcal{B}_3$ responds as in G_2. Here, note that since $\mathcal{A}$ follows the Type-5 or Type-6 strategy, $\mathcal{B}_3$ can answer all the RevLTK-query. Namely, $\mathcal{A}$ never queries $\mathsf{RevLTK}(\hat{\jmath})$ (i.e., $\mathsf{lsk}_{\hat{\jmath}} := (\mathsf{dk}_{\hat{\jmath}}, \mathsf{sk}^*)$) conditioning on $\mathsf{E}_{\mathsf{testO}}$ not occurring, which is the only query that $\mathcal{B}_3$ cannot answer.

- $\mathsf{Test}(i, s)$: $\mathcal{B}_3$ responds as in G_2. Here, in case $\mathsf{Pid}_i^s \neq \hat{\jmath}$, then event $\mathsf{E}_{\mathsf{testO}}$ is triggered so it aborts.

It is clear that $\mathcal{B}_3$ perfectly simulates the view of game G_2 to $\mathcal{A}$. Below, we analyze the probability that $\mathcal{B}_3$ breaks the $\mathsf{EUF\text{-}CMA}$ security of Π_{SIG} and relate it to $\Pr[\mathsf{E}_{\mathsf{sig}}]$.

We assume $\mathcal{A}$ issues $\mathsf{Test}(i^*, s^*)$. Let the message sent by the initiator oracle $\pi_{i^*}^{s^*}$ be ek_T^* and the message received by $\pi_{i^*}^{s^*}$ be $(\mathsf{C}^*, \mathsf{C}_T^*, \mathsf{c}^*)$. Let σ^* be the signature recovered from c^*. Then, by the definition of the Type-5 or Type-6 strategy and conditioned on $\mathsf{E}_{\mathsf{testO}}$ not occurring, the tested oracle $\pi_{i^*}^{s^*}$ satisfies the following conditions:

- $\mathsf{role}_{i^*}^{s^*} = \mathtt{init}$ and $\mathsf{Pid}_{i^*}^{s^*} = \hat{\jmath}$,
- $\pi_{i^*}^{s^*}$ is in the $\mathtt{accept}$ state. This implies $\mathsf{SIG.Verify}(\mathsf{vk}^*, P_{i^*} \| P_{\hat{\jmath}} \| \mathsf{lpk}_{i^*} \| \mathsf{lpk}_{\hat{\jmath}} \| \mathsf{ek}_T^* \| \mathsf{C}^* \| \mathsf{C}_T^*, \sigma^*) = 1$ holds,
- $P_{\hat{\jmath}}$ is not corrupted,
- $\pi_{i^*}^{s^*}$ has no partner oracles.

Since $\pi_{i^*}^{s^*}$ has no partner oracles, there exists no responder oracle $\pi_{\hat{\jmath}}^t$ that has received ek_T^* from P_{i^*} that sent $(\mathsf{C}^*, \mathsf{C}_T^*)$. In other words, there is no oracle $\pi_{\hat{\jmath}}^t$ that has signed on the message $P_{i^*} \| P_{\hat{\jmath}} \| \mathsf{lpk}_{i^*} \| \mathsf{lpk}_{\hat{\jmath}} \| \mathsf{ek}_T^* \| \mathsf{C}^* \| \mathsf{C}_T^*$. Notice that this is exactly the event $\mathsf{E}_{\mathsf{sig}}$; an initiator oracle $\pi_{i^*}^{s^*}$ receives a signature that was not signed by an oracle $\pi_{\hat{\jmath}}^t$ for any $t \in [\ell]$. Therefore, we have $\Pr[\mathsf{E}_{\mathsf{sig}}] = \mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{B}_3)$. Combining everything together, we conclude

$$\epsilon_1 \leq \mu \cdot \mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{B}_3).$$

□

Proof of Lemma B.4: Against Type-7 or Type-8 Adversary.

Lemma B.4. *For any* QPT *adversary $\mathcal{A}$ using the Type-7 or Type-8 strategy, there exist* QPT *algorithms $\mathcal{B}_4$ breaking the* IND-CCA *security of Π_{KEM} and $\mathcal{D}_3$ breaking the security of PRF* F *such that*

$$\epsilon_1 \leq \mu^2 \ell \cdot \left(\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{B}_4) + \mathsf{Adv}_{\mathsf{F}}^{\mathsf{PRF}}(\mathcal{D}_3) + \varepsilon_{\mathsf{Ext}} \right) + \mu \ell^2 \cdot \frac{1}{2^{\chi_{\mathsf{KEM}}}}.$$

Proof of Lemma B.4. We present the rest of the sequence of games from game G_1.

