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[4]

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

May 6, 2022

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 works 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 based 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 progressively strengthen it using ring signatures and/or non-interactive zero-knowledge proof systems. 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.

*This is the full version of a preliminary work that appeared in PKC 2021 [49].

The version published by Springer-Verlag is available at https://doi.org/10.1007/s00145-022-09427-1.

Contents

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 of mass-surveillance of communications, people have become more privacy-aware and the demand for SIM has been steadily increasing. While there has been a range of SIM protocols, the Signal protocol [2] is widely regarded as the gold standard. Not only is it used by the Signal app[1], the Signal protocol is also used by WhatsApp, Skype, Facebook Messenger among many others, where the number of active users is well over 2 billion. 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 [64] and the double ratchet protocol [63]. 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 state 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. [26, 27]. 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 [13, 70, 5, 38, 52, 53]. Under these works, our understanding of the double ratchet protocol has much matured. Notably, Alwen et al. [5] 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 initial session key, for instance, by executing a "secure" X3DH protocol.

The X3DH Protocol. In contrast, other than the white paper offered by Signal [64] and those indirectly considered by Cohn-Gordon et al. [26, 27], works focusing on the X3DH protocol seem to be limited. As far as we are aware, there is one recent work that studies the formalization [22] and a few papers that study one of the appealing security properties, known as (off-line) *deniability*, claimed by the X3DH protocol [75, 73, 74].

Brendel et al. [22] abstract the X3DH protocol and provide the first generic construction based on a new primitive they call a *split key encapsulation mechanism* (split 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. [48] 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. [75] study one of the security guarantees widely assumed for the X3DH protocol called (off-line) deniability [64, Section 4.4] and showed that a strong knowledge-type assumption would be necessary to formally prove it. Unger and Goldberg [73, 74] construct several protocols that can be used as drop-in replacements of the X3DH protocol that achieve 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 [17],[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 [64,

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

[2] Although [64, 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 [17], then the X3DH protocol is secure.

[3] Being vulnerable against KCI attacks seems to be intrinsic to on-line deniability [73, 74, 64].

26, 27], 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.
- The X3DH protocol is so far only instantiable from Diffie-Hellman style assumptions [22] and it is unclear whether such assumptions are inherent to the Signal protocol.
- Ideally, similarly to what Alwen et al. [5] 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 (see Section 3). 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) (see Section 4). Similar 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 [23, 55, 45, 59], 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.

We then provide details on how to recast our Signal-conforming AKE protocol into a key agreement protocol similar to what is used in the Signal protocol. We call this the Signal handshake protocol (see Section 5). Unlike standard AKE protocols, the Signal handshake protocol makes several different design choices for efficiency reasons. The most prominent difference is that informally, the Signal handshake protocol reuses the same first message of the AKE protocol for a certain period of time. While this reduces communication and computation complexity and the storage size required by the server, this negatively affects the level of forward secrecy of the underlying Signal-conforming AKE protocol. We discuss in detail the trade-off between security and efficiency incurred when transforming our Signal-conforming AKE protocol into a Signal handshake protocol in Section 5.2.

In addition, we implement a post-quantum Signal handshake protocol in C, building on the open source libraries PQClean and LibTomCrypt (see Section 6). Our implementation [57] is fully generic and can thus be instantiated with a wide range of KEMs and signature schemes. 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

[4] Although the X3DH protocol can naturally be made secure against leakage of session-specific secrets (including randomness generated within the session) by using the generic NAXOS trick, e.g., [59, 45, 56, 79], it typically requires additional computation. Since this negatively affects efficiency, we target AKE protocols without using the NAXOS trick. See Section 1.3 for more detail.

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.

Finally, while our Signal-conforming AKE already provides a weak form of deniability, we show how to progressively strengthen its deniability by using a ring signature instead of a signature scheme and adding a non-interactive zero-knowledge proof system (NIZK) (see Section 7). We propose one protocol that only uses ring signature while only being deniable against semi-honest adversaries. We then add an NIZK on top of this protocol to make it secure even against malicious adversaries. Although our construction seemingly offers (off-line) deniability against malicious adversaries similar to the X3DH protocol [75], 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 [34, 80, 75]. We briefly discuss the efficiency of our Signal-conforming AKE protocol using ring signatures in Remark 7.10.

1.2 Technical Overview

We first review 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.

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 [14, 25, 44, 71].

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 $g^x \in \mathbb{G}$ and its signature σ_{A}[6] to the server 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 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 k_{B} while Alice is offline, he can begin executing the subsequent double ratchet protocol without waiting for Alice to come online.[7] Whenever Alice comes online, she can fetch whatever message Bob sent from the server.

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 person-in-the-middle 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

[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 [64, Section 4.1]).

[6] In the actual protocol [64, 69], XEdDSA is used as the signature scheme, and the same long-term key (a, g^a) is used for both key exchange and signing.

[7] In practice, Bob may initiate the double ratchet protocol using k_{B} and send his message to Alice along with g^y to the server before Alice responds.

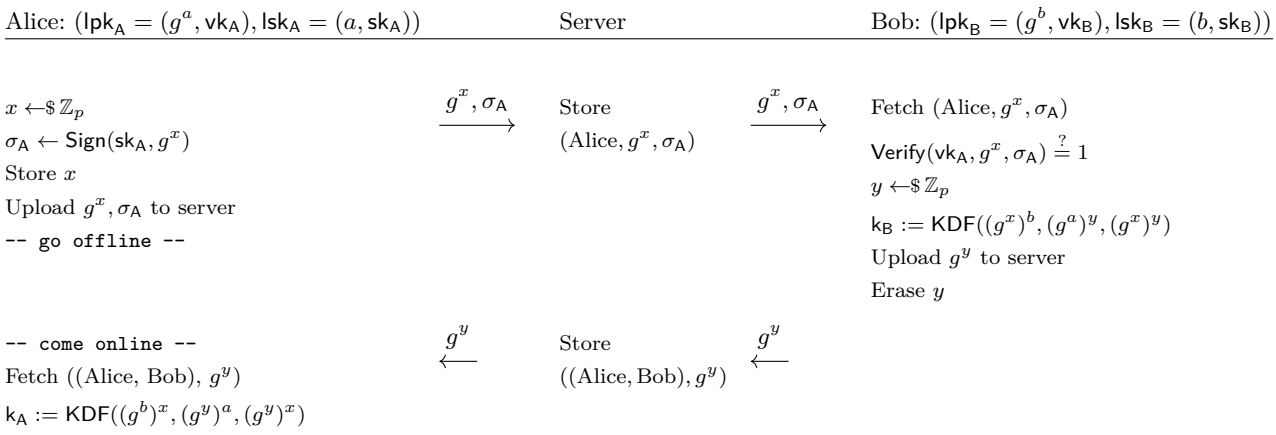


Figure 1: Simplified description of the X3DH Protocol. Alice and Bob have the long-term key pairs $(\mathsf{lpk}_\mathsf{A}, \mathsf{lsk}_\mathsf{A})$ and $(\mathsf{lpk}_\mathsf{B}, \mathsf{lsk}_\mathsf{B})$, respectively. Alice and Bob agree on a session key $\mathsf{k}_\mathsf{A} = \mathsf{k}_\mathsf{B}$, where KDF denotes a key derivation function.

protocol where the initiator message is generated *independently* from the responder. That is, Alice 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 formalizations of [47, 28] 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 [23, 55, 45, 59], which is a strengthening of the Bellare-Rogaway security model [11] considered by [47, 28]. A detailed discussion and comparison between our model 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 is *independent* of the responder. 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. [22] 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. [32] and Kawashima et al. [54]. Considering the community is still in the process of assessing the concrete parameter selection for *standard* CSIDH [19, 68], 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 constructions of AKE protocols [45, 46, 56, 79, 77, 50, 76] that can be instantiated from versatile assumptions, including post-quantum ones, we observe that they are either not Signal-conforming or require the NAXOS trick [59] (see Section 1.3) to be made secure against leakage of session-specific secrets.

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."[8] The construction is simple: let us assume Alice and Bob's long-term keys consist of KEM key pairs $(\mathsf{ek}_\mathsf{A}, \mathsf{dk}_\mathsf{A})$ and $(\mathsf{ek}_\mathsf{B}, \mathsf{dk}_\mathsf{B})$ and signature key pairs $(\mathsf{vk}_\mathsf{A}, \mathsf{sk}_\mathsf{A})$ and $(\mathsf{vk}_\mathsf{B}, \mathsf{sk}_\mathsf{B})$, respectively. The Signal-conforming AKE protocol then starts by Alice (i.e., the initiator) generating a session-specific KEM key $(\mathsf{ek}_T, \mathsf{dk}_T)$, creating a signature $\sigma_\mathsf{A} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_\mathsf{A}, \mathsf{ek}_T)$, and sending $(\mathsf{ek}_T, \sigma_\mathsf{A})$ to Bob (i.e.,

[8] This property has also been called as *post-specified peers* [24] in the context of Internet Key Exchange (IKE) protocols.

the responder). Here, observe that Alice's message does not depend on who she will be communicating with. Bob then verifies the signature and then constructs two ciphertexts: one using Alice's long-term key $(\mathsf{K}_\mathsf{A}, \mathsf{C}_\mathsf{A}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_\mathsf{A})$ 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 $\mathsf{M} := (\mathsf{C}_\mathsf{A}, \mathsf{C}_T)$ as $\sigma_\mathsf{B} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_\mathsf{B}, \mathsf{M})$, 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 k_AKE and a one-time pad key k_OTP, by running a key derivation function on input the random KEM keys $(\mathsf{K}_\mathsf{A}, \mathsf{K}_T)$. Bob then sends $(\mathsf{C}_\mathsf{A}, \mathsf{C}_T, \mathsf{c} := \sigma_\mathsf{B} \oplus \mathsf{k}_\mathsf{OTP})$ to Alice and sets the session key as k_AKE. Here, note that we do not require Alice to hide her signature σ_A since this can only reveal that she was using the Signal app, unlike σ_B that may reveal who Bob was talking to. Once Alice receives the message from Bob, she decrypts the ciphertexts $(\mathsf{C}_\mathsf{A}, \mathsf{C}_T)$, derives the two keys $(\mathsf{k}_\mathsf{AKE}, \mathsf{k}_\mathsf{OPT})$, and checks if $\sigma := \mathsf{c} \oplus \mathsf{k}_\mathsf{OTP}$ is a valid signature of Bob's. If so, she sets the session key as k_AKE. We provide a formal proof and show that our protocol 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., $\mathsf{dk}_\mathsf{A}, \mathsf{dk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, \mathsf{sk}_\mathsf{B}$) 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 [64, Section 4.6].

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. Since we only require a ring of two users, we can use existing efficient post-quantum ring signatures to instantiate this idea. For example, targeting NIST security level 1, we have 2.5 KiB for Raptor [61] (based on NTRU), 4.4 KiB for DualRing [81] (based on M-LWE/SIS), and 3.5 KiB for Calamari [15] (based on CSIDH).

Although the intuition is clear, it turns out that turning this into a formal proof is quite difficult and we observe that for some practical ring signature schemes, this method only provides deniability against *semi-honest* adversaries, which are types of adversaries that follow the protocol description honestly. We provide a concrete attack where a malicious Alice that registers malformed key packages to the server can later (informally) prove to a third-party that Bob was trying to communicate with her even if Bob used a ring signature to sign. We thus propose another protocol that additionally uses NIZKs to make it secure even against malicious adversaries. Similar to all previous works on AKE protocols satisfying a strong flavor of key-compromise security [34, 80] (including the X3DH protocol [75]), the proof of deniability against malicious adversaries relies on a strong knowledge-type assumption.

1.3 Related and Subsequent Work

On the NAXOS trick. The NAXOS trick is a generic/artificial method to boost AKE protocols to be secure with respect to randomness exposure attacks. At a high level, whenever a party is generating a message for its peer, it will not simply use a fresh randomness but extract a randomness by feeding a fresh randomness sampled within that session *and* its long-term secret key into a randomness extractor. Intuitively, this makes the protocol secure against randomness exposure attacks since even if the fresh randomness sampled during in the session is completely exposed, the extracted randomness remains random as long as the long-term secret key is not compromised.

The NAXOS trick was originally proposed by LaMacchia et al. [59] in the random oracle model. Fujioka et al. [45] proposed a new primitive called twisted pseudo-random functions (PRF) in the standard model to mimic its properties and showed how to construct a twisted PRF from any PRF. Alawatugoda et al. [4] also showed that we can mimic the NAXOS trick in the standard model by using a KEM that satisfies a

non-standard notion of pair-generation indistinguishability.

In a two-round AKE protocol, the initiator and responder can both use the NAXOS trick but with different consequences. While the NAXOS trick adds a noticeable overhead in the computation time for the initiator, it adds almost none for the responder. This reason for this asymmetry is that the initiator must perform a wasteful recomputation of (part of) the first message it sent in order to process the second message sent by the responder. For instance, the initiator may generate a public key and a secret key for a KEM using the randomness derived by the NAXOS trick. In order to be secure even when the session-specific secret (i.e., the secret key for the KEM) is exposed, the initiator must securely erase the secret key from its memory after it sends the first message and only store the fresh randomness sampled within the session. Later, when the initiator receives a message from the responder, it must recompute the key generation algorithm again using the randomness derived from the NAXOS trick in order to decrypt the ciphertext included in the message. Since the NAXOS trick assumes that secure erasure of memory is possible and adds a possibly wasteful recomputation step, we simply aim for a two-round AKE protocol that does not use this.

Signal-Conforming AKE Protocol using the NAXOS Trick. Kurosawa and Furukawa [56] generalized the Signed Diffie-Hellman key exchange to work using an IND-CPA secure KEM and a signature scheme. Since the initiator's first message does not depend on the responder's identity, the protocol is Signal-conforming. Unfortunately, the protocol is insecure against standard KCI-attacks and exposure of session-specific secrets (i.e., KEM secret key). Later, Yang et al. [79] showed how to strengthen the security of Kurosawa and Furukawa's protocol by using an IND-CCA secure KEM instead and by further applying the NAXOS trick developed by Alawatugoda et al. [4]. Their protocol results in a secure Signal-conforming AKE protocol that uses the NAXOS trick.

Subsequent Work. After the proceedings version of our paper [49] appeared, Brendel et al. [21] posted on ePrint a post-quantum key exchange protocol that can be used in place of X3DH similar to ours. Very recently, Dobson and Galbraith [36] proposed an X3DH-style protocol tailored to SIDH (SI-X3DH). Details follow.

Brendel et al. [21]. We summarize their contribution and compare it to our protocol.

1. Brendel et al. provide a generic construction of a *deniable* Signal-conforming AKE protocol based on a *designated verifier signature* (DVS) and a KEM. They show that DVS can be instantiated from a ring signature, in which case, their core AKE protocol illustrated in [21, Figure 2] becomes almost identical to our construction in Section 7.2, Figure 5. The following is the main minor differences.

   - We additionally encrypt the signature generated by the ring signature by a one-time pad, while they send it in the clear. This additional layer of encryption offers anonymity with almost no overhead since the transcript no longer leaks information regarding the sender nor the responder to a passive eavesdropper.[9] The same idea can be applied to their protocol as well.
   - They use the NAXOS trick to generate the *second message* sent by the responder. Effectively, the protocol remains secure even if the randomness sampled by the responder (i.e., Bob in Figure 1) is exposed to the adversary. This trick can be used generically to any AKE protocol, including ours for the same net effect. Unfortunately, similar to our protocol, their protocol is insecure when the randomness used to generate the *first message* is exposed. As explained above, applying the NAXOS trick on the first message requires secure erasure of memory and a wasteful recomputation step. Making our protocols secure against randomness exposure of the initiator without using the NAXOS trick remains open. We note that both protocols are secure even if the session-specific secrets (i.e., the secret that is stored by the initiator in order to process the responder's second message) are exposed.

2. In their work, they show that a DVS is implied by a ring signature (for a ring of two users) and left the other implication as an open problem. In particular, this left open the possibility that a generic

[9] To be more precise, we additionally assume that the KEM ciphertext to be anonymous (i.e., indistinguishable from random) as well. This is often the case for standard encryption schemes such as those based on lattices.

construction based on DVS to be more general than those based on ring signatures. In Appendix C, we show that a DVS can be used to construct a ring signature and thus show that a generic construction based on DVS and ring signature are theoretically equivalent.[10]

3. In their work, they depart from prior definitions of simulation-based deniability [34, 37, 80, 73, 74, 75] and introduce a new notion of indistinguishability-based deniability. As the new definition is incomparable to the prior definitions, we provide a detailed comparison between the two definitions in Remark 7.5. Very roughly, their definition considers the setting where all the users honestly follow the protocol and only register valid keys to the server. The adversary (i.e., a non-user) then tries to break deniability of the AKE protocol while given access to the secret keys of all the users. The restriction on users behaving honestly in their definition can loosely be captured by prior definitions of deniability by restricting the adversary to be semi-honest.

4. We provide a concrete attack in Remark 7.11 that breaks deniability of our AKE protocol in Section 7.2, which we show to be secure against semi-honest adversaries. The same attack works against the protocol by Brendel et al. that is proven secure in their new deniability definition. The attack exploits the fact that malicious parties (i.e., non-honest users) can register malicious long-term keys.

5. Finally, we construct an AKE protocol secure even against malicious adversaries in Section 7.3 by additionally using NIZKs and strong knowledge-type assumptions, including a variant of the *plaintext-awareness* (PA) for the KEM scheme [12, 9, 10]. It is an interesting problem if there is a reasonable definition of deniability that suffices to use in the real-world, which also allows for a construction based on more natural assumptions.

Dobson and Galbraith [36]. Dobson and Galbraith [36] proposed an X3DH-style protocol tailored to SIDH (SI-X3DH). Their construction can be seen as a replacement of the DH key exchange in the X3DH protocol with the SIDH key exchange. Their main contribution is showing that SIDH key exchange, which are in general insecure against adaptive attacks, can be used securely by adding a zero-knowledge proof that the long-term SIDH public keys are generated honestly. They explained that the SI-X3DH protocol satisfies the same notions of security as those satisfied by the X3DH protocol. Unlike ours and Brendel et al.'s protocol [21], they do not require a ring signature to argue deniability. They require a strong knowledge-type assumption to prove deniability of the SI-X3DH protocol, which follows similar arguments by [75] that establish the deniability of the X3DH protocol.

