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], and Thomas Prest[3]

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

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

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

J. A. Garay (Ed.): PKC 2021, LNCS 12711, pp. 410–440, 2021.
https://doi.org/10.1007/978-3-030-75248-4_15

1 Introduction

Secure instant messaging (SIM) ensures privacy and security by making sure that only the person you are sending the message to can read the message, a.k.a. end-to-end encryption. With the ever-growing awareness against mass-surveillance of communications, people have become more privacy-aware and the demand for SIM has been steadily increasing. While there have been a range of SIM protocols, the Signal protocol [1] 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 billions. One of the reasons for such popularity is due to the simplicity and the strong security properties it provides, such as forward secrecy and post-compromise secrecy, while simultaneously allowing for the same user experience as any (non-cryptographically secure) instant messaging app.

The Signal protocol consists of two sub-protocols: the X3DH protocol [45] and the double ratchet protocol [44]. The former protocol can be viewed as a type of key exchange protocol allowing two parties to exchange a secure initial session key. The latter protocol is executed after the X3DH protocol and it allows two parties to perform a secure back-and-forth message delivery. Below, we briefly recall the current affair of these two protocols.

The Double Ratchet Protocol. The first attempt at a full security analysis of the Signal protocol was made by Cohn-Gordon et al. [18,19]. 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 [2,9,26,36,37,49]. Under these works, our understanding of the double ratchet protocol has much matured. Notably, Alwen et al. [2] 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 [45] and those indirectly considered by Cohn-Gordon et al. [18,19], works focusing on the X3DH protocol seems to be limited. As far as we are aware, there is one recent work that studies the formalization [14] and a few papers that study one of the appealing security properties, known as (off-line) *deniability*, claimed by the X3DH protocol [51–53].

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

Brendel et al. [14] abstract the X3DH protocol and provides the first generic construction based on a new primitive they call a *split key encapsulation mechanism* (KEM). However, so far, instantiations of split KEMs with strong security guarantees required for the X3DH protocol are limited to Diffie-Hellman style assumptions. In fact, the recent result of Guo et al. [33] 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. [53] study one of the security guarantees widely assumed for the X3DH protocol called (off-line) deniability [45, Section 4.4] and showed that a strong knowledge-type assumption would be necessary to formally prove it. Unger and Goldberg [51,52] construct several protocols that can be used as a drop-in replacement of the X3DH protocol that achieves a strong flavor of (on-line) deniability from standard assumptions, albeit by making a noticeable sacrifice in the security against key-compromise attacks: a type of attack that exploits leaked secret information of a party. For instance, while the X3DH protocol is secure against key-compromise impersonation (KCI) attacks [11],[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 [18,19,45], 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 [14] and it is unclear whether such assumptions are inherent to the Signal protocol.
- Ideally, similarly to what Alwen et al. [2] 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).

[2] Although [45, 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 [11], then the X3DH protocol is secure. .

[3] Being vulnerable against KCI attacks seems to be intrinsic to on-line deniability [45, 51,52].

1.1 Our Contribution

In this work, we cast the X3DH protocol (see Fig. 1) as a specific type of authenticated key exchange (AKE) protocol, which we call a *Signal-conforming AKE* protocol, and define its security model based on the vast prior work on AKE protocols. We then provide an efficient generic construction of a Signal-conforming AKE protocol based on standard cryptographic primitives: an (IND-CCA secure) KEM, a signature scheme, and a pseudorandom function (PRF). Since all of these primitives can be based on well-established post-quantum assumptions, this results in the first post-quantum secure replacement of the X3DH protocol. Similarly to the X3DH protocol, our Signal-conforming AKE protocol offers a strong flavor of key-compromise security. Borrowing terminologies from AKE-related literature, our protocol is proven secure in the strong Canetti-Krawczyk (CK) type security models [15,30,39,42], 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 5).[4] We believe the level of security offered by our Signal-conforming AKE protocol aligns with the level of security guaranteed by the double ratchet protocol where (a specific notion of) security still holds even when such secrets are compromised. Moreover, while our Signal-conforming AKE already provides a weak form of deniability, we can strengthen its deniability by using a ring signature scheme instead of a signature scheme. Likewise to the X3DH protocol [53] although our construction seemingly offers (off-line) deniability, the formal proof relies on a strong knowledge-type assumption. However, relying on such assumptions seems unavoidable considering that all known deniable AKE protocols secure against key-compromise attacks, including the X3DH protocol, rely on them [24,53,57].

We implemented our (weakly deniable) Signal-conforming AKE protocol in C, building on the open source libraries PQClean and LibTomCrypt. Our implementation[5] 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 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.

[4] The X3DH can be made secure against leakge of session-specific secrets by using NAXOS trick [42], but it requires additional computation. Because it affects efficiency, we do not consider AKE protocols using NAXOS trick (e.g., [30,40,56]).