Game G_2. This game is identical to G_1, except that we add another abort condition. Let $\mathsf{E}_{\mathsf{coll}}$ be the event that there exists two responder oracles π_j^t and $\pi_j^{t'}$ for any $j \in [\mu]$ and $t \neq t' \in [\ell]$ such that they output the same Π_{KEM} ciphertext. That is, there exists two oracles π_j^t and $\pi_j^{t'}$ that output $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ and $(\mathsf{C}', \mathsf{C}'_T, \mathsf{c}')$ such that $\mathsf{C} = \mathsf{C}'$. Here, we only consider the case where Pid_j^t and $\mathsf{Pid}_j^{t'}$ correspond to parties generated by the game (and not parties added by the adversary). If $\mathsf{E}_{\mathsf{coll}}$ occurs, then $\mathcal{C}$ aborts. Since G_1 and G_2 proceed identically unless $\mathsf{E}_{\mathsf{coll}}$ occurs, we have

$$|\epsilon_1 - \epsilon_2| \leq \Pr[\mathsf{E}_{\mathsf{coll}}].$$

We claim

$$\Pr[\mathsf{E}_{\mathsf{coll}}] \leq \mu \ell^2 \cdot \frac{1}{2^{\chi_{\mathsf{KEM}}}}.$$

Since each oracles π_j^t are initialized with uniform random and independent randomness and ek_i is honestly generated, where $i = \mathsf{Pid}_j^t$, each ciphertext C output by oracle π_j^t has χ_{KEM}-min entropy due to the χ_{KEM}-high ciphertext min-entropy of Π_{KEM}. Fixing on one $j \in [\mu]$, the probability of a collision occurring is upper bounded by $\mu^2/2^{\chi_{\mathsf{KEM}}}$. Then, taking the union bound on all the parties, we obtain the claimed bound.

Game G_3. In this game, before starting the game, $\mathcal{C}$ chooses a responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ and a party $P_{\hat{\imath}}$ uniformly at random from $\mu\ell$ oracles and μ parties, respectively. Let $\mathsf{E}_{\mathsf{testO}}$ be the event that the tested oracle is not $\pi_{\hat{\jmath}}^{\hat{t}}$

or the peer of the tested oracle is not $P_{\hat{\imath}}$. Since $\mathsf{E}_{\mathsf{testO}}$ is an efficiently checkable event, $\mathcal{C}$ aborts as soon as it detects that event $\mathsf{E}_{\mathsf{testO}}$ occurs. $\mathcal{C}$ guesses the choice made by $\mathcal{A}$ correctly with probability $1/\mu^2\ell$, so we have

$$\epsilon_3 = \frac{1}{\mu^2\ell}\epsilon_2.$$

The following games G_4 to G_7 is almost identical to those of proof in Lemma B.2. The subtle difference is that the tested oracle does not have a partner oracle. We include the game transition for completeness.

Game G_4. In this game, we modify the way the initiator oracle $\pi_{\hat{\imath}}^s$ for any $s \in [\ell]$ responds on its second invocation. Let (K, C) be the Π_{KEM} key-ciphertext pair generated by oracle $\pi_{\hat{\jmath}}^{\hat{t}}$. Then, when $\pi_{\hat{\imath}}^s$ is invoked (on the second time) on input $(\mathsf{C}', \mathsf{C}_T, \mathsf{c})$, it first checks if $\mathsf{C}' = \mathsf{C}$. If so, it proceeds as in the previous game except that it uses the key K that was generated by $\pi_{\hat{\jmath}}^{\hat{t}}$ rather than using the key obtained through decrypting C'. Otherwise, if $\mathsf{C}' \neq \mathsf{C}$, then it proceeds exactly as in the previous game. Conditioning on event $\mathsf{E}_{\mathsf{corr}}$ (i.e., decryption failure) not occurring, the two games G_3 and G_4 are identical. Hence,

$$\epsilon_4 = \epsilon_3.$$

Game G_5. In this game, we modify the way the responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ responds. When the responder oracle $\pi_{\hat{\jmath}}^{\hat{t}}$ is invoked on input ek_T, it samples a random key $\mathsf{K} \leftarrow\$ \mathcal{KS}_{\mathsf{KEM}}$ instead of computing $(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_{\hat{\imath}})$. Note that due to the modification we made in the previous game, when the initiator oracle $\pi_{\hat{\imath}}^s$ for any $s \in [\ell]$ is invoked (on the second time) on input $(\mathsf{C}', \mathsf{C}_T, \mathsf{c})$ for $\mathsf{C}' = \mathsf{C}$, it uses the random key K generated by oracle $\pi_{\hat{\jmath}}^{\hat{t}}$. We claim G_4 and G_5 are indistinguishable assuming the $\mathsf{IND\text{-}CCA}$ security of Π_{KEM}. To prove this, we construct an algorithm $\mathcal{B}_4$ breaking the $\mathsf{IND\text{-}CCA}$ security as follows.