Difference From the Conference Version. The preliminary version appeared at the 24th edition of the International Conference on Practice and Theory of Public-Key Cryptography [49]. The main differences between our current version and the preliminary version are as follows:

- In Section 1.3, we added a discussion on what the NAXOS trick is in detail and included more details on AKE protocols that rely on them. We also made it more clear why we focus on a construction that does not rely on the NAXOS trick.
- In Section 1.3, we include a detailed comparison between the work of Brendel et al. [21] that came after our preliminary version [49] became public. We also show in Appendix C that DVS implies a ring signature, which was a problem left open in [21].
- In Section 3.5, we include related works regarding other AKE security models and provide a thorough comparison between our model and the previous models.
- We modify our Signal-conforming AKE protocol in Section 4 so that the initiator further signs its message. The preliminary version did not include this signature. This allows us to prove perfect

[10] We note that the definition of DVS and ring signature come in various flavors. Thus, we only show equivalence under the security properties that Brendel et al. [21] required to construct their AKE protocol. Namely, our implication relies on the fact that their DVS assumes the signature is *publicly* verifiable.

forward secrecy rather than weak forward secrecy. The proof is updated accordingly and is provided in Theorem 4.3.

- The preliminary version did not provide detail on how to turn our Signal-conforming AKE protocol into a full-fledged Signal handshake protocol. In Section 5, we explain how to create a post-quantum Signal handshake protocol and further explain the security implication of reusing the signed-prekey (i.e., the initiator's first message).
- We update the implementation and the associated benchmarks in Section 6.
- The missing security proofs of Theorems 7.7 and 7.14 regarding deniability are added.
- Although the technical details regarding deniability remain the same, in Section 7, we added many new discussions and remarks which we believe make it easier to understand what it means for an AKE protocol to be deniable. For instance, we discuss a concrete attack showing that deniability against only a semi-malicious adversary may not suffice in practice.

2 Preliminaries

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

2.1 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.

2.2 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*

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

where

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

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* IND-ATK *secure for security parameter* κ *if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}ATK}}_{\mathsf{KEM}}(\mathcal{A})$ *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 $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$.

We define plaintext-awareness (PA) for KEM schemes [12, 10] where multiple keys are considered [65]. This property is required to prove the deniability of our authenticated key exchange protocol in Section 7. 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. [34, 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 our deniability proof as the simulator must construct such $\mathcal{E}_{\mathcal{C}}$ given the description of the adversary.

Definition 2.6 (Plaintext-Awareness). *Let* $t = \mathsf{poly}(\kappa)$ *be an integer. We say a KEM scheme* Π_{KEM} *is* plaintext-aware (PA_t-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* $\mathcal{E}_{\mathcal{C}}(\mathtt{query}, (i, \mathsf{C}), \mathsf{rand}_{\mathcal{C}})$*.*[11]

(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* $\mathcal{C}$*.*

2.3 Digital Signatures

Definition 2.7 (Signature Schemes). *A signature scheme with message space* $\mathcal{M}$ *consists of the following four PPT algorithms* $\Pi_{\mathsf{SIG}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

$\mathsf{Setup}(1^\kappa) \to \mathsf{pp}$**:** *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.*

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

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

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

Definition 2.8 ($(1-\delta)$-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 2.9 (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*

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

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* $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A})$ *is negligible for any* QPT *adversary* $\mathcal{A}$*.*

2.4 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.

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

Definition 2.10 (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*

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

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

2.5 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 2.11 (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*

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

2.6 Ring Signatures

Definition 2.12 (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, *$\mathsf{R} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_N\}$, outputs a signature σ.*

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

Definition 2.13 ($(1-\delta)$-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,*

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

Definition 2.14 (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 2.15 (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* $\mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{RS}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$.

2.7 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 the statement and w 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.

Definition 2.16 (NIZK Arguments). *A non-interactive zero-knowledge (NIZK) argument* Π_{NIZK} *for the relation* $\mathcal{R}$ *consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$*.*

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

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

$\mathsf{Verify}(\mathsf{crs}, x, \pi) \to \top$ *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 2.17 (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 2.18 (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 2.19 (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*

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

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.*

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 [47, 28] 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;
- $\Psi_i^s \in \{\perp, \texttt{accept}, \texttt{reject}\}$ 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;
- $\mathsf{role}_i^s \in \{\perp, \texttt{init}, \texttt{resp}\}$ indicates π_i^s's role during the protocol execution.

For each oracle π_i^s, these variables, except the randomness, are initialized to $\perp$. 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 `accept` state, that is, $\mathsf{k}_i^s \neq \perp \iff \Psi_i^s = \texttt{accept}$.

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 [23, 33]. Discussions on other possible choices of the definition for partnering is provided 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 `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 $((i, s), (j, t)) \in S$, $i \leq j$.
- 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*

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

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

3.2 Security Game

We define the security of an AKE protocol Π_{AKE} via a game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. We consider two slightly different variants, each denoted as $G^{\mathsf{weakFS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$ and $G^{\mathsf{FS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$. The former and latter capture a *weakly* and *perfect* forward secure AKE protocol, respectively. Roughly, when the long-term secret key is exposed, the former only ensures the security of past sessions where the adversary did not modify the exchanged messages. In contrast, the latter ensures the security of all past sessions regardless of the adversary actively modifying the exchanged messages. Further details on the difference are provided in Section 3.3. Looking ahead, our main AKE protocol in Section 4 achieves perfect forward secrecy and its variants that satisfy deniability in Section 7 achieve weak forward secrecy.

More formally, 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 proceeds as follows, where the freshness clauses Item 5a and Item 5b is used to define $G^{\mathsf{FS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$ and $G^{\mathsf{weakFS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$, respectively:

Setup: $\mathcal{C}$ first chooses a challenge bit $b \in \{0,1\}$ at random. $\mathcal{C}$ then 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.[12]
- $\mathsf{RevLTK}(i)$: 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 random key depending on the challenge bit b.

- $\mathsf{Test}(i,s)$: If $(i,s) \notin [\mu] \times [\ell]$ or $\Psi_i^s \neq \mathtt{accept}$, $\mathcal{C}$ returns $\bot$. 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*.

[12] 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.

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 $\mathsf{Pid}_i^s = j$[13] 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
   (a) in game $G^{\mathsf{FS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$, P_j is corrupted only after π_i^s finishes the protocol execution.
   (b) in game $G^{\mathsf{weakFS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$, 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 $\mathcal{A}$.

We say that $\mathcal{A}$ wins the game if $b = b'$. The advantage of $\mathcal{A}$ in the security game $G^{\mathtt{xxx}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$ for $\mathtt{xxx} \in \{\mathsf{weakFS}, \mathsf{FS}\}$ is defined as

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

Definition 3.3 (Security of AKE Protocol). *An AKE protocol Π_{AKE} is secure with perfect (resp. weak) forward secrecy if $\mathsf{Adv}^{\mathsf{AKE\text{-}FS}}_{\Pi_{\mathsf{AKE}}}(\mathcal{A})$ (resp. $\mathsf{Adv}^{\mathsf{AKE\text{-}weakFS}}_{\Pi_{\mathsf{AKE}}}(\mathcal{A})$) is negligible for any QPT adversary $\mathcal{A}$.*

3.3 Security Properties

In this section, we explain the security properties captured by our security model. Comparison between other protocols is deferred 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. These capture *key independence*: if the revealed session keys are different from the tested oracle's key, then such session keys must not enable computing the session key of the tested oracle. Note that key independence implies resilience to "no-match attacks" presented by Li and Schäge [60]. 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 [18]: 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 items of information — the long-term secret keys and the session-states of the two parties (where one party is 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. In particular, Item 5a captures *perfect forward secrecy* [35, 23]: the adversary can obtain the long-term secret keys of both parties once the tested oracle finishes the protocol and generates a session key. On the other hand, Item 5b captures *weak forward secrecy* [55]: the adversary can obtain the long-term secret keys of both parties only when it has been passive during the protocol run of the tested oracle. In other words, if an adversary is active (i.e., inject malicious messages) during the protocol execution and further corrupts both long-term secret keys after the oracles evaluate some session keys, then the adversary can test the oracle in $G^{\mathsf{FS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$, while it cannot in $G^{\mathsf{weakFS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$. Another property captured by our model is resistance to *key-compromise impersonation* (KCI)

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

attacks [17]. 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* [23, 55, 59, 45] 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 [47, 28] 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 [64] used by the Signal protocol [2]. 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 key agreement so if the first message depended on the communication partner, then we must wait until they come online to complete the key agreement. Since we cannot send messages without agreeing on a session key, such an AKE protocol where the first message depends on the communication partner cannot be used as an alternative to the X3DH protocol.

We abstract this crucial yet implicit property achieved by the X3DH protocol as *receiver obliviousness*. As noted in Footnote 8, this property has also been called as *post-specified peers* [24] in the context of Internet Key Exchange (IKE) protocols.

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 [32, 54]) can be checked to be receiver oblivious.

3.5 Relation to Other Security Models

In the literature of AKE protocols, many security models have been proposed: the Bellare-Rogaway (BR) model [11], the Canetti-Krawczyk (CK) model [23], the CK+ model [55, 45], the extended CK (eCK) model [59], and variants therein [31, 6, 47, 28, 50, 51]. 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 [11] and its variants (e.g., [6, 47, 28])[14], 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 [23], the session-state can depend arbitrarily 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

[14] We note that the subsequent variants differ from the original BR model [11] as they also model forward secrecy and KCI attacks.

say the AKE protocol is secure with respect to the session-state defined by f.[15] The eCK model [59] and the CK+ model [55, 45] made the CK model more accessible by only considering a specific but natural set of functions.[16] 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 rand_{i*}^{s*} of the tested oracle. As Cremers [29, 30] 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 [50, 51] 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 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 is 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 [11, 55, 45, 59, 31, 6, 28, 50, 51]. 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 [60] 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 on session-identifiers [23, 33]. 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 [60] proposed another method to define partnering called *original-key partnering*. This has been used in [47]. 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 queries. 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, [6, 47, 28, 51] consider the *multi-challenge* setting, where an adversary is allowed to make multiple Test-queries.

Remark 3.5 (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

[15] 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.

[16] 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.

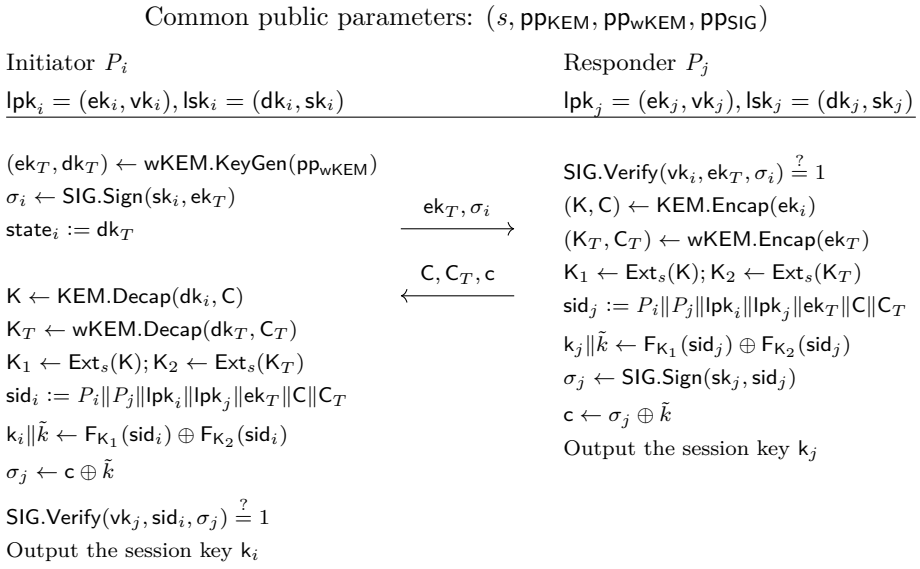


Figure 2: Our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$.

minor issue since we can always add a key confirmation step to enhance an implicit authentication AKE protocol into an explicit one [78, 28, 33]. For instance, we can send an encrypted message or a MAC tag under the established session key to check if the peer computed the same key without compromising security. In the context of Signal, the double ratchet protocol that comes after the X3DH protocol can be viewed as adding an explicit authentication step.

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 to construct a Signal's initial key agreement (Signal handshake) protocol such as 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 IND-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 IND-CPA secure (and not IND-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 EUF-CMA secure and $(1 - \delta_{\mathsf{SIG}})$-correctness. We denote d as the bit length of the signature generated by SIG.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 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 $\mathcal{S}$ for the strong randomness extractor, 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 $\sigma_i \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_i, \mathsf{ek}_T)$. Then it sends $(\mathsf{ek}_T, \sigma_i)$ to party P_j. P_i stores the ephemeral decapsulation key dk_T as the session-state, i.e., $\mathsf{state}_i^s := \mathsf{dk}_T$.[17]

2. Party P_j sets $\mathsf{Pid}_j^t := i$ and $\mathsf{role}_j^t := \mathtt{resp}$. Upon receiving $(\mathsf{ek}_T, \sigma_i)$, P_j first checks whether $\mathsf{SIG.Verify}(\mathsf{vk}_i, \mathsf{ek}_T, \sigma_i) = 1$ holds. If not, P_j sets $(\Psi_j, \mathsf{k}_j^t, \mathsf{state}_j) := (\mathtt{reject}, \bot, \bot)$ and stops. Otherwise, it 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 $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K})$ and $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$. 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., $\mathsf{state}_j^t = \bot$.

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 $\mathsf{k}_i \in \{0,1\}^\kappa$ and $\tilde{k} \in \{0,1\}^d$. P_i then decrypts $\sigma \leftarrow \mathsf{c} \oplus \tilde{k}$ and checks whether $\mathsf{SIG.Verify}(\mathsf{vk}_j, \mathsf{sid}_i^s, \sigma) = 1$ holds. If not, P_i sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{reject}, \bot, \bot)$ and stops. Otherwise, it sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{accept}, \mathsf{k}_i, \bot)$. Here, note that P_i deletes the session-state $\mathsf{state}_i^s = \mathsf{dk}_T$ at the end of the key exchange.

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 erased after 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 need to store any session-specific information. That is, all states can be erased as soon as the session key is computed.

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

Theorem 4.2 (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 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 $\delta_{\mathsf{SIG}} + 2\delta_{\mathsf{KEM}}$. 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 4.3 (Security of $\Pi_{\mathsf{SC\text{-}AKE}}$). *For any QPT adversary $\mathcal{A}$ that plays the game $G^{\mathsf{FS}}_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mu, \ell)$ with μ parties that establishes at most ℓ sessions per party, there exist QPT algorithms $\mathcal{B}_1$ breaking the IND-CPA*

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

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.

security of Π_{wKEM}, $\mathcal{B}_2$ *breaking the* IND-CCA *security of* Π_{KEM}, $\mathcal{B}_3$ *breaking the* EUF-CMA *security of* Π_{SIG}, *and* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *breaking the security of PRF* F *such that*

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

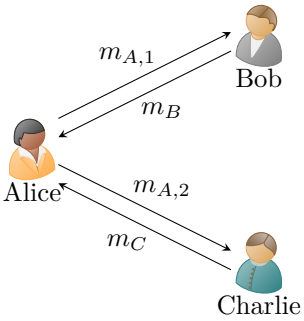
where ν_{KEM} *(resp.* χ_{KEM}*) is the encapsulation key (resp. ciphertext) min-entropy of* Π_{wKEM} *and* Π_{KEM}*. The running time of* $\mathcal{B}_1$*,* $\mathcal{B}_2$*,* $\mathcal{B}_3$*,* $\mathcal{D}_1$*, and* $\mathcal{D}_2$ *are about that of* $\mathcal{A}$*.*

The full proof of Theorem 4.3 can be found in Appendix A. Here, we provide an overview of the proof.

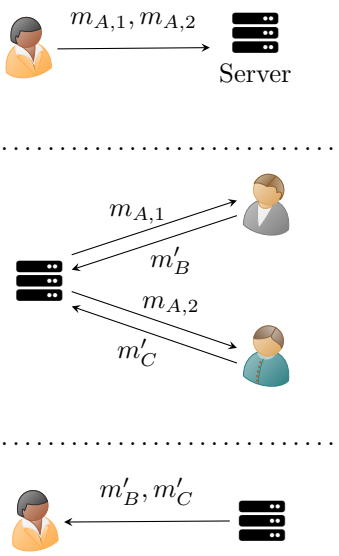
Proof sketch. Let $\mathcal{A}$ be an adversary that plays the security game $G^{\mathsf{FS}}_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mu,\ell)$. We distinguish between all possible strategies that can be taken by $\mathcal{A}$. Specifically, $\mathcal{A}$'s strategy can be divided into the eight types of strategies 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 case: (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) and (b), the session key is informally protected by the security properties of KEM, PRF, and randomness extractor Ext. In case (a), since the ephemeral decapsulation key dk_T is not revealed, K_T is indistinguishable from a random key due to the IND-CPA security of Π_{wKEM}. On the other hand, in case (b), since the initiator's decapsulation key $\mathsf{dk}_{\mathtt{init}}$ is not revealed, K is indistinguishable from a random key due to the IND-CCA security of Π_{KEM}. Here, we require IND-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 both 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.

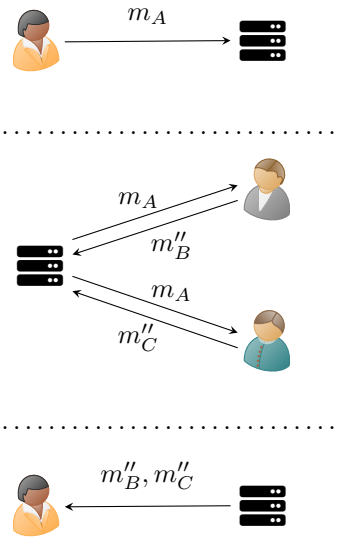
In cases (c) and (d), the session key is informally protected by the security property of the signature scheme. More concretely, in both cases, the peer's signing key sk of the tested oracle is not revealed when the tested oracle runs the protocol. Thus, due to the EUF-CMA security of Π_{SIG}, $\mathcal{A}$ cannot forge the signature for the session-identifier of the tested oracle $\mathsf{sid}_{\mathsf{test}}$ (in case (c)) or for the ephemeral encapsulation key ek_T (in



(a) AKE protocol



(b) Signal handshake as secure as AKE protocol but with large overhead (not used by the Signal app)



(c) Signal handshake slightly weaker compared to AKE protocol but with small overhead (used by the Signal app)

Figure 3: Comparison of the message flow of an AKE protocol (left) and that of a Signal handshake (center and right). The center protocol is more secure than the right protocol, while the right protocol is storage and bandwidth efficient compared to the center. The implemented Signal protocol uses the right protocol.

case (d)). In addition, since the tested oracle has no partner oracles, no oracle ever signs $\mathsf{sid}_{\mathsf{test}}$ (in case (c)) or ek_T (in case (d)). Therefore, combining these two facts, we conclude that the tested oracle cannot be in the `accept` state unless $\mathcal{A}$ breaks the signature scheme. In other words, when $\mathcal{A}$ issues Test-query, the tested oracle always returns $\perp$. Thus, the session key of the tested oracle is hidden from $\mathcal{A}$. □

5 Post-Quantum Signal Handshake

In this section, we provide a concrete discussion on how to turn our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ into a post-quantum Signal handshake protocol. Our protocol can be used as a simple drop-in for the current X3DH protocol as a post-quantum secure replacement.