[5] It is available at the URL [41].

1.2 Technical Overview

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

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 [10,17,29,50].

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 Fig. 1.[6] As the first step, Alice sends her DH component $g^x \in \mathbb{G}$ to the server[7] and then possibly goes offline. We point out that Alice does *not* need to know who she will be communicating with at this point. Bob, who may ad-hocly decide to communicate with Alice, then fetches Alice's first message from the server and uploads its DH component g^y to the server. As in a typical DH key exchange, Bob computes the session key $\mathsf{k_B}$ using the long-term secret exponent $b \in \mathbb{Z}_p$ and session-specific secret exponent $y \in \mathbb{Z}_p$. Since Bob can compute the session key $\mathsf{k_B}$ while Alice is offline, he can begin executing the subsequent double ratchet protocol without waiting for Alice to come online. Whenever Alice comes online, she can fetch whatever message Bob sent from the server.

Casting the X3DH Protocol as an AKE Protocol. It is not difficult to see that the X3DH protocol can be cast as a specific type of AKE protocol. In particular, we can think of the server as an adversary that tries to mount a man-in-the-middle (MIM) attack in a standard AKE protocol. Viewing the server as a malicious adversary, rather than some semi-honest entity, has two benefits: the parties do not need to put trust in the server since the protocol is supposed

[6] We assume Alice and Bob know each other's long-term key. In practice, this can be enforced by "out-of-bound" authentications (see [45, Section 4.1]).

[7] In the actual protocol, Alice also signs g^x sent to the server (i.e., *signed pre-keys*). We ignore this subtlety as it does not play a crucial role in the analysis of security. See Remark 4.2 for more detail. Also, we note that in practice, Bob may initiate the double ratchet protocol using $\mathsf{k_B}$ and send his message to Alice along with g^y to the server before Alice responds. .

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

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

to be secure even against a malicious server, while the server or the company providing the app is relieved from having to "prove" that it is behaving honestly. One distinguishing feature required by the X3DH protocol when viewed as an AKE protocol is that it needs to be a two-round protocol where the initiator message is generated *independently* from the receiver. That is, Alice needs to be able to store her first message to the server without knowing who she will be communicating with. In this work, we define an AKE protocol with such functionality as a *Signal-conforming* AKE protocol.

Regarding the security model for a Signal-conforming AKE protocol, we base it on the vast prior works on AKE protocols. Specifically, we build on the recent formalization of [20,32] 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 [15,30,39,42], which is a strengthening of the Bellare-Rogaway security model [7] considered by [20,32]. A detailed discussion and comparison between ours and the numerous other security models of AKE protocols are provided in Sect. 3.

Lack of Signal-Conforming AKE Protocol. The main feature of a Signal-conforming AKE protocol is that the initiator's message does *not* depend on the receiver. Although this seems like a very natural feature considering DH-type AKE protocols, it turns out that they are quite unique (see Brendel et al. [14] 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. [22] and Kawashima et al. [38]. Considering the community is still in the process of assessing the concrete parameter selection for *standard* CSIDH [13,48], it would be desirable to base the X3DH protocol on more well-established and versatile assumptions. On the other hand, when we turn our eyes to known generic construction of AKE protocols [30,31,34,54,55] that can be instantiated from versatile assumptions, including post-quantum ones, we observe that none of them is Signal-conforming. That is, they are all either non-2-round or the initiator's message depends on the public key of the receiver.