$\mathcal{B}_4$ receives a public parameter $\mathsf{pp}_{\mathsf{KEM}}$, a public key ek^*, and a challenge $(\mathsf{K}^*, \mathsf{C}^*)$ from its challenger. $\mathcal{B}_4$ then samples a random $(\hat{\imath}, \hat{\jmath}, \hat{t}) \leftarrow\$ [\mu]^2 \times [\ell]$, sets up the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ using $\mathsf{pp}_{\mathsf{KEM}}$, and generates the long-term key pairs as follows. For party $P_{\hat{\imath}}$, $\mathcal{B}_4$ runs $(\mathsf{vk}_{\hat{\imath}}, \mathsf{sk}_{\hat{\imath}}) \leftarrow \mathsf{SIG.KeyGen}(1^\kappa)$ and sets the long-term public key as $\mathsf{lpk}_{\hat{\imath}} := (\mathsf{ek}^*, \mathsf{vk}_{\hat{\imath}})$ and implicitly sets the long-term secret key as $\mathsf{lsk}_{\hat{\imath}} := (\mathsf{dk}^*, \mathsf{sk}_{\hat{\imath}})$, where note that $\mathcal{B}_4$ does not know dk^*. For all the other parties $i \in [\mu \backslash \hat{\imath}]$, $\mathcal{B}_4$ computes the long-term key pairs $(\mathsf{lpk}_i, \mathsf{lsk}_i)$ as in G_5. Finally, $\mathcal{B}_4$ invokes $\mathcal{A}$ on input the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ and $\{\mathsf{lpk}_i \mid i \in [\mu]\}$ and answers the queries made by $\mathcal{A}$ as follows:

- $\mathsf{Send}(i, s, \langle \mathtt{START} : \mathsf{role}, j\rangle)$: $\mathcal{B}_4$ proceeds as in G_5.
- $\mathsf{Send}(j, t, m = \mathsf{ek}_T)$: Let $i := \mathsf{Pid}_j^t$. Depending on the values of (j, t, i), it performs the following:
  - If $(j, t, i) = (\hat{\jmath}, \hat{t}, \hat{\imath})$, then $\mathcal{B}_4$ responds as in G_5 except that it sets $(\mathsf{K}, \mathsf{C}) := (\mathsf{K}^*, \mathsf{C}^*)$ rather than generating them on its own. It then returns the message $(\mathsf{C}^*, \mathsf{C}_T, \mathsf{c})$.
  - If $(j, t, i) \neq (\hat{\jmath}, \hat{t}, \hat{\imath})$, then $\mathcal{B}_4$ responds as in G_5.
- $\mathsf{Send}(i, s, m = (\mathsf{C}, \mathsf{C}_T, \mathsf{c}))$: Depending on the value of i, it performs the following:
  - If $i = \hat{\imath}$, then $\mathcal{B}_4$ checks if $\mathsf{C} = \mathsf{C}^*$. If so, it responds as in G_5 except that it sets $\mathsf{K} := \mathsf{K}^*$. Otherwise, if $\mathsf{C} \neq \mathsf{C}^*$, then it queries the decapsulation oracle on C and receives back K'. $\mathcal{B}_4$ then responds as in G_5 except that it sets $\mathsf{K} := \mathsf{K}'$.
  - If $i \neq \hat{\imath}$, then $\mathcal{B}_4$ responds as in G_5.
- $\mathsf{RevLTK}(i)$, $\mathsf{RegisterLTK}(i)$, $\mathsf{RevState}(i, s)$, $\mathsf{RevSessKey}(i, s)$: $\mathcal{B}_4$ responds as in G_5. Here, note that since $\mathcal{A}$ follows the Type-7 or Type-8 strategy, $\mathcal{B}_4$ can answer all the RevLTK-query. Namely, $\mathcal{A}$ never queries $\mathsf{RevLTK}(\hat{\imath})$ (i.e., $\mathsf{lsk}_{\hat{\imath}} := (\mathsf{dk}^*, \mathsf{sk}_{\hat{\imath}})$) conditioning on $\mathsf{E}_{\mathsf{testO}}$ not occurring, which is the only query that $\mathcal{B}_4$ cannot answer.
- $\mathsf{Test}(i, s)$: $\mathcal{B}_4$ responds to the query as the definition. Here, in case $(i, s) \neq (\hat{\jmath}, \hat{t})$, then event $\mathsf{E}_{\mathsf{testO}}$ is triggered so it aborts.

If $\mathcal{A}$ outputs a guess b', $\mathcal{B}_4$ outputs b'. It can be checked that $\mathcal{B}_4$ perfectly simulates game G_4 (resp. G_5) to $\mathcal{A}$ when the challenge K^* is the real key (resp. a random key). Thus we have

$$|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}_4).$$

Game G_6. In this game, whenever we need to derive $\mathsf{K}_1^* \leftarrow \mathsf{Ext}_s(\mathsf{K}^*)$, we instead use a uniformly and randomly chosen PRF key $\mathsf{K}_1^* \leftarrow_{\$} \mathcal{FK}$ (fixed once and for all), where K^* is the KEM key chosen by oracle $\pi^{\hat{t}}_{\hat{j}}$. Due to the modification we made in the previous game, K^* is chosen uniformly at random from $\mathcal{KS}_{\mathsf{KEM}}$ so K has $\log_2(|\mathcal{KS}_{\mathsf{KEM}}|) \geq \gamma_{\mathsf{KEM}}$ min-entropy. Then, by the definition of the strong $(\gamma_{\mathsf{KEM}}, \varepsilon_{\mathsf{Ext}})$-extractor Ext, we have