5.1 Signal Handshake From Signal-conforming AKE protocol

We first explain how to obtain a Signal handshake (i.e., Signal's initial key agreement protocol) from a Signal-conforming AKE protocol. Figure 3a depicts the message flow in AKE protocols. If Alice wants to share session keys with Bob and Charlie respectively, she sends the first message $m_{A,1}$ (resp. $m_{A,2}$) to Bob (resp. Charlie). On receiving the message, Bob and Charlie return the second message m_B and m_C to Alice respectively, and she derives the session keys. In AKE protocols, each party communicates synchronously and the communicating parties are online at the same time.

In contrast, in secure messaging, parties are not always online, and even more, the communicating partner may be unknown at the time of registration. Therefore, we need a mechanism that allows parties to start communication even when the partners are offline and undefined. Signal handshake realizes an asynchronous and anonymous communication by using a possibly untrusted server (see Figure 3b). In its most secure version (which is not used by the Signal app), if Alice anticipates communicating with at most two parties,

she first uploads two first messages $m_{A,1}$ and $m_{A,2}$ of the AKE protocol to the server and goes offline. She will replenish the first messages on the server periodically so that the server does not use up all the first messages. Since Alice does not know who will use the first messages in this case, only Signal-conforming AKE protocols (i.e., two-round and the first message can be generated independently of the responder) can be used for the Signal handshake. Next, if Bob wants to share a session key with Alice, he accesses the server and receives the unused $m_{A,1}$. Then, Bob computes the session key and uploads the second message m'_B. If Charlie also exchanges a session key with Alice, he executes the protocol in the same way as Bob, using a different unused first message $m_{A,2}$. Finally, when Alice comes online, she downloads the second messages m'_B and m'_C from the server and derives the session keys between Bob and Charlie. In this way, parties can exchange session keys asynchronously. Moreover, the Signal handshake in Figure 3b can be shown to be as secure as the underlying AKE protocol. This is because we can view the server as a person-in-the-middle adversary in AKE protocols, and AKE protocols are defined to be secure against such adversaries.

Although secure, the downside of this approach is that Alice must upload as many first messages as the number of parties she anticipates communicating with. To reduce storage overhead, the Signal protocol reuses the first message for multiple key exchange sessions (see Figure 3c). Alice uploads one first message m_A to the server, and periodically updates it, e.g., once a week or once a month. Bob and Charlie exchange session keys with Alice using the same first message m_A. Effectively, both the party's and server's storage overhead are reduced. The downside of reusing the first message is that it may make the protocol less secure compared to the underlying AKE protocol with the cost of better storage size. For example, if an adversary obtains the randomness used to generate m_A, depending on the underlying AKE protocol, it may expose both session keys exchanged between Bob and Charlie. In contrast, if the first message was not reused, then the security of the AKE protocol guarantees that an adversary can recover only the session key that used m_A. A more detailed discussion on the side effect of reusing the first message in our Signal handshake is provided next.

5.2 Details of Our Post-Quantum Signal Handshake

We provide the details of our post-quantum Signal handshake based on our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$. Unlike the X3DH protocol, our protocol can be made post-quantum by choosing appropriate post-quantum building blocks. The protocol description is presented in Figure 4.

As explained in the previous section, parties communicate with each other with the help of the server and reuse the first messages of the AKE protocol for a certain interval (see Figure 3c). First, Alice and Bob generate their long-term keys $\mathsf{lpk_A}$ and $\mathsf{lpk_B}$, respectively, and register them to the server (see the top of Figure 4). Note that parties upload their long-term keys only once. As in the Signal *app*, we assume the validity of the long-term keys are checked between the parties by some "out-of-bound" authentication mechanism (see [64, Section 4.1]). Next, Alice uploads a first message of the AKE protocol: an encapsulation key ek_T (called *signed pre-key* in the Signal white paper [64]) along with a signature σ_{A} (called *pre-key signature* in the Signal white paper [64]) to the server (see the center of Figure 4)[18]. The signed pre-key is reused for multiple key exchange sessions and is updated at some interval (e.g. once a week or once a month). Finally, when Bob wants to communicate with Alice, he accesses the server and downloads Alice's long-term key and signed pre-key $(\mathsf{lpk_A}, \mathsf{ek}_T, \sigma_{\mathsf{A}})$. Then, he runs the AKE protocol of the responder's part and uploads the response message $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ to the server. Finally, when Alice comes online, she downloads the long-term key and the response message from Bob and derives the shared initial secret (see the bottom of Figure 4). It is clear that if the signed pre-keys are not reused, then the protocol is secure as the underlying Signal-conforming AKE protocol.

Let us discuss the security implication of reusing signed pre-keys. It is clear that it *seems* insecure to reuse the signed pre-keys compared to not reusing them. The question is, to what extent are they insecure? In our post-quantum Signal handshake, multiple session keys are exchanged using the same pair of long-term key and signed pre-key. We first argue that, if either the long-term key or the signed pre-key are not leaked, then the exchanged keys remain secure. Our security model (defined in Section 3) guarantees that the session key of oracle π is secure even if an adversary forwards the first message, i.e., a signed pre-key and pre-key signature,

[18] Unlike in the figure, the signed pre-key and pre-key signature are uploaded by all the parties and not only by Alice.

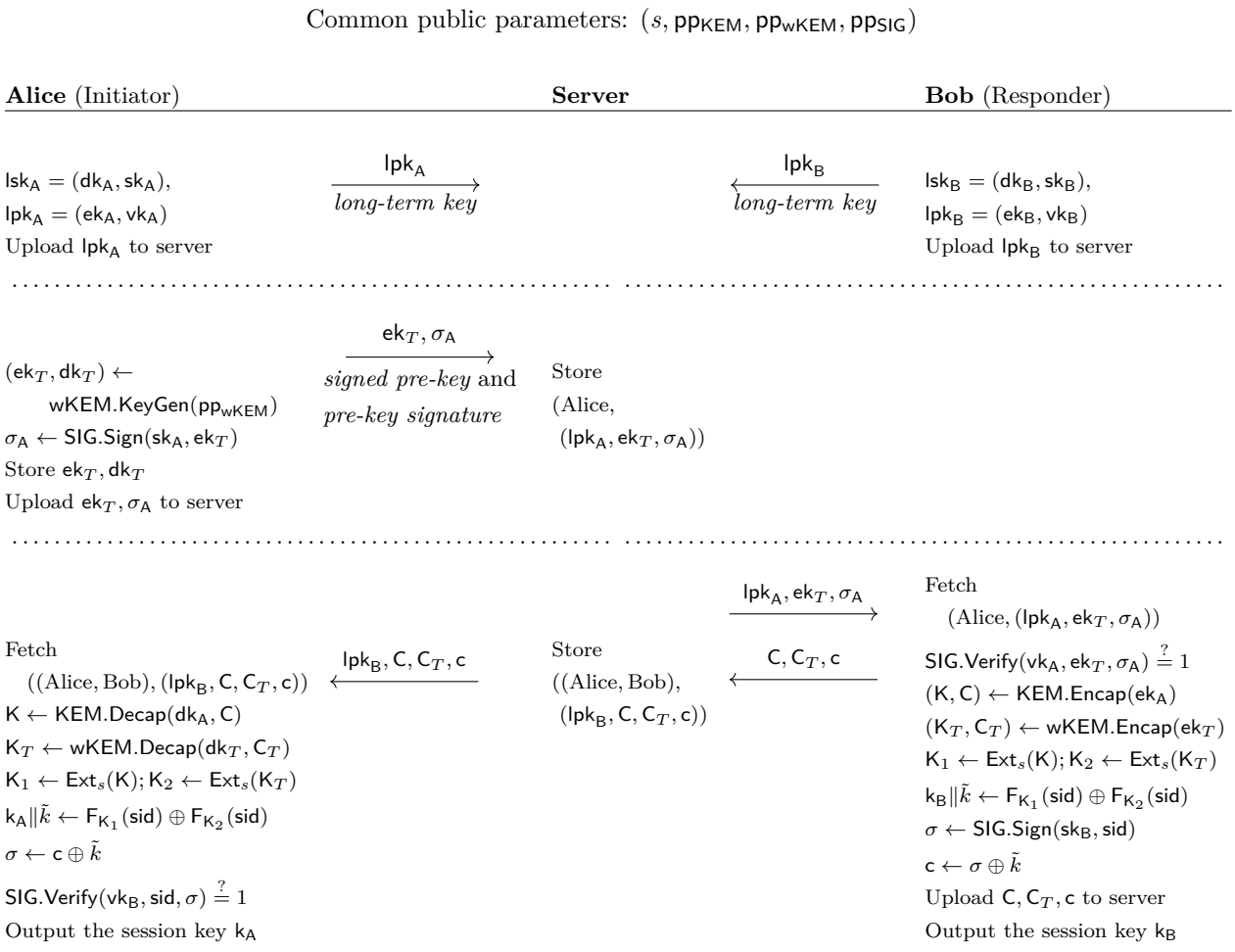


Figure 4: Post-quantum Signal handshake protocol based on our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ that reuses the same first message (i.e., signed pre-key and pre-key signature). The session identifier is defined as $\mathsf{sid} := \mathsf{A} \| \mathsf{B} \| \mathsf{lpk}_\mathsf{A} \| \mathsf{lpk}_\mathsf{B} \| \mathsf{ek}_T \| \mathsf{C} \| \mathsf{C}_T$. Alice and Bob only need to upload their long-term (public) keys to the server once, and the signed pre-key and pre-key signature are reused by multiple responders throughout some interval.

generated by oracle π, to multiple responder oracles and obtains at most one of the two KEM keys include in the long-term key or the signed pre-key used by π. This attack captures the scenario where a party sends the same first message to multiple responders, i.e., reuses its signed pre-key. Therefore, our post-quantum Signal handshake is as secure as a variant that never reuses a signed pre-key (i.e., our Signal-conforming AKE protocol) as long as one of the long-term key or the signed pre-key are not leaked. The distinction between our post-quantum Signal handshake and our Signal-conforming AKE protocol becomes clear when both the long-term key and the signed pre-key are leaked. Observe that once both keys are leaked, the adversary can compute *all* session keys that were exchanged using them. In other words, the number of session keys that are compromised is the same as the number of times the signed pre-key was reused. Thus, the Signal handshake is less secure than the underlying AKE protocol since the number of session keys that are exposed is larger when both the long-term key and the signed pre-key are leaked. To mitigate the number of exposed session keys, those parties looking for better security can use signed pre-keys only once or add a so-called *one-time pre-key* in the first message (i.e., an additional non-signed one-time KEM key). This is exactly what the Signal app does. In this case, even if all the KEM keys used in a specific session are leaked, an adversary can not compute the session keys of the other sessions since a different KEM key is used for each session. Thus, this mitigation can be used to enhance the security in a scenario where the long-term key and the signed pre-key are both exposed but the one-time pre-key is not.

6 Instantiating Post-Quantum Signal Handshake

In this section, we present the implementation details of our post-quantum Signal handshake protocol presented in Figure 4. We take existing implementations of post-quantum KEMs and signature schemes submitted for the NIST PQC standardization. To instantiate our Signal handshake, 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 20 variants of KEM and 14 variants of signature schemes, we can create at least 93 different instantiations[19] of post-quantum Signal handshake protocols. The provided implementation simulates post-quantum, weakly deniable authenticated key exchange between two parties. We study the efficiency of our instantiations through two metrics — the total amount of data exchanged between parties and run-time performance. Our implementation [57] is available in the form of open-source software.

6.1 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/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 [43],
- 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 [7, 8],
- b: equals the security level of the underlying post-quantum KEM scheme, where $b \in \{128, 192, 256\}$,
- d: equals the byte length of the signature generated by the post-quantum signature scheme Π_{SIG},
- Π_{KEM}, Π_{wKEM}, Π_{SIG}: the implementation uses pairs of KEM and signature schemes. The list of the schemes used can be found in Table 2. 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	RAINBOW, FALCON, DILITHIUM SPHINCS†
3	SABER,NTRU‡, CLASSIC-MCELIECE, KYBER, HQC, FRODOKEM	DILITHIUM, RAINBOW SPHINCS†
5	SABER, CLASSIC-MCELIECE, NTRU, KYBER FRODOKEM, HQC	FALCON, RAINBOW,DILITHIUM SPHINCS†

Table 2: Considered KEM and signature schemes under NIST security level 1, 3, and 5. †: Use two SPHINCS instantiations with different hash functions. ‡: Use two NTRU instantiations with different parameter sets.

At a high level, the implementation is split into 4 main parts. A setup phase, where both parties perform long-term key generation and initialization of required during memory benchmarking. The session establishment phase implements an initiator's signed pre-key generation (the `offer` function), the responder's session key generation (`accept` function), and initiator's session key generation (`finalize` function), which finalizes session establishment resulting in session key. To evaluate the cost of our post-quantum Signal handshake, we instantiate the protocol with KEM and signature schemes from Table 2.

The concrete implementation of post-quantum schemes is provided by *PQ Crypto Catalog* library [58], which is a collection of implementations submitted to NIST PQC standardization process. We also use *LibTomCrypt* library [1] which provides an implementation of the building blocks HMAC, HKDF and SHA-256. We note that we use portable `C` code implementations of schemes, which do not include platform specific optimizations. 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 *PQ Crypto Catalog* library does not provide hardware-assisted optimizations for all schemes, hence enabling those optimizations only for some algorithms would result in unfair comparison.

6.2 Efficiency Analysis

In this subsection, we provide an assessment of the costs related to running the concrete instantiation of the post-quantum Signal handshake.

To properly assess the cost, we modeled a scenario according to Figure 4. Namely, two parties try to establish a session key. Alice (the initiator) and Bob (the responder) generate and make their long-term public keys ($\mathsf{lpk_A}$, $\mathsf{lpk_B}$) available to others. Alice then generates her *signed pre-key* ek_T and creates her *pre-key signature* σ_A by signing it. The pair (ek_T, σ_A) is also uploaded to the publicly accessible server. Bob retrieves the pre-key bundle $(\mathsf{lpk_A}, \mathsf{ek}_T, \sigma_\mathsf{A})$ and uses it to perform his part of the session establishment. Namely, Bob generates the triple $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ and makes it available for Alice to download from the server. Once Alice comes online, she downloads the session initialization bundle from Bob together with his long-term key, $\mathsf{lpk_B}$. She then finalizes the process by computing the session key on her side. Note that in the Signal protocol, long-term public keys lpk are fetched from the server. Parties do not store the keys lpk corresponding to those that they have not communicated with before.[20]

We provide three metrics:

- Data transfer cost: the amount of data exchanged when two parties establish a session key.
- Storage cost: the amount of data that needs to be stored on the server to allow a session establishment between parties.

[19] In Table 2, pairing KEMs and signatures schemes with the same NIST security level yields $7 \times 5 + 7 \times 4 + 6 \times 5 = 93$ distinct combinations (some schemes offer multiple instantiations at a given NIST level).

[20] The X3DH protocol assumes the parties authenticate the long-term public keys through some authenticated channel [64, Section 4.1].

- Computational cost: the number of CPU cycles spent in computation during session establishment by both parties.

Cost analysis for each metric is provided separately.

Data Transfer Cost. Table 3 provides the selected results for Round 3 candidates of the NIST PQC standardization process.[21] The lpk column contains the byte size of a long-term key. The following four columns contain the byte size of the data exchanged by the initiator, the server and the responder during a session key establishment (as per Figure 4). Finally, the column **Total** contains the total size of data exchanged between Alice and Bob.

From Table 3, we can conclude that the transfer cost for Falcon512 paired with SIKEp434 is the order of magnitude lower than in the case of the other two pairs. Also, the small size of the Falcon public key and signature size makes it an attractive choice for the signature scheme in the case of that particular application.

Note that the long-term public keys (lpk) are uploaded to the server only once by each party (initiator and responder), hence the cost of uploading them is probably negligible for most applications. To further minimize the transfer cost, some implementations may decide to use caching mechanisms, meaning long-term keys are downloaded only once and cached locally. In this case, the validity of the key may be checked by hashing the lpk_A at both sides and comparing the hash values. In this case, Bob sends a hash of cached lpk_A when requesting the pre-key bundle, the server compares hashes and depending on the result of such comparison sends a response either with long-term keys or without.

Remark 6.1 (Note on Low Quality Network Links). We anticipate the Signal handshake 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 that 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, an increased packet loss may negatively impact session establishment time (see [66]). For example, a scheme instantiated with Falcon512 paired with Saber Light requires exchange of $npacks = 7$ packets over the network, where instantiation with SPHINCS-SHAKE256-128f-simple paired with Saber Light requires 27 packets. Assuming increased packet rate loss of 2%, the probability of losing a packet in the former case is $1 - (1 - rate)^{npacks} = 13\%$, where in the latter it is 42%. In the latter case, at the median, every third session key establishment will experience packet retransmission and hence a delay.

Storage Cost. The Signal handshake protocol assumes the usage of an intermediate server during session key establishment. This allows parties to be offline during the establishment. The server stores long-term keys of each party uploaded during registration, signed pre-keys and pre-key signatures needed to initiate session establishment, as well as data generated during session establishment. Hence, it is important to correctly assess the amount of storage required.