$$|\Pr[\mathsf{S}_5] - \Pr[\mathsf{S}_6]| \leq \varepsilon_{\mathsf{Ext}}.$$

Game G_7. In this game, we sample a random function RF and whenever we need to compute $\mathsf{F}_{\mathsf{K}_1^*}(\mathsf{sid})$ for any sid, we instead compute $\mathsf{RF}(\mathsf{K}_1^*, \mathsf{sid})$. Due to the modification we made in the previous game, K_1^* is sampled uniformly from $\mathcal{FK}$. Therefore, the two games can be easily shown to be indistinguishable assuming the pseudo-randomness of the PRF. In particular, we can construct a PRF adversary $\mathcal{D}_3$ such that

$$|\Pr[\mathsf{S}_6] - \Pr[\mathsf{S}_7]| \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_3).$$

The only difference from the proof of Lemma B.2 is how we argue $\Pr[\mathsf{S}_7] = 1/2$. *Details follow.*

It remains to show that the session key outputted by the tested oracle in the game G_7 is uniformly random regardless of the challenge bit $b \in \{0,1\}$ chosen by the game. We consider the case where $b = 0$ and prove that the honestly generated session key by the tested oracle is distributed uniformly random. First conditioning on event $\mathsf{E}_{\mathsf{testO}}$ not occurring, it must be the case that the tested oracle $\pi^{\hat{t}}_{\hat{j}}$ prepares the session key as $\mathsf{k}^* \| \tilde{k} \leftarrow \mathsf{RF}(\mathsf{K}_1^*, \mathsf{sid}^*) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}^*)$ for some sid^*. Here, recall K_1^* is the random PRF key sampled by the oracle $\pi^{\hat{t}}_{\hat{j}}$ (see game G_6). Next, since the tested oracle has no partner oracle (by definition of the Type-7 and Type-8 strategy), there are no oracles π^s_i such that $i \neq i$ that runs $\mathsf{RF}(\mathsf{K}_1^*, \cdot)$ on input sid^*. Moreover, conditioning on event $\mathsf{E}_{\mathsf{coll}}$ not occurring, no oracles $\pi^t_{\hat{i}}$ for $t \neq \hat{t}$ run $\mathsf{RF}(\mathsf{K}_1^*, \cdot)$ on input sid^* as well since $(\mathsf{C}, \mathsf{C}_T)$ output by these oracles must be distinct from what $\pi^{\hat{t}}_{\hat{j}}$ outputs. Therefore, we conclude that $\mathsf{RF}(\mathsf{K}_1^*, \mathsf{sid}^*)$ is only used to compute the session key of the tested oracle and used nowhere else. Since the output of RF is distributed uniformly random for different inputs, we conclude that $\Pr[\mathsf{S}_7] = 1/2$. Combining all the arguments together, we obtain

$$\epsilon_1 \leq \mu^2 \ell \cdot \left(\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}_2) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_2) + \varepsilon_{\mathsf{Ext}} \right) + \mu \ell^2 \cdot \frac{1}{2^{\chi_{\mathsf{KEM}}}}.$$

□

C Omitted Proofs for Deniable Signal-Conforming AKE $\Pi_{\mathsf{SC\text{-}DAKE}}$

In this section, we provide the proofs of the correctness and security of our deniable Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$.

C.1 Correctness of Deniable Signal-Conforming AKE $\Pi_{\mathsf{SC\text{-}DAKE}}$

We prove the correctness of our deniable Signal-Conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$.

Proof of Theorem 6.5. This proof is similar to the proof of Theorem 4.4. It is clear that an initiator oracle and a responder oracle become partners when they execute the protocol faithfully. Moreover, if no correctness error occurs in the underlying KEM schemes and ring signature scheme, the partner oracles compute an identical session key. Since each oracle is assigned to uniform randomness, the probability that a correctness error occurs in one of the underlying schemes is bounded by $\delta_{\mathsf{RS}} + 2\delta_{\mathsf{KEM}}$. Since there are at most $\mu\ell/2$ responder oracles, the AKE protocol is correct except with probability $\mu\ell \cdot (\delta_{\mathsf{RS}} + 2\delta_{\mathsf{KEM}})/2$. □

C.2 Security of Deniable Signal-Conforming AKE $\Pi_{\mathsf{SC\text{-}DAKE}}$

We prove the security of our deniable Signal-Conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$.

Proof of Theorem 6.6. Let $\mathcal{A}$ be an adversary that plays the security game $G_{\Pi_{\mathsf{SC\text{-}DAKE}}}(\mu, \ell)$ with the challenger $\mathcal{C}$ with advantage $\mathsf{Adv}^{\mathsf{AKE}}_{\Pi_{\mathsf{SC\text{-}DAKE}}}(\mathcal{A}) = \epsilon$. The bulk of the proof is identical to the proof of Theorem 4.5 for the (non-deniable) protocol $\Pi_{\mathsf{SC\text{-}AKE}}$. Namely, we divide the strategy that can be taken by $\mathcal{A}$ (listed in Table 1) and we construct an algorithm that breaks one of the underlying assumptions by using such an $\mathcal{A}$ as a subroutine. Formally, we construct seven algorithms $\mathcal{B}_1, \dots, \mathcal{B}_4$ and $\mathcal{D}_1, \dots, \mathcal{D}_3$ satisfying the following:

1. If $\mathcal{A}$ uses the Type-1 (or Type-2) strategy, then $\mathcal{B}_1$ succeeds in breaking the $\mathsf{IND\text{-}CPA}$ security of Π_{wKEM} with advantage $\approx \frac{1}{\mu^2\ell^2}\epsilon$ or $\mathcal{D}_1$ succeeds in breaking the security of PRF F with advantage $\approx \frac{1}{\mu^2\ell^2}\epsilon$.

2. If $\mathcal{A}$ uses the Type-3 (or Type-4) strategy, then $\mathcal{B}_2$ succeeds in breaking the $\mathsf{IND\text{-}CCA}$ security of Π_{KEM} with advantage $\approx \frac{1}{\mu^2\ell}\epsilon$ or $\mathcal{D}_2$ succeeds in breaking the security of PRF F with advantage $\approx \frac{1}{\mu^2\ell}\epsilon$.

3. If $\mathcal{A}$ uses the Type-5 or Type-6 strategy, then $\mathcal{B}_3$ succeeds in breaking the unforgeability of Π_{RS} with advantage $\approx \epsilon$.

4. If $\mathcal{A}$ uses the Type-7 (or Type-8) strategy, then $\mathcal{B}_4$ succeeds in breaking the $\mathsf{IND\text{-}CCA}$ security of Π_{KEM} with advantage $\approx \frac{1}{\mu^2\ell}\epsilon$ or $\mathcal{D}_3$ succeeds in breaking the security of PRF F with advantage $\approx \frac{1}{\mu^2\ell}\epsilon$.

We present a security proof structured as a sequence of games. Without loss of generality, we assume that $\mathcal{A}$ always issues a Test-query. In the following, let S_j denote the event that $b = b'$ occurs in game G_j and let $\epsilon_j := |\Pr[\mathsf{S}_j] - 1/2|$ denote the advantage of the adversary in game G_j. Regardless of the strategy taken by $\mathcal{A}$, all proofs share a common game sequence G_0-G_1 as described below. Although they are identical to those of Theorem 4.5, we provide them for completeness.

Game G_0. This game is identical to the original security game. We thus have

$$\epsilon_0 = \epsilon.$$

Game G_1. This game is identical to G_0, except that we add an abort condition. Let $\mathsf{E}_{\mathsf{corr}}$ be the event that there exist two partner oracles π_i^s and π_j^t that do not agree on the same session key. If $\mathsf{E}_{\mathsf{corr}}$ occurs, then $\mathcal{C}$ aborts (i.e., sets $\mathcal{A}$'s output to be a random bit) at the end of the game.

There are at most $\mu\ell/2$ responder oracles and each oracle is assigned uniform randomness. From Theorem 6.5, the probability of error occurring during the security game is at most $\mu\ell(\delta_{\mathsf{RS}} + 2\delta_{\mathsf{KEM}})/2$. Therefore, $\mathsf{E}_{\mathsf{corr}}$ occurs with probability at most $\mu\ell(\delta_{\mathsf{RS}} + 2\delta_{\mathsf{KEM}})/2$. We thus have

$$|\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_1]| \leq \frac{\mu\ell}{2} \cdot (\delta_{\mathsf{RS}} + 2\delta_{\mathsf{KEM}}).$$

In the following games we assume no decryption error or signature verification error occurs.

We now divide the game sequence depending on the strategy taken by the adversary $\mathcal{A}$. Regardless of $\mathcal{A}$'s strategy, we prove that ϵ_1 is negligible, which in particular implies that ϵ is also negligible. Formally, this is shown in Lemmata C.1 to C.4 provided below. We first complete the proof of the theorem. Specifically, by combining all the lemmata together, we obtain the following desired bound:

$$\mathsf{Adv}^{\mathsf{AKE}}_{\Pi_{\mathsf{SC\text{-}DAKE}}}(\mathcal{A}) \leq \max \left\{ \begin{array}{l} \mu^2\ell^2 \cdot (\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{wKEM}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_1) + \varepsilon_{\mathsf{Ext}}), \\ \mu^2\ell \cdot (\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{wKEM}}(\mathcal{B}_2) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_2) + \varepsilon_{\mathsf{Ext}}) + \mu\ell^2 \cdot \left(\frac{1}{2^{2\chi_{\mathsf{KEM}}}} + \frac{1}{2^{\nu_{\mathsf{KEM}}}}\right), \\ \mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{RS}}(\mathcal{B}_3), \\ \mu^2\ell \cdot \left(\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}_4) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{D}_3) + \varepsilon_{\mathsf{Ext}}\right) + \mu\ell^2 \cdot \frac{1}{2^{\chi_{\mathsf{KEM}}}} \end{array} \right\} + \frac{\mu\ell}{2} \cdot (\delta_{\mathsf{RS}} + 2\delta_{\mathsf{KEM}}).$$