The cost can be split into two parts. One part contains storage of the long-term key lpk and *signed pre-key* pair (ek_T, σ_A). The latter is updated on regular basis, but the server always stores one *signed pre-key*, hence that cost is constant and depends on the number of parties registered. The second part of the cost is data produced during session establishment, namely triple $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ uploaded by the receiver. It is a variable cost, as data can be deleted from the server as soon as the initiator downloads it to finalize the session establishment.

Table 4 shows the split between both costs for a selected number of post-quantum schemes. We can see that to reduce the storage cost, it is beneficial to pair Falcon with SIKE for security levels 1 and 5 and Dilithium with NTRU for security level 3. In some applications, it could be interesting to reduce only the

[21] The results for all 93 instantiations can be found in the repository containing the implementation [57].

Scheme	|pk|	I→S	S→R	R→S	S→I	Total
NIST security level 1						
Falcon512/Saber Light	1569	1362	2931	2162	3731	10186
Falcon512/SIKEp434	1227	1020	2247	1382	2609	7258
Dilithium2/NTRU hps2048509	2011	3119	5130	3818	5829	17896
SPHINCS-SHAKE256-128f-s/Saber Light	704	17760	18464	18560	19264	74048
NIST security level 3						
Dilithium3/NTRU hps2048677	2882	4223	7105	5153	8035	24516
Dilithium3/Saber	2944	4285	7229	5469	8413	25396
Rainbow III/McEliece460896	1406240	524324	1930564	540	1406780	3862208
SPHINCS-SHAKE256-192f-s/Kyber768	1232	36848	38080	37840	39072	151840
NIST security level 5						
Falcon1024/NTRU hps4096821	3023	2560	5583	3790	6813	18746
Falcon1024/Saber Fire	3105	2642	5747	4274	7379	20042
SPHINCS-SHAKE256-256f-s/Saber Fire	1376	51168	52544	52800	54176	210688

Table 3: Data transfer cost in bytes of Figure 4 instantiated with various post-quantum schemes. We use the following abbreviations: I = Initiator, S = Server, R = Responder. Note that:
(a) (S→R) − (I→S) = (S→I) − (R→S) = |pk|.
(b) Total := (I→S) + (S→R) + (R→S) + (S→I).

variable cost. In that case, instantiation can use Rainbow and McEliece pair of algorithms at a higher cost of long-term key storage.

Scheme	Data per user	Data per session
NIST security level 1		
Falcon512/Saber Light	2931	2162
Falcon512/SIKEp434	2247	1382
Dilithium2/NTRU hps2048509	2985	2088
SPHINCS-SHAKE256-128f-s/Saber Light	18464	18560
NIST security level 3		
Dilithium3/NTRU hps2048677	7105	5153
Dilithium3/Saber	7229	5469
Rainbow III/McEliece460896	1930564	540
SPHINCS-SHAKE256-192f-s/Kyber768	38080	37840
NIST security level 5		
Falcon1024/NTRU hps4096821	5583	3790
Falcon1024/Saber Fire	5747	4274
Dilithium5/Kyber1024	10323	7731
SPHINCS-SHAKE256-256f-s/Saber Fire	52544	52800

Table 4: Data storage cost in bytes of Figure 4 instantiated with various post-quantum schemes.

Computational Cost. 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. Our post-quantum Signal handshake performs 9 such operations during a session agreement: the initiator runs a KEM key generation, two KEM decapsulations, one signature generation and one signature

verification, and the responder performs two KEM encapsulations, one signature generation and one signature verification.

It is important that our post-quantum Signal handshake protocol runs efficiently. But as it is an offline protocol, the performance is less critical when compared to online protocols (like for example TLS). The most important difference is that, in the case of Signal handshake, the server does not perform any CPU heavy operations, the session establishment happens less often and parties perform session establishment asynchronously when they are online.

The most performance-critical part of the protocol is the final part of session establishment done by the initiator. At that stage, it may happen that multiple parties request to establish a session with the initiator. In that case, the initiator downloads multiple (and unknown) triples of $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$, which then need to be efficiently processed. Hence, when optimizing for speed the performance of KEM decapsulation and signature verification is most important.

Table 5 contains the number of CPU cycles spent during session establishment for selected post-quantum schemes. The **3-way handshake** column shows the cost of the whole session establishment, the **Finish** column contains the final part of the handshake, performed by the initiator. Presented results exclude the cost of long-term key (lpk) generation.

The best performance comes from instantiations that use Saber KEM and either Falcon or Dilithium signature scheme. We see instantiation with SIKE, SPHINCS, Rainbow or McEliece schemes negatively impacts performance, resulting in orders of magnitude slower execution.

We note that the computational cost is far less absolute as it depends on the concrete implementation of the post-quantum schemes.

Scheme	**3-way handshake**	**Finish**
NIST security level 1		
Falcon512/Saber Light	3596396	638610
Falcon512/SIKEp434	1803371672	810254556
Dilithium2/NTRU hps2048509	6159630	748797
SPHINCS-SHAKE256-128f-s/Saber Light	256821041	11827077
NIST security level 3		
Dilithium3/NTRU hps2048677	11747527	1493656
Dilithium3/Saber	7597588	1476533
Rainbow III/McEliece460896	2798014516	513810164
SPHINCS-SHAKE256-192f-s/Kyber768	710632430	17476256
NIST security level 5		
Falcon1024/NTRU hps4096821	12718741	1328866
Falcon1024/Saber Fire	7611706	1417632
Dilithium5/Kyber1024	10576515	1672158
SPHINCS-SHAKE256-256f-s/Saber Fire	1343959982	18091448

Table 5: Computational cost in CPU cycles of Figure 4 instantiated with various post-quantum schemes. Benchmarking run on the Intel Xeon E3-1220v3 @3.1GhZ with Turbo Boost disabled.

In conclusion. Instantiations of our post-quantum Signal handshake protocol that use Saber as a KEM and either Falcon or Dilithium as signature scheme seem to be the most promising choice for minimizing transfer and storage cost and maximizing performance.

Lastly, our implementation is based on open-source libraries. In total, we created 93 instantiations with different post-quantum schemes. We store the results in the repository containing the implementation [57]. We note that a variety of fine-tuning can be done, leading to different results. For example, one could imagine a scenario crafted for IoT devices, in which devices are pre-configured to communicate only with selected

parties. In such a case, the exchange of long-term keys can be done ahead of time.

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

In this section, we discuss to what extent our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ satisfies deniability and show how to modify the protocol to satisfy a progressively stronger notion of deniability. We first motivate what deniability is and then provide an overview of this section.

Difference Between Deniability of an AKE Protocol and The Signal Handshake. Due to the subtle difference in the model of the standard AKE protocol and the Signal handshake, there is also a subtle difference in what it means to be deniable. In an AKE protocol, roughly, deniability states that the exchanged transcript does not leave any trace of the two parties that supposedly communicated with each other. Namely, both the initiator *and* responder in Figure 2 should be able to deny the fact that they engaged in a key exchange protocol. In contrast, in the Signal handshake, we mainly care about the deniability of the responder (i.e, Bob in Figure 4). This is because the initiator (i.e., Alice in Figure 4) only uploads materials that are independent of the responder. Specifically, an adversary can at most prove to a third-party that the initiator was using the Signal app by showing the pre-key signature of the initiator, and nothing more. Therefore, in the Signal handshake, the main focus is to prevent an adversary from later proving that a certain responder tried to exchange a key with some (possibly malicious) initiator. In summary, the deniability required by an AKE protocol is arguably stronger than what is required by the Signal handshake since it also considers the deniability of the initiator.

That being said, in this section, we mainly focus on the deniability of our AKE protocol rather than the Signal handshake for three reasons. First, we believe our AKE protocol is interesting even outside the context of Signal so it is worth investigating what kind of deniability it offers. Second, it is easy to argue deniability of the Signal handshake once the deniability of the AKE protocol is established since it only consists of ignoring the deniability of the initiator. Finally, if our AKE protocol (or a variant of it) can be shown to be deniable, then when viewed as the Signal handshake, we can further show that the initiator can deny the fact of using the Signal app. Specifically, since the uploaded content of the initiator would also be deniable, it cannot be used as evidence that the initiator was using the Signal app.

Overview of This Section. We first informally show that our AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ already has a very weak form of deniability that may be acceptable in some applications. We then show that we can slightly modify $\Pi_{\mathsf{SC\text{-}AKE}}$ by replacing a standard signature with a *ring signature* to satisfy a stronger notion of deniability. Although this satisfies a much stronger notion of deniability compared to our vanilla $\Pi_{\mathsf{SC\text{-}AKE}}$, it still assumes the parties follow the protocol description (i.e., honest-but-curious). We discuss in Remark 7.11 why this notion of deniability can still be insufficient in practice. Finally, we show how to make the protocol even secure against malicious adversaries that can deviate arbitrarily from the protocol by additionally relying on NIZKs. As it is common with all deniable AKE protocols[22] secure against key-compromise attacks [34, 80, 75], we rely on strong knowledge-type assumptions, including a variant of the *plaintext-awareness* (PA) for the KEM scheme [12, 9, 10].

We note that our protocol is deniable against *quantum* adversaries when they are limited to be honest-but-curious. However, we were not able to formally prove if our protocol satisfies deniability against quantum adversaries taking arbitrary strategies. We believe this is an artifact of the current definition or proof strategy of deniability and leave it as an interesting open problem to formalize and prove quantum deniability of our protocol (see Remark 7.4 for more discussion).

Weak Deniability of $\Pi_{\mathsf{SC\text{-}AKE}}$. Our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ already satisfies a very weak notion of deniability, where the communication transcript does not leave a trace of the *responder* if the two parties honestly execute the AKE protocol. Note that it clearly leaves a trace of the initiator since a signature is included. Concretely, an adversary (e.g., the server) that is passively collecting the communication

[22] We only consider schemes that are proven secure in the (possibly slight variant of the) deniability framework proposed by the seminal work of Di Raimondo et al. [34].

transcript cannot convince a third-party that some responder tried to communicate with some initiator. Informally, this can be checked by observing that the message sent from the responder can be simulated by the adversary on its own. This notion of weak deniability may suffice for some particular settings: only the deniability of the responder is required; the two engaging parties fully trust each other for the correct execution of the protocol; and if they can tolerate the assumption that corruption will not occur. For instance, this includes the Signal handshake setting where the server is trying to provide a proof to a judge that some responder tried to engage in a conversation with some initiator *without* the help of either of the parties. Our protocol will guarantee deniability in such a scenario.

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. In the above example, if the server is colluding with the initiator of the protocol, then they can provide a proof that the responder wanted to start a conversation with the initiator by using knowledge of the session key. This is clear from the fact that in our $\Pi_{\mathsf{SC\text{-}AKE}}$ protocol, the responder generates a signature which nobody else can. Furthermore, in the context of an AKE protocol, it is also desirable for the initiator to be able to deny the fact that it was trying to engage in a key exchange protocol with some responder.

We now discuss how to make our protocol satisfy a stronger notion of deniability where both the initiator and responder can deny even when the communicating peer may be compromised. To this end, we first define deniability for AKE protocols.

7.1 Definition of Deniability and Tool Preparation

We follow a simplified definition of deniability for AKE protocols introduced in the seminal work by Di Raimondo et al. [34]. Discussion on the simplification is provided in Remark 7.3. At a high level, if there exists a simulator $\mathsf{SIM}_{\mathcal{M}}$ that uses only public information that can produce the same view to an adversary $\mathcal{M}$ that engages in a real AKE protocol with honest parties, then the protocol is deniable. The intuition is that if such a $\mathsf{SIM}_{\mathcal{M}}$ exists, then when $\mathcal{M}$ presents a protocol transcript as a "proof" that some party was trying to communicate with it, the party can deny the fact by claiming that the transcript could have been generated by $\mathcal{M}$ running $\mathsf{SIM}_{\mathcal{M}}$ (that only uses public information).

Let Π be an AKE protocol and KeyGen be the key generation algorithm. That is, for any integer $\mu = \mathsf{poly}(\kappa)$ 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 (*possibly maliciously generated*) long-term public key $\mathsf{lpk}_{\mathcal{M}}$ that can be either associated to or not to $\mathcal{M}$'s identity. We then define the *view* of the adversary $\mathcal{M}$ as the entire sets of inputs and outputs 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 $\mathsf{View}_{\mathcal{M}}(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$. Note that if the session key is deniable, then the subsequent communications (that does not use any long-term keys) are deemed deniable as well. In other words, if the establishment of the session key is deniable, then any communications only using that session key between the two parties are deniable as well.

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. Here, unlike the zero-knowledge

simulator for NIZKs, the simulator for an AKE protocol must be executable in the real world [67]. More formally, we have the following.

Definition 7.1 (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*.*

In the above definition, a semi-honest adversary $\mathcal{M}$ is equivalent to an adversary that engages in the protocol honestly but it may try to learn as much as possible from the messages they receive from other parties. Semi-honest adversaries are also termed *passive* since they are only allowed to break security by observing a view of an honest protocol execution.

Remark 7.2 (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 exchanged contents of the sessions authenticated by the session key can also be denied.

Remark 7.3 (Comparison Between Prior Definition). Our definition is weaker than the deniability notion originally proposed by Di Raimondo et al. [34]. 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 $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$. For instance, this allows capturing information that $\mathcal{M}$ may have obtained by eavesdropping on conversations between honest parties (which is not modeled by $\mathsf{View}_{\mathcal{M}}$). Since our goal is to provide a preliminary result 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. [34].[23] We also note that stronger forms of deniability are known and formalized in the universally composable (UC) model [37, 73, 74], however, AKE protocols satisfying such a strong deniability notion are known to achieve weaker security guarantees. For instance, as noted in [74], an AKE protocol cannot be on-line deniable while also being secure against KCI attacks.

Remark 7.4 (Extending to Malicious Quantum Adversaries). We only consider *classical* deniability in this work, where the adversary $\mathcal{M}$ is restricted to be classical. To be precise, although we are able to easily show deniability against semi-honest quantum adversaries, we are not able to do so against 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 the existence of an extractor that can invoke an adversary multiple of times on the *same* randomness. The notion of fixing a randomness is not well-defined in the quantum setting and rewinding an adversary without disturbing the adversary's quantum state is a non-trivial task. We leave it as an interesting problem to formally define a set of tools that allow to show deniability even against malicious quantum adversaries.

Remark 7.5 (A Note on the Deniability Definition of [21]). After the proceedings version of our paper [49] appeared, Brendel et al. [21] introduced a new definition of deniability for AKE protocols. Unlike prior

[23] We observe that although in [34, Definition 2], aux is defined as fixed information that $\mathcal{M}$ cannot adaptively choose, their proof implicitly assumes that aux is sampled adaptively from some distribution dependent on $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$. Such adaptivity of aux is necessary to invoke PA-2 security of the underlying encryption scheme in their security proof. We view enhancing the deniability definition of [34] to capture this adaptivity to be an important future work.

definitions for AKE deniability [34, 37, 80, 73, 74, 75], Brendel et al. considers an indistinguishability-based definition rather than a simulation-based definition. They consider a scenario where all the users honestly generate their keys and the adversary $\mathcal{M}$ is given the secret keys to all of the users. Informally, $\mathcal{M}$ can receive a transcript by querying the challenge oracle on a pair (I, R), representing the Initiator and Receiver. In one mode, $\mathcal{M}$ is given an honest transcript of a real AKE protocol between I and R. In another mode, $\mathcal{M}$ is given a transcript simulated by a simulator SIM who is only given the secret key of R (i.e., the responder) as input. An AKE protocol is then said to be deniable if no efficient adversary $\mathcal{M}$ can tell apart the two modes.

Other than the fact that their definition is not simulation-based, there are three main differences between the definition of Brendel et al. and Definition 7.1: (1) the users are assumed to generate their keys honestly; (2) the adversary $\mathcal{M}$ is assumed to remain passive during the execution of the AKE protocol between I and R; and (3) $\mathcal{M}$ is given all the secret keys of the users. Regardless of $\mathcal{M}$ being malicious or semi-honest in Definition 7.1, the definition of Brendel et al. is stronger regarding (3) since the secret keys of the users are not provided to $\mathcal{M}$ in Definition 7.1. On the other hand, Definition 7.1 is always stronger than Brendel et al. regarding (2) since $\mathcal{M}$ is allowed to deviate from the AKE protocol. Finally, when $\mathcal{M}$ can act maliciously in Definition 7.1, it is stronger than Brendel et al. regarding (1) since $\mathcal{M}$ can inject malicious keys to the system. In general, the two definitions are incomparable.

It is not immediately clear what the real-world impact is of whether or not (1), (2), and (3) are satisfied. In Remark 7.11, we show that if $\mathcal{M}$ can register malicious keys, then it can break deniability, thus showing that deniability under (1) can be insufficient in some practical applications. Put differently, considering only a semi-honest adversary $\mathcal{M}$ in Definition 7.1 or the definition of Brendel et al. may be insufficient. We leave investigation of the impact of (2) and (3) as an interesting future work.

Required Tools. To argue deniability in the following sections we rely on the following tools: ring signature, plaintext-aware (PA-1) secure KEM scheme, and a non-interactive zero-knowledge (NIZK) argument. We use standard notions of ring signatures and NIZK arguments as provided in Sections 2.6 and 2.7. On the other hand, we use a slightly stronger variant of PA-1 secure KEM schemes than those originally defined in [12, 9, 10]. Informally, a KEM scheme is PA-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 associating plaintext K. In our work, we require PA-1 security to hold even when $\mathcal{M}$ is given multiple public keys rather than a single public key [65]. We note that although Di Raimondo et al. [34] considered the standard notion of PA-1 security in their seminal work on deniability of AKE protocols, 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 $\mathcal{M}$ (which is another subtle restriction omitted in the definition used in [34]). The formal definition is provided in Section 2.2.

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

We 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 the initiator no longer signs the first message and a standard signature is replaced by a ring signature. In the context of the Signal handshake, this means the initiator no longer uploads a pre-key signature. We show that this modification provides a secure AKE protocol that has weak forward secrecy as in Definition 3.3. In Remark 7.12, we provide some discussion on what happens if the initiator signs the first message as in $\Pi_{\mathsf{SC\text{-}AKE}}$, while the responder uses a ring signature.

In Section 7.3, we show how to further modify $\Pi_{\mathsf{SC\text{-}DAKE}}$ to a protocol that is deniable even against malicious adversaries by relying on other tools. 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 5, where the gray and dotted-box components are only used to obtain deniability against malicious adversaries.