Here, the running time of the algorithms $\mathcal{B}_1, \dots, \mathcal{B}_4$ and $\mathcal{D}_1, \dots, \mathcal{D}_3$ consist essentially the time required to simulate the security game for $\mathcal{A}$ once, plus a minor number of additional operations. □

It remains to prove Lemmata C.1 to C.4. Since the proof of Lemmata C.2 to C.4 is a direct consequence of the proof of the corresponding Lemmata B.1, B.2 and B.4 of Theorem 4.5,[28] we focus on proving Lemma C.1 below.

Lemma C.1. *For any* QPT *adversary* $\mathcal{A}$ *using the Type-5 or Type-6 strategy, there exists a* QPT *algorithm* $\mathcal{B}_3$ *breaking the unforgeability of* Π_{RS} *such that*

$$\epsilon_1 \leq \mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{RS}}(\mathcal{B}_3).$$

Proof of Lemma C.1. We present the rest of the sequence of games from game G_1.

Game G_2. This game is identical to G_1, except that we add an abort condition. Let S_j be a list of message-signature pairs that P_j generated as being a responder oracle. That is, every time π_j^t for some $t \in [\ell]$ is invoked as a responder, it updates the list S_j by appending the message-signature pair $(\mathsf{sid}_j^t, \sigma_j^t)$ that it generates. Then, when an initiator oracle π_i^s for any $(i, s) \in [\mu] \times [\ell]$ is invoked on input $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ from party P_j (i.e., $\mathsf{Pid}_i^s = j$), it first computes sid_i^s and σ as in the previous game and checks if $\mathsf{RS.Verify}(\{\mathsf{vk}_T, \mathsf{vk}_j\}, \mathsf{sid}_i^s, \sigma) = 1$ and $(\mathsf{sid}_i^s, \sigma) \in S_j$. If not, the game aborts. Otherwise, it proceeds as in the previous game. We call the event that abort occurs as $\mathsf{E_{sig}}$. Since the two games are identical until abort, we have

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3]| \leq \Pr[\mathsf{E_{sig}}].$$

Before, bounding $\Pr[\mathsf{E_{sig}}]$, we finish the proof of the lemma. We show that no adversary $\mathcal{A}$ following the Type-5 or Type-6 strategy has winning advantage in game G_2, i.e., $\Pr[\mathsf{S}_2] = 1/2$. To see this, first let us assume $\mathcal{A}$ issued $\mathsf{Test}(i^*, s^*)$ and received a key that is not a $\perp$. In other words, $\pi_{i^*}^{s^*}$ is in the $\mathtt{accept}$ state. By the definition of the Type-5 or Type-6 strategy, $\pi_{i^*}^{s^*}$ has no partner oracle π_j^t for any $(j, t) \in [\mu] \times [\ell]$. On the other hand, if $\pi_{i^*}^{s^*}$ is in the $\mathtt{accept}$ state, then event $\mathsf{E_{sig}}$ must have not triggered. Consequently, there exists some oracle π_j^t that output $(\mathsf{sid}_{i^*}^{s^*}, \sigma^*)$. Parsing $\mathsf{sid}_{i^*}^{s^*}$ as $P_{i^*} \| P_j \| \mathsf{lpk}_{i^*} \| \mathsf{lpk}_j \| \mathsf{ek}_T^* \| \mathsf{vk}_T^* \| \mathsf{C}^* \| \mathsf{C}_T^*$, this implies that π_j^t and $\pi_{i^*}^{s^*}$ are partner oracles. Since this forms a contradiction, $\mathcal{A}$ can only receive $\perp$ when it issues $\mathsf{Test}(i^*, s^*)$. Hence, since the challenge bit b is statistically hidden from $\mathcal{A}$, we have $\Pr[\mathsf{S}_2] = 1/2$.