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

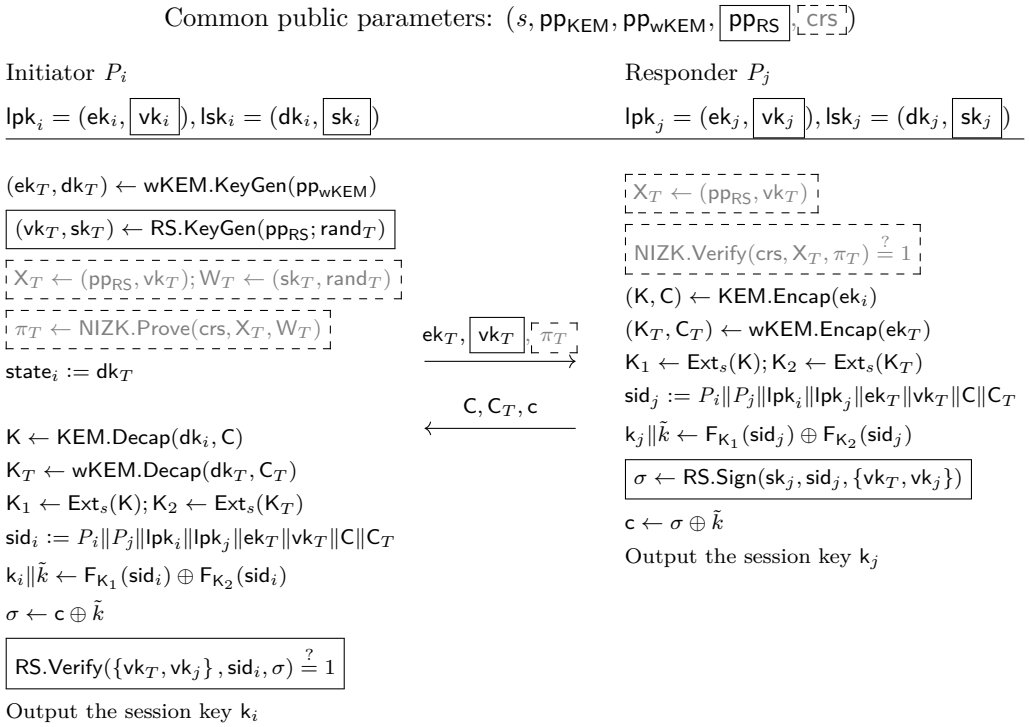


Figure 5: Deniable Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ and $\Pi'_{\mathsf{SC\text{-}DAKE}}$. The initiator no longer signs the first message and the other components that differ from the non-deniable protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ are indicated by a box. The protocol with (resp. without) the gray and dotted-box component satisfies deniability against malicious (resp. semi-honest) adversaries.

- $\Pi_{\mathsf{KEM}} = (\mathsf{KEM.Setup}, \mathsf{KEM.KeyGen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ is a KEM scheme that is IND-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 IND-CPA secure (and not IND-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 $\delta_{\mathsf{wKEM}} = \delta_{\mathsf{KEM}}$, $\nu_{\mathsf{wKEM}} = \nu_{\mathsf{KEM}}$, $\chi_{\mathsf{wKEM}} = \chi_{\mathsf{KEM}}$.
- $\Pi_{\mathsf{RS}} = (\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Verify})$ is a ring signature scheme that is anonymous and unforgeable and assume we have $(1-\delta_{\mathsf{RS}})$-correctness. 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 pp_{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., $\mathsf{state}_i^s := \mathsf{dk}_T$.[24]
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., $\mathsf{state}_j^t = \bot$.
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 $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{reject}, \bot, \bot)$ and stops. Otherwise, P_i sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{accept}, \mathsf{k}_i, \bot)$. 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 B. The main difference from the security proof of $\Pi_{\mathsf{SC\text{-}AKE}}$ is that we have to make sure that removing the initiator's signature only affects forward secrecy, and 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 7.6 (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.*

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

Theorem 7.7 (Security of $\Pi_{\mathsf{SC\text{-}DAKE}}$). *For any* QPT *adversary* $\mathcal{A}$ *that plays the game* $G^{\mathsf{weakFS}}_{\Pi_{\mathsf{SC\text{-}DAKE}}}(\mu,\ell)$ *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* $\mathcal{D}_1,\dots,\mathcal{D}_3$ *breaking the security of PRF* F *such that*

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

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* $\mathcal{B}_1,\dots,\mathcal{B}_4$ *and* $\mathcal{D}_1,\dots,\mathcal{D}_3$ *are about that of* $\mathcal{A}$.

Remark 7.8 (Why the Protocol Does Not Satisfy Full Forward Secrecy). For completeness, we show that $\Pi_{\mathsf{SC\text{-}DAKE}}$ does not satisfy full forward secrecy by constructing an adversary $\mathcal{A}$ that wins the game $G^{\mathsf{FS}}_{\Pi_{\mathsf{SC\text{-}DAKE}}}(\mu,\ell)$ with overwhelming probability. For simplicity, we consider the game with 2 parties P_1 and P_2 that establishes one session per party, i.e., $\mu=2$ and $\ell=1$. $\mathcal{A}$ performs the following attack:

1. $\mathcal{C}$ setups the oracles π^1_1 and π^1_2, and $\mathcal{A}$ obtains the public parameter and the long-term public keys lpk_1 and lpk_2 from the challenger $\mathcal{C}$.
2. $\mathcal{A}$ sends $\mathsf{Send}(2,1,m=\langle\texttt{START}:\texttt{resp},1\rangle)$ to $\mathcal{C}$, which initializes the oracle π^1_2 as the responder and sets its partner to P_1.
3. $\mathcal{A}$ generates the first message $(\mathsf{ek}_T,\mathsf{dk}_T)$ and $(\mathsf{vk}_T,\mathsf{sk}_T)$ according to the protocol description and sends $\mathsf{Send}(2,1,m=(\mathsf{ek}_T,\mathsf{vk}_T))$ to $\mathcal{C}$ and receives $(\mathsf{C},\mathsf{C}_T,\mathsf{c})$ from $\mathcal{C}$. Note that π^1_2 terminates at this point.
4. $\mathcal{A}$ then corrupts P_1 by sending $\mathsf{RevLTK}(1)$ to $\mathcal{C}$ and receives P_1's long-term secret key $\mathsf{lsk}_1=(\mathsf{dk}_1,\mathsf{sk}_1)$.
5. $\mathcal{A}$ decrypts C and C_T using dk_1 and dk_T, respectively. Then it computes the session key k and verifies the signature according to the protocol description.
6. $\mathcal{A}$ sends $\mathsf{Test}(2,1)$ and receive k'. If $\mathsf{k}=\mathsf{k}'$, $\mathcal{A}$ outputs 0 as the guessed bit; otherwise outputs 1.

It is clear that $\mathcal{A}$ perfectly impersonates P_1 acting as an initiator. The session key k computed by $\mathcal{A}$ is the same session key computed by π^1_2, and thus, $\mathcal{A}$ can guess the challenge bit with overwhelming probability. Moreover, the tested oracle π^1_2 is fresh because $\mathcal{A}$ did not send $\mathsf{RevSessKey}$ and $\mathsf{RevState}$ queries, and it corrupted P_1 only after π^1_2 finished the protocol execution. Therefore, $\mathcal{A}$ is a valid adversary against the full forward secrecy game.

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

Theorem 7.9 (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.*[25]

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

[25] Although we only consider a classical adversary $\mathcal{M}$, it can be checked that the exact same proof holds even for a quantum adversary.

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. □

Remark 7.10 (Efficiency of $\Pi_{\mathsf{SC\text{-}DAKE}}$). We evaluate the message size of $\Pi_{\mathsf{SC\text{-}DAKE}}$. The first message of $\Pi_{\mathsf{SC\text{-}DAKE}}$ contains the additional ring signature verification key vk_T. Thus, the size of the first message increases by the amount of vk_T compared to $\Pi_{\mathsf{SC\text{-}AKE}}$.[26] The second message of $\Pi_{\mathsf{SC\text{-}DAKE}}$ contains the ring signature for a ring of two users instead of the standard signature. Examples of post-quantum ring signatures sizes for a ring of two users at the NIST security level 1 are 2.5 KiB (Raptor [61], based on NTRU), 4.4 KiB (DualRing [81], based on M-LWE/SIS), or 3.5 KiB (Calamari [15], based on CSIDH). On the other hand, examples of standard signatures sizes at the same security level are 0.6 KiB (Falcon [72], based on NTRU), 2.3 KiB (Dilithium [62], based on M-LWE/SIS) or 0.26 KiB (CSI-FiSh [16], based on CSIDH). Therefore, when using ring signature schemes [61, 15, 81], it is possible to get the size of the second message to be about 2-3 KiB larger than $\Pi_{\mathsf{SC\text{-}AKE}}$.

Remark 7.11 (Why Deniability Against a Semi-Malicious Adversary May Not Suffice). We provide a concrete attack on $\Pi_{\mathsf{SC\text{-}DAKE}}$ in case the adversary may act maliciously.[27] The scenario is as follows: Alice, the initiator in Figure 5, wants to prove that Bob, the responder in Figure 5, was engaging in a communication with her. In the context of Signal, this means that Alice who uploads her (possibly maliciously generated) key package to the server wants to later prove that Bob was trying to communicate with her.

We consider a specific type of ring signature where for any public parameter $\mathsf{pp}_{\mathsf{RS}}$, the language of all possible verification key vk that can be output by $\mathsf{RS.KeyGen}(\mathsf{pp}_{\mathsf{RS}}, \cdot)$ is an $\mathsf{NP} \cap \mathsf{coNP}$ language. That is, if vk is in the image of $\mathsf{RS.KeyGen}(\mathsf{pp}_{\mathsf{RS}}, \cdot)$, then the randomness used to generate it will be the NP-witness, and if vk is not in the image, we assume there is a coNP-witness to prove it. We further assume the $\mathsf{NP} \cap \mathsf{coNP}$ language can be sampled efficiently along with an accompanying witness. Although it depends on the concrete set of parameters, many lattice-based ring signatures such as [20, 61, 39, 40, 41, 15] where the verification key includes an LWE or NTRU instance could satisfy this property since the (approximated) gap closest vector GapCVP problem lies in $\mathsf{NP} \cap \mathsf{coNP}$ [3].

With such a ring signature, the attack is simple. Alice generates her ring signature verification key vk_A with an accompanying coNP-witness w_{no} to prove that vk_A does not have a corresponding secret key sk_A. This can informally be used as cryptographic evidence that justifies Alice's incapability of signing any message using vk_A. Now, if Bob generates a ring signature $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{sk}_B, \mathsf{sid}_B, \{\mathsf{vk}_A, \mathsf{vk}_B\})$ that includes Alice's verification key, then Alice can later claim to a third-party (e.g., a judge) by presenting $(\mathsf{vk}_A, w_{\mathsf{no}}, \sigma)$ that Bob

[26] To be fair, we compare $\Pi_{\mathsf{SC\text{-}DAKE}}$ with a variant of $\Pi_{\mathsf{SC\text{-}AKE}}$ who not sign the first message. Presented in [49], such variant is as secure as $\Pi_{\mathsf{SC\text{-}DAKE}}$ (modulo the difference between weak and perfect forward secrecy), and the main difference of the two schemes is deniability.

[27] The attack equally works for the subsequent protocol proposed by Brendel et al. [21]. We note that this does not contradict their security proof since the new definition of indistinguishability-based deniability they introduce does not capture malicious adversaries.

engaged in a conversation with her. We note that to make this argument formal in a theoretical sense, Alice would also need to prove that any adversary that can output a valid σ can forge a signature using vk_B. That is, unless Alice knew Bob's verification key, she would not have been able to create σ. As a concrete example, we can consider the Raptor signature scheme [61]. Alice can set her verification key to be a zero-polynomial element. Then, the ring signature produced by Bob reduces to a standard signature created by Bob since the components that depend on Alice's verification key disappears.

Although informal, we believe there are several other simpler ways Alice may convince a real-world third-party her incapability of signing the ring signature provided from Bob. For instance, she may create the verification key using a cryptographically secure hash function, i.e., $\mathsf{vk}_A = \mathsf{H}(\mathsf{rand})$. Even if vk_A is not in $\mathsf{NP} \cap \mathsf{coNP}$, this may already be "good enough" evidence in practice to convince a third-party that Alice could not have signed the message. This is because we can safely assume that computing the secret key from a random vk_A is infeasible.

Considering that the extent of cryptographic evidence that can be considered as real-world judicious evidence is unclear, Bob may want to be able to prove that Alice could have signed the ring signature in a cryptographically sound manner. We provide one possible way on how to achieve this in the next section.

Remark 7.12 (Taking Advantage of the Asymmetry). As we explained at the beginning of this section, there is a subtle difference between the level of deniability we can target for a standard AKE protocol and the Signal handshake. Alice, the initiator in Figure 5, may not need to deny the fact that she was using the Signal app. In this case, Alice may be willing to sign her first message as in our original Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$. Such a signature allows us to prove perfect forward secrecy in Theorem 7.7 rather than weak forward secrecy while still providing deniability for the responder.

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

We provide a Signal-conforming AKE protocol $\Pi'_{\mathsf{SC\text{-}DAKE}}$ that is secure even against malicious adversaries. The construction is provided in Figure 5. To achieve deniability against malicious adversaries, we modify our $\Pi_{\mathsf{SC\text{-}DAKE}}$ 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 [34, 80, 75], this seems unavoidable. On the other hand, there are protocols achieving a strong form of deniability from standard assumptions [37, 73, 74], 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. [34] 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[28] like those provide by Fischlin's NIZK [42]. We leave it to future works to

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

investigate the credibility of the following assumption and those required to prove deniability of the X3DH protocol [75]. We also believe defining a more relaxed notion of deniability that still suffices in practice while also being satisfiable from standard assumptions to be of great importance.

Assumption 7.13 (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 7.1, 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 but the concrete proof required 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 of the adversary and considering PA_μ-1 security against a non-uniform adversary.

Theorem 7.14 (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,*[29] *and the key-awareness assumption in Assumption 7.13 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 7.9. 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 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 $(\mathsf{ek}_T, \mathsf{vk}_T, \mathsf{sk}_T, \pi_T)$, $\mathsf{SIM}_{\mathcal{M},2}$ 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.

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

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 7.6. Moreover, the security of $\Pi'_{\mathsf{SC\text{-}DAKE}}$ follows almost immediately from the proof of Theorem 7.7. The only difference is that in the proof of Lemma B.1 (which is a sub-lemma used to prove Theorem 7.7), 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

[1] LibTomCrypt. https://github.com/libtom/libtomcrypt. 27

[2] Signal protocol: Technical documentation. https://signal.org/docs/. 3, 18

[3] D. Aharonov and O. Regev. Lattice problems in NP cap coNP. In *45th FOCS*, pages 362–371. IEEE Computer Society Press, Oct. 2004. 38

[4] J. Alawatugoda, D. Stebila, and C. Boyd. Modelling after-the-fact leakage for key exchange. In S. Moriai, T. Jaeger, and K. Sakurai, editors, *ASIACCS 14*, pages 207–216. ACM Press, June 2014. 7, 8

[5] J. Alwen, S. Coretti, and Y. Dodis. The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 129–158. Springer, Heidelberg, May 2019. 3, 4

[6] C. Bader, D. Hofheinz, T. Jager, E. Kiltz, and Y. Li. Tightly-secure authenticated key exchange. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 629–658. Springer, Heidelberg, Mar. 2015. 18, 19

[7] M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Heidelberg, Aug. 2006. 26

[8] M. Bellare. New proofs for NMAC and HMAC: Security without collision resistance. *Journal of Cryptology*, 28(4):844–878, Oct. 2015. 26

[9] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer, Heidelberg, Aug. 1998. 9, 31, 34

[10] M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 48–62. Springer, Heidelberg, Dec. 2004. 9, 11, 31, 34

[11] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, Aug. 1994. 6, 18, 19

[12] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995. 9, 11, 31, 34

[13] M. Bellare, A. C. Singh, J. Jaeger, M. Nyayapati, and I. Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 619–650. Springer, Heidelberg, Aug. 2017. 3

[14] D. J. Bernstein. Curve25519: New Diffie-Hellman speed records. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer, Heidelberg, Apr. 2006. 5

[15] W. Beullens, S. Katsumata, and F. Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 464–492. Springer, Heidelberg, Dec. 2020. 7, 38

[16] W. Beullens, T. Kleinjung, and F. Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, Dec. 2019. 38

[17] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer, Heidelberg, Dec. 1997. 3, 18

[18] S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In H. Imai and Y. Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 154–170. Springer, Heidelberg, Mar. 1999. 17

[19] X. Bonnetain and A. Schrottenloher. Quantum security analysis of CSIDH. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 493–522. Springer, Heidelberg, May 2020. 6

[20] Z. Brakerski and Y. T. Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. Cryptology ePrint Archive, Report 2010/086, 2010. https://eprint.iacr.org/2010/086. 38

[21] J. Brendel, R. Fiedler, F. Günther, C. Janson, and D. Stebila. Post-quantum asynchronous deniable key exchange and the signal handshake. Cryptology ePrint Archive, Report 2021/769, 2021. 8, 9, 33, 38, 60, 61, 62

[22] J. Brendel, M. Fischlin, F. Günther, C. Janson, and D. Stebila. Towards post-quantum security for signal's X3DH handshake. In O. Dunkelman, M. J. Jacobson, Jr., and C. O'Flynn, editors, *Selected Areas in Cryptography*, pages 404–430, Cham, 2020. Springer International Publishing. 3, 4, 6

[23] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, Heidelberg, May 2001. 4, 6, 15, 17, 18, 19

[24] R. Canetti and H. Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 143–161. Springer, Heidelberg, Aug. 2002. https://eprint.iacr.org/2002/120/. 6, 18

[25] D. Cash, E. Kiltz, and V. Shoup. The twin Diffie-Hellman problem and applications. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 127–145. Springer, Heidelberg, Apr. 2008. 5

[26] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the signal messaging protocol. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 451–466, 2017. 3, 4

[27] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the signal messaging protocol. *Journal of Cryptology*, 33(4):1914–1983, 2020. 3, 4

[28] K. Cohn-Gordon, C. Cremers, K. Gjøsteen, H. Jacobsen, and T. Jager. Highly efficient key exchange protocols with optimal tightness. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 767–797. Springer, Heidelberg, Aug. 2019. 6, 15, 18, 19, 20

[29] C. Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In B. S. N. Cheung, L. C. K. Hui, R. S. Sandhu, and D. S. Wong, editors, *ASIACCS 11*, pages 80–91. ACM Press, Mar. 2011. 19

[30] C. J. F. Cremers. Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS authenticated key exchange protocol. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 20–33. Springer, Heidelberg, June 2009. 19

[31] C. J. F. Cremers and M. Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In S. Foresti, M. Yung, and F. Martinelli, editors, *ESORICS 2012*, volume 7459 of *LNCS*, pages 734–751. Springer, Heidelberg, Sept. 2012. 18, 19

[32] B. de Kock, K. Gjøsteen, and M. Veroni. Practical isogeny-based key-exchange with optimal tightness. In O. Dunkelman, M. J. Jacobson, Jr., and C. O'Flynn, editors, *Selected Areas in Cryptography*, pages 451–479, Cham, 2020. Springer International Publishing. 6, 18

[33] C. de Saint Guilhem, M. Fischlin, and B. Warinschi. Authentication in key-exchange: Definitions, relations and composition. In L. Jia and R. Küsters, editors, *CSF 2020 Computer Security Foundations Symposium*, pages 288–303. IEEE Computer Society Press, 2020. 15, 19, 20

[34] M. Di Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 400–409. ACM Press, Oct. / Nov. 2006. 5, 7, 9, 11, 31, 32, 33, 34, 39

[35] W. Diffie, P. C. Van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and cryptography*, 2(2):107–125, 1992. 17

[36] S. Dobson and S. D. Galbraith. Post-quantum signal key agreement with SIDH. Cryptology ePrint Archive, Report 2021/1187, 2021. https://ia.cr/2021/1187. 8, 9

[37] Y. Dodis, J. Katz, A. Smith, and S. Walfish. Composability and on-line deniability of authentication. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 146–162. Springer, Heidelberg, Mar. 2009. 9, 33, 34, 39

[38] F. B. Durak and S. Vaudenay. Bidirectional asynchronous ratcheted key agreement with linear complexity. In N. Attrapadung and T. Yagi, editors, *IWSEC 19*, volume 11689 of *LNCS*, pages 343–362. Springer, Heidelberg, Aug. 2019. 3

[39] M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 115–146. Springer, Heidelberg, Aug. 2019. 38

[40] M. F. Esgin, R. Steinfeld, A. Sakzad, J. K. Liu, and D. Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 67–88. Springer, Heidelberg, June 2019. 38

[41] M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 567–584. ACM Press, Nov. 2019. 38

[42] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, Aug. 2005. 39

[43] P.-A. Fouque, D. Pointcheval, and S. Zimmer. HMAC is a randomness extractor and applications to TLS. In M. Abe and V. Gligor, editors, *ASIACCS 08*, pages 21–32. ACM Press, Mar. 2008. 26

[44] E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Heidelberg, Feb. / Mar. 2013. 5

[45] A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 467–484. Springer, Heidelberg, May 2012. 4, 6, 7, 18, 19

[46] A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama. Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In K. Chen, Q. Xie, W. Qiu, N. Li, and W.-G. Tzeng, editors, *ASIACCS 13*, pages 83–94. ACM Press, May 2013. 6

[47] K. Gjøsteen and T. Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 95–125. Springer, Heidelberg, Aug. 2018. 6, 15, 18, 19

[48] S. Guo, P. Kamath, A. Rosen, and K. Sotiraki. Limits on the efficiency of (ring) LWE based non-interactive key exchange. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 374–395. Springer, Heidelberg, May 2020. 3

[49] K. Hashimoto, S. Katsumata, K. Kwiatkowski, and T. Prest. An efficient and generic construction for signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. In J. Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 410–440. Springer, Heidelberg, May 2021. 1, 8, 9, 33, 38

[50] K. Hövelmanns, E. Kiltz, S. Schäge, and D. Unruh. Generic authenticated key exchange in the quantum random oracle model. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 389–422. Springer, Heidelberg, May 2020. 6, 18, 19

[51] T. Jager, E. Kiltz, D. Riepel, and S. Schäge. Tightly-secure authenticated key exchange, revisited. In A. Canteaut and F.-X. Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 117–146. Springer, Heidelberg, Oct. 2021. 18, 19

[52] D. Jost, U. Maurer, and M. Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 159–188. Springer, Heidelberg, May 2019. 3

[53] D. Jost, U. Maurer, and M. Mularczyk. A unified and composable take on ratcheting. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 180–210. Springer, Heidelberg, Dec. 2019. 3

[54] T. Kawashima, K. Takashima, Y. Aikawa, and T. Takagi. An efficient authenticated key exchange from random self-reducibility on CSIDH. In D. Hong, editor, *ICISC 20*, volume 12593 of *LNCS*, pages 58–84. Springer, Heidelberg, Dec. 2020. 6, 18

[55] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, Heidelberg, Aug. 2005. 4, 6, 17, 18, 19

[56] K. Kurosawa and J. Furukawa. 2-pass key exchange protocols from CPA-secure KEM. In J. Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 385–401. Springer, Heidelberg, Feb. 2014. 4, 6, 8

[57] K. Kwiatkowski. An efficient and generic construction for signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. proof of concept implementation, December 2020. https://github.com/post-quantum-cryptography/post-quantum-state-leakage-secure-ake. 4, 26, 28, 30

[58] K. Kwiatkowski. PQ Crypto Catalog, December 2020. https://github.com/kriskwiatkowski/pqc. 27

[59] B. A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In W. Susilo, J. K. Liu, and Y. Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 1–16. Springer, Heidelberg, Nov. 2007. 4, 6, 7, 18, 19

[60] Y. Li and S. Schäge. No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 1343–1360. ACM Press, Oct. / Nov. 2017. 17, 19

[61] X. Lu, M. H. Au, and Z. Zhang. Raptor: A practical lattice-based (linkable) ring signature. In R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 110–130. Springer, Heidelberg, June 2019. 7, 38, 39

[62] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehlé, and S. Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions. 38

[63] M. Marlinspike and T. Perrin. The double ratchet algorithm, November 2016. https://signal.org/docs/specifications/doubleratchet/. 3

[64] M. Marlinspike and T. Perrin. The X3DH key agreement protocol, November 2016. https://signal.org/docs/specifications/x3dh/. 3, 4, 5, 7, 18, 24, 27

[65] S. Myers, M. Sergi, and a. shelat. Blackbox construction of a more than non-malleable CCA1 encryption scheme from plaintext awareness. In I. Visconti and R. D. Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 149–165. Springer, Heidelberg, Sept. 2012. 11, 34

[66] C. Paquin, D. Stebila, and G. Tamvada. Benchmarking post-quantum cryptography in TLS. In J. Ding and J.-P. Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 72–91. Springer, Heidelberg, 2020. 28

[67] R. Pass. On deniability in the common reference string and random oracle model. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, Aug. 2003. 33

[68] C. Peikert. He gives C-sieves on the CSIDH. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 463–492. Springer, Heidelberg, May 2020. 6

[69] T. Perrin. The XEdDSA and VXEdDSA signature schemes, October 2016. https://signal.org/docs/specifications/xeddsa/. 5

[70] B. Poettering and P. Rösler. Towards bidirectional ratcheted key exchange. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 3–32. Springer, Heidelberg, Aug. 2018. 3

[71] D. Pointcheval and O. Sanders. Forward secure non-interactive key exchange. In M. Abdalla and R. D. Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 21–39. Springer, Heidelberg, Sept. 2014. 5

[72] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions. 38

[73] N. Unger and I. Goldberg. Deniable key exchanges for secure messaging. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 1211–1223. ACM Press, Oct. 2015. 3, 9, 33, 34, 39

[74] N. Unger and I. Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *PoPETs*, 2018(1):21–66, Jan. 2018. 3, 9, 33, 34, 39

[75] N. Vatandas, R. Gennaro, B. Ithurburn, and H. Krawczyk. On the cryptographic deniability of the Signal protocol. In M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, editors, *ACNS 20, Part II*, volume 12147 of *LNCS*, pages 188–209. Springer, Heidelberg, Oct. 2020. 3, 5, 7, 9, 31, 34, 39, 40

[76] H. Xue, M. H. Au, R. Yang, B. Liang, and H. Jiang. Compact authenticated key exchange in the quantum random oracle model. Cryptology ePrint Archive, Report 2020/1282, 2020. https://eprint.iacr.org/2020/1282. 6

[77] H. Xue, X. Lu, B. Li, B. Liang, and J. He. Understanding and constructing AKE via double-key key encapsulation mechanism. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 158–189. Springer, Heidelberg, Dec. 2018. 6

[78] Z. Yang. Modelling simultaneous mutual authentication for authenticated key exchange. In J. L. Danger, M. Debbabi, J.-Y. Marion, J. Garcia-Alfaro, and N. Zincir Heywood, editors, *Foundations and Practice of Security*, pages 46–62, Cham, 2014. Springer International Publishing. 20

[79] Z. Yang, Y. Chen, and S. Luo. Two-message key exchange with strong security from ideal lattices. In N. P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 98–115. Springer, Heidelberg, Apr. 2018. 4, 6, 8

[80] A. C.-C. Yao and Y. Zhao. Deniable internet key exchange. In J. Zhou and M. Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 329–348. Springer, Heidelberg, June 2010. 5, 7, 9, 31, 34, 39

[81] T. H. Yuen, M. F. Esgin, J. K. Liu, M. H. Au, and Z. Ding. DualRing: Generic construction of ring signatures with efficient instantiations. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 251–281. Springer, Heidelberg, Aug. 2021. 7, 38

A Full 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 4.3. Let $\mathcal{A}$ be an adversary that plays the security game $G^{\mathsf{FS}}_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mu,\ell)$ with the challenger $\mathcal{C}$ with advantage $\mathsf{Adv}^{\mathsf{AKE\text{-}FS}}_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mathcal{A}) = \epsilon$. In order to prove Theorem 4.3, 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.[30] 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. 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 six algorithms $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_{3,0}$, $\mathcal{B}_{3,1}$, $\mathcal{D}_1$ and $\mathcal{D}_2$ 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 $\mathcal{B}_{3,0}$ 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 $\mathcal{B}_{3,1}$ succeeds in breaking the $\mathsf{EUF\text{-}CMA}$ security of Π_{SIG} with advantage $\approx \frac{1}{\mu}\epsilon$.

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 4.2, 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 A.1 to A.4 provided in their respective subsections below. We first complete the proof of the theorem. Specifically, by combining all the lemmata together and folding adversaries $\mathcal{B}_{3,0}$ and $\mathcal{B}_{3,1}$ into one adversary $\mathcal{B}_3$, we obtain the following desired bound:

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

[30] 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.

Here, the running time of the algorithms $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, $\mathcal{D}_1$ and $\mathcal{D}_2$ 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 A.1 to A.4.

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

Lemma A.1. *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*

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

Proof of Lemma A.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 $\pi_i^{\hat{s}}$ and a responder oracle $\pi_j^{\hat{t}}$ uniformly at random from the $\mu\ell$ oracles. Let $\mathsf{E}_{\mathsf{testO}}$ be the event that the tested oracle is neither $\pi_i^{\hat{s}}$ nor $\pi_j^{\hat{t}}$, or $\pi_i^{\hat{s}}$ and $\pi_j^{\hat{t}}$ 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.[31] $\mathcal{C}$ 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 $\pi_i^{\hat{s}}$ responds on its second invocation. In particular, when $\pi_i^{\hat{s}}$ 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 $\pi_j^{\hat{t}}$ 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 $\pi_j^{\hat{t}}$ 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 $\pi_i^{\hat{s}}$ outputs on the first invocation. This is because otherwise, the oracles $\pi_i^{\hat{s}}$ and $\pi_j^{\hat{t}}$ 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 $\pi_j^{\hat{t}}$ responds. When the responder oracle $\pi_j^{\hat{t}}$ 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 $\pi_i^{\hat{s}}$ 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 IND-CPA security of Π_{wKEM}. To prove this, we construct an algorithm $\mathcal{B}_1$ breaking the IND-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.
- $\mathsf{Send}(j, t, m = (\mathsf{ek}_T, \sigma_i))$: 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 $\pi_i^{\hat{s}}$ and $\pi_j^{\hat{t}}$ cannot be partner oracles. Therefore, since event $\mathsf{E}_{\mathsf{testO}}$ is triggered $\mathcal{B}_1$ aborts.

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

– 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 $\pi_i^{\hat{s}}$ 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., $\mathsf{state}_i^{\hat{s}} := \mathsf{dk}^*$) conditioning on $\mathsf{E}_{\mathsf{testO}}$ not occurring, which is the only query that $\mathcal{B}_1$ cannot answer.

- $\mathsf{Test}(i,s)$: $\mathcal{B}_1$ 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 $\pi_i^{\hat{s}}$ 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.[32] 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 A.2: Against Type-3 or Type-4 Adversary.

[32] We note that for Lemma A.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.

Lemma A.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 A.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

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

We claim

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

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

Case 1. We first consider the case π^s_i is an initiator and π^t_j and $\pi^{t'}_j$ are responders. Let ek_T be the ephemeral encapsulation key generated by π^s_i. In this case, $\mathsf{E}_{\mathsf{uniq}}$ occurs if the responder oracles π^t_j and $\pi^{t'}_j$ 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^s_i is a responder and π^t_j and $\pi^{t'}_j$ are initiators. In this case, $\mathsf{E}_{\mathsf{uniq}}$ occurs if the initiator oracles π^t_j and $\pi^{t'}_j$ 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 $P_{\hat{i}}$ from the μ parties and a random responder oracle $\pi^{\hat{t}}_{\hat{j}}$ 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 $\pi^{\hat{s}}_{\hat{i}}$ for some $s \in [\ell]$ and its partner oracle is $\pi^{\hat{t}}_{\hat{j}}$, or the tested oracle is $\pi^{\hat{t}}_{\hat{j}}$ and its peer is $P_{\hat{i}}$. 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 $\pi^s_{\hat{i}}$ for any $s \in [\ell]$ responds on its second invocation. Let (K, C) be the Π_{KEM} key-ciphertext pair generated by oracle $\pi^{\hat{t}}_{\hat{j}}$. Then, when $\pi^s_{\hat{i}}$ 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 $\pi^{\hat{t}}_{\hat{j}}$ 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 $\pi_{\hat{j}}^{\hat{t}}$ responds. When the responder oracle $\pi_{\hat{j}}^{\hat{t}}$ is invoked on input ek_T, it samples a random key $\mathsf{K} \leftarrow\!\!\$\, \mathcal{KS}_{\mathsf{KEM}}$ instead of computing $(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_{\hat{i}})$. Note that due to the modification we made in the previous game, when the initiator oracle $\pi_{\hat{i}}^{s}$ 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 $\pi_{\hat{j}}^{\hat{t}}$. We claim G_4 and G_5 are indistinguishable assuming the IND-CCA security of Π_{KEM}. To prove this, we construct an algorithm $\mathcal{B}_2$ breaking the IND-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.
- $\mathsf{Send}(j, t, m = (\mathsf{ek}_T, \sigma_i))$: 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

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

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 $\pi_{\hat{j}}^{\hat{t}}$. 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

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

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 $\pi_{\hat{j}}^{\hat{t}}$ 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 A.3: Against Type-5 or Type-6 Adversary.

Lemma A.3. *For any* QPT *adversary* $\mathcal{A}$ *using the Type-5 or Type-6 strategy, there exists a* QPT *algorithm* $\mathcal{B}_{3,0}$ *breaking the* EUF-CMA *of* Π_{SIG} *such that*

$$\epsilon_1 \leq \mu \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B}_{3,0}).$$

Proof of Lemma A.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 $\pi_{\hat{j}}^t$ for some $t \in [\ell]$ is invoked as a responder, it updates the list S by appending the message-signature pair $(\mathsf{sid}_{\hat{j}}^t, \sigma_{\hat{j}}^t)$ 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 $P_{\hat{\jmath}}$ (i.e., $\mathsf{Pid}_i^s = \hat{\jmath}$), it first computes sid_i^s and σ as in the previous game, and it checks if $\mathsf{SIG.Verify}(\mathsf{vk}_{\hat{\jmath}}, \mathsf{sid}_i^s, \sigma) = 1$ and $P_{\hat{\jmath}}$ is not corrupted, then $(\mathsf{sid}_i^s, \sigma) \in S$. 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 $\pi_{\hat{j}}^t$ for any $t \in [\ell]$ and the peer $P_{\hat{\jmath}}$ was not corrupted before $\pi_{i^*}^{s^*}$ completes the protocol execution 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 $\pi_{\hat{j}}^t$ 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 $\pi_{\hat{j}}^t$ 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 $\Pr[\mathsf{S}_3] = 1/2$.