It remains to bound $\Pr[\mathsf{E_{sig}}]$. We do this by constructing an algorithm $\mathcal{B}_3$ against the unforgeability of Π_{RS}. The description of $\mathcal{B}_3$ follows: $\mathcal{B}_3$ receives the public parameter $\mathsf{pp}_{\mathsf{RS}}$ and $\mu + \mu\ell$ verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_\mu$ and $\mathsf{vk}_1^1, \ldots, \mathsf{vk}_\mu^\ell$. $\mathcal{B}_3$ sets up the public parameter of $\Pi_{\mathsf{SC\text{-}DAKE}}$ as in game G_2 using $\mathsf{pp}_{\mathsf{RS}}$. $\mathcal{B}_3$ then runs $(\mathsf{dk}_i, \mathsf{ek}_i) \leftarrow \mathsf{KEM.KeyGen}(\mathsf{pp}_{\mathsf{KEM}})$ and sets the long-term public key of party P_i as $\mathsf{lpk}_i := (\mathsf{ek}_i, \mathsf{vk}_i)$. The long-term secret key is implicitly set as $\mathsf{lsk}_i := (\mathsf{dk}_i, \mathsf{sk}_i)$, where sk_i is unknown to $\mathcal{B}_3$. Finally, $\mathcal{B}_3$ invokes $\mathcal{A}$ on input the public parameter of $\Pi_{\mathsf{SC\text{-}DAKE}}$ and $\{\mathsf{lpk}_i \mid i \in [\mu]\}$ and answers the queries by $\mathcal{A}$ as follows:

- $\mathsf{Send}(i, s, \langle \mathtt{START} : \mathsf{role}, j \rangle)$: $\mathcal{B}_3$ responds as in G_1 except that it sets $\mathsf{vk}_T := \mathsf{vk}_i^s$.
- $\mathsf{Send}(j, t, m = (\mathsf{ek}_T, \mathsf{vk}_T))$: $\mathcal{B}_3$ responds as in G_1 except that rather than constructing the signature σ on its own, it sends $(\mathsf{sign}, j, \mathsf{sid}_j^t, \{\mathsf{vk}_T, \mathsf{vk}_j\})$ to its signing oracle and uses the signature σ' that it receives.
- $\mathsf{Send}(i, s, m = (\mathsf{C}, \mathsf{C}_T, \mathsf{c}))$: $\mathcal{B}_3$ responds as in G_1.
- $\mathsf{RevLTK}(i)$: $\mathcal{B}_3$ sends $(\mathsf{corrupt}, i)$ to its corruption oracle and receives back a signing key sk_i'. $\mathcal{B}_3$ then sets $\mathsf{sk}_i := \mathsf{sk}_i'$ and returns $\mathsf{lsk}_i = (\mathsf{dk}_i, \mathsf{sk}_i)$.
- $\mathsf{RevState}(i, s)$, $\mathsf{RevSessKey}(i, s)$: $\mathcal{B}_3$ responds as in G_1.
- $\mathsf{Test}(i, s)$: $\mathcal{B}_3$ responds as in G_1.

It is clear that $\mathcal{B}_3$ perfectly simulates the view of game G_2 to $\mathcal{A}$. Below, we analyze the probability that $\mathcal{B}_3$ breaks the unforgeability of Π_{RS} and relate it to $\Pr[\mathsf{E_{sig}}]$.

We assume $\mathcal{A}$ issues $\mathsf{Test}(i^*, s^*)$. Let the message sent by the initiator oracle $\pi_{i^*}^{s^*}$ be $(\mathsf{ek}_T^*, \mathsf{vk}_T^*)$ and the message received by $\pi_{i^*}^{s^*}$ be $(\mathsf{C}^*, \mathsf{C}_T^*, \mathsf{c}^*)$. Let σ^* be the signature recovered from c^*. Then, by the definition of the Type-5 or Type-6 strategy, the tested oracle $\pi_{i^*}^{s^*}$ satisfies the following conditions:

[28] Note that Lemma C.2 (resp. Lemma C.3, Lemma C.4) corresponds to Lemma B.1 (resp. Lemma B.2, Lemma B.4).

- $\mathsf{role}_{i^*}^{s^*} = \mathtt{init}$,
- P_j is not corrupted where $\mathsf{Pid}_{i^*}^{s^*} = j$ and $j \in [\mu]$,
- $\pi_{i^*}^{s^*}$ is in the $\mathtt{accept}$ state. This implies $\mathsf{RS.Verify}(\{\mathsf{vk}_T^*, \mathsf{vk}_j\}, P_{i^*} \| P_j \| \mathsf{lpk}_{i^*} \| \mathsf{lpk}_j \| \mathsf{ek}_T^* \| \mathsf{vk}_T^* \| \mathsf{C}^* \| \mathsf{C}_T^*, \sigma^*) = 1$ holds,
- $\pi_{i^*}^{s^*}$ has no partner oracles.