It remains to bound $\Pr[\mathsf{E}_{\mathsf{sig}}]$. We do this by constructing an algorithm $\mathcal{B}_{3,0}$ against the EUF-CMA security of Π_{SIG}. The description of $\mathcal{B}_{3,0}$ follows: $\mathcal{B}_{3,0}$ receives the public parameter $\mathsf{pp}_{\mathsf{SIG}}$ and the challenge verification

key vk^*. $\mathcal{B}_{3,0}$ sets up the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ as in G_2 using $\mathsf{pp}_{\mathsf{SIG}}$. $\mathcal{B}_{3,0}$ 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 $\mathcal{B}_{3,0}$. For the rest of the parties P_i for $i \in [\mu \backslash \hat{\jmath}]$, $\mathcal{B}_{3,0}$ generates $(\mathsf{lpk}_i, \mathsf{lsk}_i)$ as in G_2. Finally, $\mathcal{B}_{3,0}$ 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:

- $\mathsf{Send}(i, s, \langle \mathtt{START} : \mathsf{role}, j \rangle)$: $\mathcal{B}_{3,0}$ responds as in G_2.
- $\mathsf{Send}(j, t, m = (\mathsf{ek}_T, \sigma_i))$: Depending on the value of j, it performs the following:
  - If $j = \hat{\jmath}$, then $\mathcal{B}_{3,0}$ 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^*. $\mathcal{B}_{3,0}$ then responds as in G_2 except that it sets $\sigma := \sigma'$.
  - If $j \neq \hat{\jmath}$, then $\mathcal{B}_{3,0}$ responds as in G_2.
- $\mathsf{Send}(i, s, m = (\mathsf{C}, \mathsf{C}_T, \mathsf{c}))$: $\mathcal{B}_{3,0}$ responds as in G_2.
- $\mathsf{RevLTK}(i)$, $\mathsf{RegisterLTK}(i)$, $\mathsf{RevState}(i, s)$, $\mathsf{RevSessKey}(i, s)$: $\mathcal{B}_{3,0}$ responds as in G_2. Here, note that since $\mathcal{A}$ follows the Type-5 or Type-6 strategy, $\mathcal{B}_{3,0}$ 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 $\mathcal{B}_{3,0}$ cannot answer.
- $\mathsf{Test}(i, s)$: $\mathcal{B}_{3,0}$ 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 $\mathcal{B}_{3,0}$ perfectly simulates the view of game G_2 to $\mathcal{A}$. Below, we analyze the probability that $\mathcal{B}_{3,0}$ 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 $(\mathsf{ek}_T^*, \sigma_{i^*})$ 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,
- $P_{\hat{\jmath}}$ was not corrupted before $\pi_{i^*}^{s^*}$ completes the protocol execution,
- $\pi_{i^*}^{s^*}$ 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^*} and output $(\mathsf{C}^*, \mathsf{C}_T^*)$. 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 $t \in [\ell]$, and $P_{\hat{\jmath}}$ was not corrupted when $\pi_{i^*}^{s^*}$ receives the signature. Therefore, $\mathcal{B}_{3,0}$ obtains a valid forgery $(P_{i^*} \| P_{\hat{\jmath}} \| \mathsf{lpk}_{i^*} \| \mathsf{lpk}_{\hat{\jmath}} \| \mathsf{ek}_T^* \| \mathsf{C}^* \| \mathsf{C}_T^*, \sigma^*)$, and we have $\Pr[\mathsf{E}_{\mathsf{sig}}] = \mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{B}_{3,0})$. Combining everything together, we conclude

$$\epsilon_1 \leq \mu \cdot \mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{B}_{3,0}).$$

□

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

Lemma A.4. *For any* QPT *adversary* $\mathcal{A}$ *using the Type-7 or Type-8 strategy, there exists a* QPT *algorithm* $\mathcal{B}_{3,1}$ *breaking the* $\mathsf{EUF\text{-}CMA}$ *of* Π_{SIG} *such that*

$$\epsilon_1 \leq \mu \cdot \mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{B}_{3,1}).$$

Proof of Lemma A.4. 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 $P_{\hat{\imath}}$ uniformly at random from the μ parties. Let $\mathsf{E}_{\mathsf{testO}}$ be the event that the peer of the tested oracle is not $P_{\hat{\imath}}$. 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 $P_{\hat{\imath}}$ generated as being an initiator oracle. That is, every time π_i^s for some $s \in [\ell]$ is invoked as an initiator, it updates the list S by appending the message-signature pair $(\mathsf{ek}_i^s, \sigma_i^s)$ that it generates. Then, when a responder oracle π_j^t for any $(j,t) \in [\mu] \times [\ell]$ is invoked on input (ek_T, σ) from party $P_{\hat{\imath}}$ (i.e., $\mathsf{Pid}_j^t = \hat{\imath}$), it checks if $\mathsf{SIG.Verify}(\mathsf{vk}_{\hat{\imath}}, \mathsf{ek}_T, \sigma) = 1$ and $P_{\hat{\imath}}$ is not corrupted, then $(\mathsf{ek}_T, \sigma) \in S$. 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-7 or Type-8 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 $\mathsf{Test}(j^*, t^*)$ and received a key that is not a $\perp$. That is $\pi_{j^*}^{t^*}$ is in the `accept` state. Due to the modification we made in game G_2 and by the definition of the Type-7 or Type-8 strategy, $\pi_{j^*}^{t^*}$ has no partner oracle π_i^s for any $s \in [\ell]$ and the peer $P_{\hat{\imath}}$ was not corrupted before $\pi_{j^*}^{t^*}$ completes the protocol execution conditioning on $\mathsf{E}_{\mathsf{testO}}$ not occurring. On the other hand, if $\pi_{j^*}^{t^*}$ is in the `accept` state, then event $\mathsf{E}_{\mathsf{sig}}$ must have not been triggered. Consequently, there exists some oracle π_i^s that output $(\mathsf{ek}_i^s, \sigma_i^s)$ and $\pi_{j^*}^{t^*}$ receives it. This implies that π_i^s and $\pi_{j^*}^{t^*}$ are partner oracles. Since this forms a contradiction, $\mathcal{A}$ can only receive $\perp$ when it issues $\mathsf{Test}(j^*, t^*)$. 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 $\mathcal{B}_{3,1}$ against the $\mathsf{EUF\text{-}CMA}$ security of Π_{SIG}. The description of $\mathcal{B}_{3,1}$ follows: $\mathcal{B}_{3,1}$ receives the public parameter $\mathsf{pp}_{\mathsf{SIG}}$ and the challenge verification key vk^*. $\mathcal{B}_{3,1}$ sets up the public parameter of $\Pi_{\mathsf{SC\text{-}AKE}}$ as in G_2 using $\mathsf{pp}_{\mathsf{SIG}}$. $\mathcal{B}_{3,1}$ then samples $\hat{\imath}$ randomly from $[\mu]$, runs $(\mathsf{dk}_{\hat{\imath}}, \mathsf{ek}_{\hat{\imath}}) \leftarrow \mathsf{KEM.KeyGen}(\mathsf{pp}_{\mathsf{KEM}})$, and sets the long-term public key of party $P_{\hat{\imath}}$ as $\mathsf{lpk}_{\hat{\imath}} := (\mathsf{ek}_{\hat{\imath}}, \mathsf{vk}^*)$. The long-term secret key is implicitly set as $\mathsf{lsk}_{\hat{\imath}} := (\mathsf{dk}_{\hat{\imath}}, \mathsf{sk}^*)$, where sk^* is unknown to $\mathcal{B}_{3,1}$. For the rest of the parties P_i for $i \in [\mu \backslash \hat{\imath}]$, $\mathcal{B}_{3,1}$ generates $(\mathsf{lpk}_i, \mathsf{lsk}_i)$ as in G_2. Finally, $\mathcal{B}_{3,1}$ 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:

- $\mathsf{Send}(i, s, \langle \texttt{START} : \mathsf{role}, j \rangle)$: Depending on the value of i, it performs the following:
  - If $i = \hat{\imath}$, then $\mathcal{B}_{3,1}$ prepares ek_T as in G_2, and then sends ek_T to its signing oracle and receives back a signature σ' for message ek_T under sk^*. $\mathcal{B}_{3,1}$ then responds as in G_2 except that it sets $\sigma_i := \sigma'$.
  - If $i \neq \hat{\imath}$, then $\mathcal{B}_{3,1}$ responds as in G_2.
- $\mathsf{Send}(j, t, m = (\mathsf{ek}_T, \sigma_i))$: $\mathcal{B}_{3,1}$ responds as in G_2.
- $\mathsf{Send}(i, s, m = (\mathsf{C}, \mathsf{C}_T, \mathsf{c}))$: $\mathcal{B}_{3,1}$ responds as in G_2.
- $\mathsf{RevLTK}(i)$, $\mathsf{RegisterLTK}(i)$, $\mathsf{RevState}(i, s)$, $\mathsf{RevSessKey}(i, s)$: $\mathcal{B}_{3,1}$ responds as in G_2. Here, note that since $\mathcal{A}$ follows the Type-7 or Type-8 strategy, $\mathcal{B}_{3,1}$ can answer all the RevLTK-query. Namely, $\mathcal{A}$ never queries $\mathsf{RevLTK}(\hat{\imath})$ (i.e., $\mathsf{lsk}_{\hat{\imath}} := (\mathsf{dk}_{\hat{\imath}}, \mathsf{sk}^*)$) conditioning on $\mathsf{E}_{\mathsf{testO}}$ not occurring, which is the only query that $\mathcal{B}_{3,1}$ cannot answer.
- $\mathsf{Test}(i, s)$: $\mathcal{B}_{3,1}$ responds as in G_2. Here, in case $\mathsf{Pid}_i^s \neq \hat{\imath}$, then event $\mathsf{E}_{\mathsf{testO}}$ is triggered, so it aborts.

It is clear that $\mathcal{B}_{3,1}$ perfectly simulates the view of game G_2 to $\mathcal{A}$. Below, we analyze the probability that $\mathcal{B}_{3,1}$ breaks the $\mathsf{EUF\text{-}CMA}$ security of Π_{SIG} and relate it to $\Pr[\mathsf{E}_{\mathsf{sig}}]$.

We assume $\mathcal{A}$ issues $\mathsf{Test}(j^*, t^*)$. Let the message received by the responder oracle $\pi_{j^*}^{t^*}$ be $(\mathsf{ek}_T^*, \sigma^*)$. Then, by the definition of the Type-7 or Type-8 strategy and conditioned on $\mathsf{E}_{\mathsf{testO}}$ not occurring, the oracle $\pi_{j^*}^{t^*}$ satisfies the following conditions:

- $\mathsf{role}_{j^*}^{t^*} = \mathtt{resp}$ and $\mathsf{Pid}_{j^*}^{t^*} = \hat{\imath}$,
- $\pi_{j^*}^{t^*}$ is in the $\mathtt{accept}$ state. This implies $\mathsf{SIG.Verify}(\mathsf{vk}^*, \mathsf{ek}_T^*, \sigma^*) = 1$ holds,
- $P_{\hat{\imath}}$ was not corrupted before $\pi_{j^*}^{t^*}$ completes the protocol execution,
- $\pi_{i^*}^{s^*}$ has no partner oracles.

Since $\pi_{j^*}^{t^*}$ has no partner oracles, there exists no initiator oracle $\pi_{\hat{\imath}}^{s}$ that has output $(\mathsf{ek}_T^*, \sigma^*)$. In other words, there is no oracle $\pi_{\hat{\imath}}^{s}$ that has signed the message ek_T^*. Notice that this is exactly the event $\mathsf{E}_{\mathsf{sig}}$; a responder oracle $\pi_{j^*}^{t^*}$ receives a signature that was not signed by an oracle $\pi_{\hat{\imath}}^{s}$ for any $s \in [\ell]$, and $P_{\hat{\imath}}$ was not corrupted when $\pi_{j^*}^{t^*}$ receives the signature. Therefore, $\mathcal{B}_{3,1}$ obtains a valid forgery $(\mathsf{ek}_T^*, \sigma^*)$, and we have $\Pr[\mathsf{E}_{\mathsf{sig}}] = \mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{B}_{3,1})$

Combining everything together, we conclude

$$\epsilon_1 \leq \mu \cdot \mathsf{Adv}_{\mathsf{SIG}}^{\mathsf{EUF\text{-}CMA}}(\mathcal{B}_{3,1}).$$

□

B Full 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 $\Pi_{\mathsf{SC\text{-}DAKE}}$.

B.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 7.6. This proof is similar to the proof of Theorem 4.2. 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 $\mu\ell \cdot (\delta_{\mathsf{RS}} + 2\delta_{\mathsf{KEM}})/2$. □

B.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 7.7. Let $\mathcal{A}$ be an adversary that plays the security game $G_{\Pi_{\mathsf{SC\text{-}DAKE}}}^{\mathsf{weakFS}}(\mu, \ell)$ with the challenger $\mathcal{C}$ with advantage $\mathsf{Adv}_{\Pi_{\mathsf{SC\text{-}DAKE}}}^{\mathsf{AKE\text{-}weakFS}}(\mathcal{A}) = \epsilon$. The bulk of the proof is identical to the proof of Theorem 4.3 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 $\mathcal{B}_1, \ldots, \mathcal{B}_4$ 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 $\mathsf{IND\text{-}CPA}$ 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 4.3, 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 7.6, 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 B.1 to B.4 provided below. We first complete the proof of the theorem. Specifically, by combining all the lemmata together, we obtain the following desired bound:

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

Here, the running time of the algorithms $\mathcal{B}_1, \ldots, \mathcal{B}_4$ 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. Since the proof of Lemmata B.3 and B.4 is a direct consequence of the proof of the corresponding Lemmata A.1 and A.2 of Theorem 4.3,[33] we focus on proving Lemma B.1 and Lemma B.2 below.

Lemma B.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*

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

[33] Note that Lemma B.3 (resp. Lemma B.4) corresponds to Lemma A.1 (resp. Lemma A.2).

Proof of Lemma B.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 $\mathsf{E}_{\mathsf{sig}}$. 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_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 `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 `accept` state, then event $\mathsf{E}_{\mathsf{sig}}$ 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 $\Pr[\mathsf{E}_{\mathsf{sig}}]$. 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:

- $\mathsf{Send}(i, s, \langle \texttt{START} : \mathsf{role}, j \rangle)$: $\mathcal{B}_3$ 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 $\Pr[\mathsf{E}_{\mathsf{sig}}]$.

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:

- $\mathsf{role}_{i^*}^{s^*} = \texttt{init}$,
- P_j is not corrupted where $\mathsf{Pid}_{i^*}^{s^*} = j$ and $j \in [\mu]$,
- $\pi_{i^*}^{s^*}$ is in the `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

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

□

Lemma B.2. *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.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 $\mathsf{E}_{\mathsf{coll}}$ 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

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

We claim

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

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 $\pi_{\hat{j}}^{\hat{t}}$ 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 $\pi_{\hat{j}}^{\hat{t}}$ 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

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

Game G_4. In this game, we modify the way the initiator oracle π_i^s for any $s \in [\ell]$ responds on its second invocation. Let (K, C) be the Π_{KEM} key-ciphertext pair generated by oracle $\pi_{\hat{j}}^{\hat{t}}$. Then, when π_i^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 $\pi_{\hat{j}}^{\hat{t}}$ 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 $\pi_{\hat{j}}^{\hat{t}}$ responds. When the responder oracle $\pi_{\hat{j}}^{\hat{t}}$ is invoked on input ek_T, it samples a random key $\mathsf{K} \leftarrow\!\!\$\, \mathcal{KS}_{\mathsf{KEM}}$ instead of computing $(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_{\hat{i}})$. Note that due to the modification we made in the previous game, when the initiator oracle $\pi_{\hat{i}}^{s}$ 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 $\pi_{\hat{j}}^{\hat{t}}$. 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 $(\mathsf{vk}_{\hat{\imath}}, \mathsf{sk}_{\hat{\imath}}) \leftarrow \mathsf{RS.KeyGen}(1^\kappa)$ 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 $\mathcal{B}_{3,1}$ 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.
- $\mathsf{Send}(j, t, m = (\mathsf{ek}_T, \sigma_i))$: 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'. $\mathcal{B}_{3,1}$ 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, $\mathcal{B}_{3,1}$ 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 $\mathcal{B}_{3,1}$ cannot answer.
- $\mathsf{Test}(i, s)$: $\mathcal{B}_4$ responds to the query as in 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 $\pi_{\hat{j}}^{\hat{t}}$. 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 $\mathcal{D}_3$ such that

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

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 $\pi_{\hat{j}}^{\hat{t}}$ 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 $\pi_{\hat{j}}^{\hat{t}}$ (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_i^s 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 π_i^t 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 $\pi_{\hat{j}}^{\hat{t}}$ 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

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

□

For completeness, we state the remaining Lemmata B.3 and B.4 and provide a proof sketch.

Lemma B.3. *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*

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

Lemma B.4. *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 Sketch of Lemmata B.3 and B.4. The 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; and the initiator does not sign the first message. In addition, the former considers weak forward secrecy ($\mathcal{A}$ plays $G^{\mathsf{weakFS}}_{\Pi_{\mathsf{SC\text{-}DAKE}}}(\mu, \ell)$), and the latter considers perfect forward secrecy ($\mathcal{A}$ plays $G^{\mathsf{FS}}_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mu, \ell)$). However, it can be easily verified that this modification brings no advantage to the adversary following the strategies in the statement. In particular, when $\mathcal{A}$ uses the Type-1, Type-2, Type-3 or Type-4 strategy (i.e., the tested oracle has a partner oracle), the winning condition (cf. freshness clauses Items 1 to 4) of the two security game is identical. Specifically, the proofs are identical to the proofs of Lemmata A.1 and A.2.

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 A.1 and A.2 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 remove the signature from the first message and 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 A.1 and A.2 follow. □

C Equivalence Between DVS and Ring Signature

In a subsequent work, Brendel et al. [21] showed a generic construction of a deniable Signal-conforming AKE protocol based on a designated verifier signature (DVS) and a KEM. They showed how to instantiate DVS

from a ring signature (for a ring of two users) but left open the opposite implication and speculated the possibility of constructing DVS easier than a ring signature.

In this section, we solve this open problem. We show how to instantiate a ring signature (for a ring of two users) from DVS and show that DVS is a ring signature in disguise. As discussed in Footnote 10, the security notion of DVS and ring signatures may come in different flavors so it is not always the case that they are equivalent. We only focus on DVS and ring signatures that Brendel et al. [21] required to construct their AKE protocol. Namely, the definition of ring signature we provide in Section 2.6 is strictly stronger than those considered in [21]. We make this clear when we provide the security proof of our ring signature based on DVS.

The following syntax and security definition of DVS is taken almost verbatim from [21, Section 3]. One thing to keep in mind is that even though it is called *designated verifier*, the syntax of Brendel et al. allows the signature to be publicly verifiable. This will be essential when building a ring signature.