Since P_j is not corrupted, $\mathcal{A}$ has never queried $\mathsf{RevLTK}(j)$-query. Moreover, since an honest initiator discards sk_T^* on generation, $\mathcal{B}_3$ never uses them for simulation. These two facts imply that $(\mathsf{corrupt}, j)$ and $(\mathsf{corrupt}, (i, T))$ has never been queried, where $(\mathsf{corrupt}, (i, T))$ is a query regarding the verification key $\mathsf{vk}_{i^*}^{s^*}$. In particular, the ring $\{\mathsf{vk}_T^*, \mathsf{vk}_j\}$ consists of non-corrupted verification keys. Moreover, since $\pi_{i^*}^{s^*}$ has no partner oracles, there exists no responder oracle π_j^t that has received $(\mathsf{ek}_T^*, \mathsf{vk}_T^*)$ from P_{i^*} and sent $(\mathsf{C}^*, \mathsf{C}_T^*)$. In other words, there is no oracle π_j^t that has signed on the message $P_{i^*} \| P_j \| \mathsf{lpk}_{i^*} \| \mathsf{lpk}_j \| \mathsf{ek}_T^* \| \mathsf{vk}_T^* \| \mathsf{C}^* \| \mathsf{C}_T^*$. Notice that this is exactly the event $\mathsf{E}_{\mathsf{sig}}$; an initiator oracle $\pi_{i^*}^{s^*}$ receives a signature that was not signed by an oracle π_j^t for any $t \in [\ell]$. Therefore, we have $\Pr[\mathsf{E}_{\mathsf{sig}}] = \mathsf{Adv}_{\mathsf{RS}}^{\mathsf{Unf}}(\mathcal{B}_3)$.

Combining everything together, we conclude

$$\epsilon_1 \leq \mathsf{Adv}_{\mathsf{RS}}^{\mathsf{Unf}}(\mathcal{B}_3).$$

□

For completeness, we state the remaining Lemmata C.2 to C.4 and provide a proof sketch.

Lemma C.2. *For any* QPT *adversary $\mathcal{A}$ using the Type-1 or Type-2 strategy, there exist* QPT *algorithms $\mathcal{B}_1$ breaking the* IND-CPA *security of Π_{wKEM} and $\mathcal{D}_1$ breaking the security of PRF* F *such that*

$$\epsilon_1 \leq \mu^2 \ell^2 \cdot \left(\mathsf{Adv}_{\mathsf{wKEM}}^{\mathsf{IND\text{-}CPA}}(\mathcal{B}_1) + \mathsf{Adv}_{\mathsf{F}}^{\mathsf{PRF}}(\mathcal{D}_1) + \varepsilon_{\mathsf{Ext}} \right).$$

Lemma C.3. *For any* QPT *adversary $\mathcal{A}$ using the Type-3 or Type-4 strategy, there exist* QPT *algorithms $\mathcal{B}_2$ breaking the* IND-CCA *security of Π_{KEM} and $\mathcal{D}_2$ breaking the security of PRF* F *such that*

$$\epsilon_1 \leq \mu^2 \ell \cdot \left(\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{B}_2) + \mathsf{Adv}_{\mathsf{F}}^{\mathsf{PRF}}(\mathcal{D}_2) + \varepsilon_{\mathsf{Ext}} \right) + \mu \ell^2 \cdot \left(\frac{1}{2^{2\chi_{\mathsf{KEM}}}} + \frac{1}{2^{\nu_{\mathsf{KEM}}}} \right).$$

Lemma C.4. *For any* QPT *adversary $\mathcal{A}$ using the Type-7 or Type-8 strategy, there exist* QPT *algorithms $\mathcal{B}_4$ breaking the* IND-CCA *security of Π_{KEM} and $\mathcal{D}_3$ breaking the security of PRF* F *such that*

$$\epsilon_1 \leq \mu^2 \ell \cdot \left(\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{B}_4) + \mathsf{Adv}_{\mathsf{F}}^{\mathsf{PRF}}(\mathcal{D}_3) + \varepsilon_{\mathsf{Ext}} \right) + \mu \ell^2 \cdot \frac{1}{2^{\chi_{\mathsf{KEM}}}}.$$

Proof Sketch of Lemmata C.2 to C.4. The only difference between $\Pi_{\mathsf{SC\text{-}DAKE}}$ and $\Pi_{\mathsf{SC\text{-}AKE}}$ is that the former uses a ring signature and the first message sent by the initiator includes the ephemeral verification key vk_T. However, it can be easily verified that this modification brings no advantage to the adversary following the strategies in the statement. Specifically, the proofs are identical to the proofs of Lemmata B.1, B.3 and B.4.

In slightly more detail, notice the session key derivation step in $\Pi_{\mathsf{SC\text{-}DAKE}}$ is exactly the same as those in $\Pi_{\mathsf{SC\text{-}AKE}}$. Namely, the value of the derived session key is independent of the signature conditioning on the signature being valid. Further notice the proofs of Lemmata B.1, B.3 and B.4 only relies on the security properties of the KEM, PRF, and extractor. That is, the proof does not hinge on the security offered by the signature scheme and this holds even if replace the signature scheme with a ring signature scheme. Here, we note that the validity of the ephemeral ring signature verification key never comes in play in the security proof. Therefore, the proofs of Lemmata B.1, B.3 and B.4 follow. □

Contents