Definition C.1. *A* DVS *is a tuple of algorithms* $\mathsf{DVS} = (\mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Sim})$ *along with a message space* $\mathcal{M}$*.*

- $\mathsf{SKGen}() \to (\mathsf{pk_S}, \mathsf{sk_S})$*: A probabilistic key generation algorithm that outputs a public-/secret-key pair for the* signer*.*
- $\mathsf{VKGen}() \to (\mathsf{pk_D}, \mathsf{sk_D})$*: A probabilistic key generation algorithm that outputs a public-/secret-key pair for the* verifier*.*
- $\mathsf{Sign}(\mathsf{sk_S}, \mathsf{pk_D}, \mathsf{M}) \to \sigma$*: A probabilistic signing algorithm that uses a signer's secret key* $\mathsf{sk_S}$ *to produce a signature* σ *for a message* $\mathsf{M} \in \mathcal{M}$ *for a designated verifier with public key* $\mathsf{pk_D}$*.*
- $\mathsf{Vrfy}(\mathsf{pk_S}, \mathsf{pk_D}, \mathsf{M}, \sigma) \to 1/0$*: A deterministic verification algorithm that checks a message* M *and a signature* σ *against a signer's public key* $\mathsf{pk_S}$ *and a verifier's public key* $\mathsf{pk_D}$*.*
- $\mathsf{Sim}(\mathsf{pk_S}, \mathsf{sk_D}, \mathsf{M}) \to \sigma$*: A probabilistic signature simulation algorithm that uses the verifier's secret key* $\mathsf{sk_D}$ *to produce a signature* σ *on a message* M *for signer's public key* $\mathsf{pk_S}$*.*

Definition C.2 (Unforgeability). *A* DVS *is unforgeable if for any efficient adversary* $\mathcal{A}$ *we have* $\Pr[G^{\mathsf{uf}}(\mathcal{A}) = 1]$ *is negligible, where the game* G^{uf} *is defined in Figure 6.*

Definition C.3 (Source Hiding). *A* DVS *is source hiding if for any efficient adversary* $\mathcal{A}$ *we have* $\left|\Pr[G^{\mathsf{srchid}}(\mathcal{A}) = 1] - 1/2\right|$ *is negligible, where the game* G^{srchid} *is defined in Figure 6.*

Using a standard hybrid argument, we can assume without loss of generality that $\mathcal{A}$ queries oracle `Chall` once.

Construction. We now provide a generic construction of a ring signature from any DVS satisfying the above syntax and security definitions. Following Brendel et al. [21], we assume there is no public parameter and omit RS.Setup. We also only consider a ring signature for a ring of two users as this is sufficient to construct an AKE protocol. Moreover, we assume without loss of generality that $\mathsf{pk_S}$ can be ordered lexicographically, e.g., $\mathsf{pk_S} < \mathsf{pk_S}'$.

$\mathsf{RS.KeyGen}()$: Run $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{SKGen}()$ and $(\mathsf{pk_D}, \mathsf{sk_D}) \leftarrow \mathsf{VKGen}()$, and output $(\mathsf{RS.vk} := (\mathsf{pk_S}, \mathsf{pk_D}), \mathsf{RS.sk} := (\mathsf{sk_S}, \mathsf{sk_D}))$.

$\mathsf{RS.Sign}(\mathsf{RS.sk}, \mathsf{M}, \mathsf{R} = \{\mathsf{RS.vk}, \mathsf{RS.vk}'\})$: Parse $(\mathsf{pk_S}, \mathsf{pk_D}) \leftarrow \mathsf{RS.vk}$ and $(\mathsf{pk_S}', \mathsf{pk_D}') \leftarrow \mathsf{RS.vk}'$. If $\mathsf{pk_S} < \mathsf{pk_S}'$, then output $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk_S}, \mathsf{pk_D}', \mathsf{M})$. Otherwise, output $\sigma \leftarrow \mathsf{Sim}(\mathsf{pk_S}', \mathsf{sk_D}, \mathsf{M})$.

$\mathsf{RS.Verify}(\mathsf{R} = \{\mathsf{RS.vk}, \mathsf{RS.vk}'\}, \mathsf{M}, \sigma)$: Parse $(\mathsf{pk_S}, \mathsf{pk_D}) \leftarrow \mathsf{RS.vk}$ and $(\mathsf{pk_S}', \mathsf{pk_D}') \leftarrow \mathsf{RS.vk}'$. If $\mathsf{pk_S} < \mathsf{pk_S}'$, then output $\mathsf{Vrfy}(\mathsf{pk_S}, \mathsf{pk_D}', \mathsf{M}, \sigma)$. Otherwise, output $\mathsf{Vrfy}(\mathsf{pk_S}', \mathsf{pk_D}, \mathsf{M}, \sigma)$.

$G^{\mathsf{uf}}(\mathcal{A})$

1 : $Q \leftarrow \emptyset$
2 : $L \leftarrow \emptyset$
3 : $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{SKGen}()$
4 : $(\mathsf{pk_D}, \mathsf{sk_D}) \leftarrow \mathsf{VKGen}()$
5 : **for** $i \in [n]$ **do**
6 : $\quad (\mathsf{pk}_{\mathsf{D},i}, \mathsf{sk}_{\mathsf{D},i}) \leftarrow \mathsf{VKGen}()$
7 : $\quad L \leftarrow L \cup \{(\mathsf{pk}_{\mathsf{D},i}, \mathsf{sk}_{\mathsf{D},i})\}$
8 : $(\mathsf{M}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathtt{Sign}(\cdot,\cdot)}(\mathsf{pk_S}, \mathsf{pk_D}, L)$
9 : $d \leftarrow \mathsf{Vrfy}(\mathsf{pk_S}, \mathsf{pk_D}, \mathsf{M}^*, \sigma^*)$
10 : **return** $[d = 1 \wedge \mathsf{M}^* \notin Q]$

$\mathtt{Sign}(\mathsf{pk}, \mathsf{M})$

1 : **if** $\mathsf{pk} = \mathsf{pk_D}$ **then**
2 : $\quad Q \leftarrow Q \cup \{\mathsf{M}\}$
3 : **elseif** $(\mathsf{pk}, \mathsf{sk}) \notin L$ **then**
4 : $\quad$ **return** $\perp$
5 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk_S}, \mathsf{pk}, \mathsf{M})$
6 : **return** σ

$G^{\mathsf{srchid}}(\mathcal{A})$

1 : $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{SKGen}()$
2 : $(\mathsf{pk_D}, \mathsf{sk_D}) \leftarrow \mathsf{VKGen}()$
3 : $b \leftarrow \{0, 1\}$
4 : $b' \leftarrow \mathcal{A}^{\mathtt{Chall}(\cdot)}(\mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_D}, \mathsf{sk_D})$
5 : **return** $[b = b']$

$\mathtt{Chall}(\mathsf{M})$

1 : **if** $b = 0$ **then**
2 : $\quad \sigma \leftarrow \mathsf{Sign}(\mathsf{sk_S}, \mathsf{pk_D}, \mathsf{M})$
3 : **else**
4 : $\quad \sigma \leftarrow \mathsf{Sim}(\mathsf{pk_S}, \mathsf{sk_D}, \mathsf{M})$
5 : **return** σ

Figure 6: Unforgeability and source hiding for DVS.

Security. We first prove anonymity of the ring signature. The anonymity definition considered by Brendel et al. [21] is almost identical to those in Definition 2.14 except that they additionally consider the verification and signing keys to be generated honestly, rather than being generated by possibly malicious randomness. This suffices to prove their deniability since the AKE keys are assumed to be generated honestly.

Lemma C.4. *If* DVS *satisfies source hiding, then the ring signature is anonymous (with respect to honestly generated verification and signing keys with rings of size two).*

Proof. Assume there exists an adversary $\mathcal{B}$ against the anonymity of the ring signature. We construct an adversary $\mathcal{A}$ against the source hiding of DVS as follows.

$\mathcal{A}$ is provided $(\mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_D}, \mathsf{sk_D})$ from the DVS challenger. It queries M to oracle $\mathtt{Chall}$ and receives σ. It then generates $(\overline{\mathsf{pk_S}}, \overline{\mathsf{sk_S}}) \leftarrow \mathsf{SKGen}()$ and $(\overline{\mathsf{pk_D}}, \overline{\mathsf{sk_D}}) \leftarrow \mathsf{VKGen}()$ conditioned on $\mathsf{pk_S} < \overline{\mathsf{pk_S}}$. Note that this is without loss of generality since $\mathcal{A}$ can simply regenerate $\overline{\mathsf{pk_S}}$ until it succeeds (and possibly halt after it exceeds some number of trials to make $\mathcal{A}$ run in strict polynomial time). It then samples a random bit $d \leftarrow \{0, 1\}$ and sets

$$(\mathsf{RS.vk}_d, \mathsf{RS.sk}_d) := ((\mathsf{pk_S}, \overline{\mathsf{pk_D}}), (\mathsf{sk_S}, \overline{\mathsf{sk_D}})) \text{ and } (\mathsf{RS.vk}_{1-d}, \mathsf{RS.sk}_{1-d}) := ((\overline{\mathsf{pk_S}}, \mathsf{pk_D}), (\overline{\mathsf{sk_S}}, \mathsf{sk_D})).$$

It finally provides $\mathcal{B}$ with $\{(\mathsf{RS.vk}_i, \mathsf{RS.sk}_i)\}_{i \in \{0,1\}}$ and σ. When $\mathcal{B}$ outputs d' as its guess, $\mathcal{A}$ outputs its guess as $b' := d \oplus d'$.

Let us analyze the advantage of $\mathcal{A}$. First of all, since d is information theoretically hidden from $\mathcal{B}$, the ring signature keys $\{(\mathsf{RS.vk}_i, \mathsf{RS.sk}_i)\}_{i \in \{0,1\}}$ are distributed identically to the anonymity game even conditioned on $\mathsf{pk_S} < \overline{\mathsf{pk_S}}$. Moreover, if oracle $\mathtt{Chall}$ was using $b = 0$, then $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk_S}, \mathsf{pk_D}, \mathsf{M})$. Since $\mathsf{pk_S} < \overline{\mathsf{pk_S}}$, σ is distributed identical to $\mathsf{RS.Sign}(\mathsf{RS.sk}_d, \mathsf{M}, \mathsf{R} = \{\mathsf{RS.vk}_0, \mathsf{RS.vk}_1\})$. On the other hand, if oracle $\mathtt{Chall}$ was using $b = 1$, then $\sigma \leftarrow \mathsf{Sim}(\mathsf{pk_S}, \mathsf{sk_D}, \mathsf{M})$. Then, this is distributed identical to $\mathsf{RS.Sign}(\mathsf{RS.sk}_{1-d}, \mathsf{M}, \mathsf{R} = \{\mathsf{RS.vk}_0, \mathsf{RS.vk}_1\})$. Hence, if $\mathcal{B}$ outputs a guess d' and $d = 0$, then $\mathcal{A}$ simply needs to output d' as its guess. Otherwise, $\mathcal{A}$ flips the guess d' in order to uncompute the swap induced by $d = 1$. This completes the proof. □

The unforgeability definition considered by Brendel et al. [21] is similar to those in Definition 2.15 except that they restrict the adversary to only query the signing oracle on rings consisting of honestly generated verification keys. This weaker definition suffices for their application since they consider deniability only against honestly generated long-term keys.

Lemma C.5. *If* DVS *satisfies source hiding and unforgeability, then the ring signature is unforgeable (with respect to honestly generated rings of size two).*

Proof. Before providing the reduction, we first modify the unforgeability game of the ring signature (with respect to honestly generated rings of size two) to the following. The modification from the original Definition 2.15 is underlined in black. The only difference is that the challenger samples two random distinct indices $(i_0^*, i_1^*) \in [N] \times [N]$ and hopes that the adversary outputs a forgery on the ring $\{\mathsf{RS.vk}_{i_0^*}, \mathsf{RS.vk}_{i_1^*}\}$. Moreover, whenever the adversary $\mathcal{A}$ queries the signing oracle, the challenger will never use $\mathsf{RS.sk}_{i_0^*}$ to sign the message.

(i) The challenger generates key pairs $(\mathsf{RS.vk}_i, \mathsf{RS.sk}_i) = \mathsf{RS.KeyGen}()$ for all $i \in [N]$. It sets $\mathsf{VK} := \{\mathsf{RS.vk}_i \mid i \in [N]\}$ and initializes two empty sets SL and CL. It also samples two distinct $(i_0^*, i_1^*) \leftarrow [N] \times [N]$.

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

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

- $(\mathsf{sign}, i, \mathsf{M}, \mathsf{R} = \{\mathsf{RS.vk}_i, \mathsf{RS.vk}_j\})$: The challenger checks if $(\mathsf{RS.vk}_i, \mathsf{RS.vk}_j) \subseteq \mathsf{R}$ and outputs $\perp$ if not. Otherwise, if $i = i_0^*$, then it computes the signature $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{RS.sk}_j, \mathsf{M}, \mathsf{R})$. If $i \neq i_0^*$, then it computes the signature $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{RS.sk}_i, \mathsf{M}, \mathsf{R})$. Finally, the challenger provides σ to $\mathcal{A}$ and adds $(i, \mathsf{M}, \mathsf{R})$ to SL;
- $(\mathsf{corrupt}, i)$: If $i \in \{i_0^*, i_1^*\}$, then abort the game. Otherwise, the challenger adds $\mathsf{RS.vk}_i$ to CL and returns $\mathsf{RS.sk}_i$ to $\mathcal{A}$.

(iv) $\mathcal{A}$ outputs $(\mathsf{R}^*, \mathsf{M}^*, \sigma^*)$. If $\mathsf{R}^* = \{\mathsf{RS.vk}_{i_0^*}, \mathsf{RS.vk}_{i_1^*}\}$, $(\cdot, \mathsf{M}^*, \mathsf{R}^*) \notin \mathsf{SL}$, and $\mathsf{Verify}(\mathsf{R}^*, \mathsf{M}^*, \sigma^*) = 1$, then we say the adversary $\mathcal{A}$ wins.

It is straightforward to show that this modified unforgeability game is as hard as the original unforgeability game assuming that the ring signature is anonymous (which from Lemma C.4 is an implication of the source hiding of DVS). Concretely, we first modify the original game to a game in which the challenger simply guesses the non-corrupted indices $(i_0^*, i_1^*) \in [N] \times [N]$ that the adversary will use for its forgery. Since these indices are information theoretically hidden from the adversary, this is indistinguishable from the original game (except for a loss of $1/N^2$ in the reduction). Next, assuming $\mathcal{A}$ makes at most Q-queries to the signing oracle, we can define Q-hybrids, where in the k-th hybrid, the challenger answers as in the original game up to the $(k-1)$-th signing query and as in the modified game from the k-th signing query. Each adjacent hybrids $(k-1)$ and k are indistinguishable assuming the anonymity of the ring signature; the reduction samples a random index $j^* \leftarrow [N]$ and embeds its two verification keys provided by the anonymity game in the two indices (i_0^*, j^*). It generates all other verification keys as in the unforgeability game. Note that the reduction knows the signing keys to all parties. It answers all k'-th signing query for $k' \neq k$ as in hybrids $(k-1)$ and k. If $i = i_0^*$ is used in the k-th query, it further checks if vk_{j^*} is used. If so, the reduction simulates the signing oracle by embedding its challenge. If vk_{j^*} is not used, then aborts the game. Otherwise, if $i \neq i_0^*$, then it answers the signing oracle as in the $(k-1)$th and k-th hybrids. This completes the reduction. In case the signature is created using $\mathsf{sk}_{i_0^*}$ (resp. sk_j), it perfectly simulates the $(k-1)$-th (resp. k-th) hybrid (condition on not aborting). Therefore, assuming the ring signature is anonymous, the two hybrids are indistinguishable.

Now, we are ready to show that this modified unforgeability for the ring signature is hard assuming the unforgeability of the DVS. Assume there exists an adversary $\mathcal{B}$ against the modified unforgeability of the ring signature. We construct an adversary $\mathcal{A}$ against the unforgeability of DVS as follows:

$\mathcal{A}$ is given $\mathsf{pk_S}$, $\mathsf{pk_D}$, and $L = \{(\mathsf{pk}_{\mathsf{D},i}, \mathsf{sk}_{\mathsf{D},i})\}_{i\in[n]}$. It generates $(\overline{\mathsf{pk_D}}, \overline{\mathsf{sk_D}}) \leftarrow \mathsf{VKGen}()$, $(\overline{\mathsf{pk_S}}, \overline{\mathsf{sk_S}}) \leftarrow \mathsf{SKGen}()$, and $(\mathsf{pk}_{\mathsf{S},i}, \mathsf{sk}_{\mathsf{S},i}) \leftarrow \mathsf{SKGen}()$ for $i \in [n]$. It then creates $(n+2)$ pairs of verification key pair for the ring signature $(\mathsf{pk_S}, \overline{\mathsf{pk_D}})$, $(\overline{\mathsf{pk_S}}, \mathsf{pk_D})$, and $\{(\mathsf{pk}_{\mathsf{S},i}, \mathsf{pk}_{\mathsf{D},i})\}_{i\in[n]}$ and randomly permutes them and sets them as $(\mathsf{RS.vk}_i)_{i\in[n+2]}$. Let i_0^* be the index such that $\mathsf{RS.vk}_{i_0^*} := (\overline{\mathsf{pk_S}}, \mathsf{pk_D})$ and i_1^* be the index such that $\mathsf{RS.vk}_{i_1^*} := (\mathsf{pk_S}, \overline{\mathsf{pk_D}})$. $\mathcal{A}$ finally provides $\mathsf{VK} := \{\mathsf{RS.vk}_i \mid i \in [n+2]\}$ to $\mathcal{B}$. Notice $\mathcal{A}$ knows the signing keys for indices in $[n+2] \setminus \{i_0^*, i_1^*\}$, so it can simulate the signing query and corrupt query for any $i \notin \{i_0^*, i_1^*\}$.

Moreover, since $\mathcal{A}$ aborts when $i \in \{i_0^*, i_1^*\}$ is queried to the corruption oracle, it remains to see how $\mathcal{A}$ simulates the signing queries when $i \in \{i_0^*, i_1^*\}$. Due to the modification we made to the unforgeability game, $\mathcal{A}$ never needs to sign using the signing key corresponding to index i_0^*, so it suffices to check the case $i = i_1^*$. Now, when $i = i_1^*$ and $\mathsf{pk}_\mathsf{S} < \mathsf{pk}_{\mathsf{S},j}$, then $\mathcal{A}$ queries its signing oracle and obtains a signature using sk_S. Otherwise, it uses $\overline{\mathsf{sk}_\mathsf{D}}$ to generate $\sigma \leftarrow \mathsf{Sim}(\mathsf{pk}_{\mathsf{S}_j}, \overline{\mathsf{sk}_\mathsf{D}}, \mathsf{M})$. This completes the description of $\mathcal{A}$.

Notice the winning condition of the modified unforgeability of the ring signature and the unforgeability of the DVS is identical. Moreover, since $\mathcal{A}$ randomly permutes the indices in $[n+2]$, $\mathcal{A}$ simulates the distribution of the two indices (i_0^*, i_1^*) perfectly. Therefore, $\mathcal{A}$ has the same advantage as $\mathcal{B}$. This concludes the proof. □