Our Construction. To this end, in this work, we provide a new practical generic construction of a Signal-conforming AKE protocol from an (IND-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: The construction is simple: Let us assume Alice and Bob's long-term key consist of KEM key pairs $(\mathsf{ek_A}, \mathsf{dk_A})$ and $(\mathsf{ek_B}, \mathsf{dk_B})$ and signature key pairs $(\mathsf{vk_A}, \mathsf{sk_A})$ and $(\mathsf{vk_B}, \mathsf{sk_B})$, respectively. The Signal-conforming AKE protocol then starts by Alice (i.e., the initiator) generating a session-specific KEM key $(\mathsf{ek}_T, \mathsf{dk}_T)$ and sending ek_T to Bob (i.e., the receiver).[9] Here, observe that Alice's message does not depend on who she will be communicating with. Bob then constructs two ciphertexts: one using Alice's long-term key $(\mathsf{K_A}, \mathsf{C_A}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek_A})$ and another using the session-specific key $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_T)$. It then signs these ciphertext $\mathsf{M} := (\mathsf{C_A}, \mathsf{C}_T)$ as $\sigma_\mathsf{B} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk_B}, \mathsf{M})$, where we include other session-specific components in M in the actual construction. Since sending σ_B in the clear may serve as public evidence that Bob communicated with Alice, Bob will hide this. To this end, he derives two keys, a session key $\mathsf{k_{AKE}}$ and a one-time pad key $\mathsf{k_{OTP}}$, by running a key derivation function on input the random KEM keys $(\mathsf{K_A}, \mathsf{K}_T)$. Bob then sends $(\mathsf{C_A}, \mathsf{C}_T, \mathsf{c} := \sigma_\mathsf{B} \oplus \mathsf{k_{OTP}})$ to Alice and sets the session key as $\mathsf{k_{AKE}}$. Once Alice receives the message from Bob, she decrypts the ciphertexts $(\mathsf{C_A}, \mathsf{C}_T)$, derives the two keys $(\mathsf{k_{AKE}}, \mathsf{k_{OPT}})$, and checks if $\sigma := \mathsf{c} \oplus \mathsf{k_{OTP}}$ is a valid signature of Bob's. If so, she sets the session key as $\mathsf{k_{AKE}}$. At a high level, Alice (explicitly) authenticates Bob through verifying Bob's signature and Bob (implicitly) authenticates Alice since Alice is the only party that can decrypt *both* ciphertexts $(\mathsf{C_A}, \mathsf{C}_T)$. We turn this intuition into a formal proof and show that our scheme satisfies a strong flavor of security where the shared session key remains pseudorandom even to an adversary that can obtain any non-trivial combinations of the long-term private keys (i.e., $\mathsf{dk_A}, \mathsf{dk_B}, \mathsf{sk_A}, \mathsf{sk_B}$) and session-specific secret keys (i.e., dk_T). Notably, our protocol satisfies a stronger notion of security compared to the X3DH protocol since it prevents an adversary to

[8] This property has also been called as *post-specified peers* [16] in the context of Internet Key Exchange (IKE) protocols.

[9] As we briefly commented in Footnote 10, Alice can sign her message ek_T as in the X3DH protocol. This will only make our protocol more secure. See Remark 4.2 for more detail.

impersonate Alice even if her session-specific secret key is compromised [45, 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. Although the intuition is clear, it turns out that turning this into a formal proof is quite difficult. Similar to all previous works on AKE protocols satisfying a strong flavor of key-compromise security [24,57] (including the X3DH protocol [53]), the proof of deniability must rely on a strong knowledge-type assumption. We leave it as future work to investigate the deniability of our Signal-conforming AKE protocols from more standard assumptions.

2 Preliminaries

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. Due to page limitation, we refer standard definitions to the full version.

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 Sects. 3.1 to 3.3 a game-based security model building on the recent formalization of [20,32] targeting general AKE protocols. We then discuss in Sect. 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 Sect. 3.5.

3.1 Execution Environment

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

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

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

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 [15,23]. Discussions on other possible choices of the definition for partnering is provide in Sect. 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 $\mathtt{accept}$ 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 $\mathit{\Pi}_{\mathsf{AKE}}$ 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*

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

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

3.2 Security Game

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

Setup: $\mathcal{C}$ first chooses a secret bit $b \leftarrow\!\$ \{0,1\}$. Then $\mathcal{C}$ generates the public parameter of $\mathit{\Pi}_{\mathsf{AKE}}$ 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.[10]
- $\mathsf{RevLTK}(i)$: For $i \in [\mu]$, this query allows $\mathcal{A}$ to learn the long-term secret key lsk_i of party P_i. After this query, P_i is said to be *corrupted*.
- $\mathsf{RegisterLTK}(i, \mathsf{lpk}_i)$: For $i \in \mathbb{N} \setminus [\mu]$, this query allows $\mathcal{A}$ to register a new party P_i with public key lpk_i. We do not require that the adversary knows the corresponding secret key. After the query, the pair (i, lpk_i) is distributed to all other oracles. Parties registered by $\mathsf{RegisterLTK}$ are corrupted by definition.
- $\mathsf{RevState}(i, s)$: This query allows $\mathcal{A}$ to learn the session-state state_i^s of oracle π_i^s. After this query, state_i^s is said to be *revealed*.
- $\mathsf{RevSessKey}(i, s)$: This query allows $\mathcal{A}$ to learn the session key k_i^s of oracle π_i^s.

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

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

[10] 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 Sect. 3.4 for more details.

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

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

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. At this point, the tested oracle must be *fresh*. Here, an oracle π_i^s with $\mathsf{Pid}_i^s = j$[11] is *fresh* if all the following conditions hold:

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

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

The advantage of $\mathcal{A}$ in the security game $G_{\Pi_{\mathsf{AKE}}}(\mu, \ell)$ is defined as $\mathsf{Adv}_{\Pi_{\mathsf{AKE}}}^{\mathsf{AKE}}(\mathcal{A}) := \left|\Pr[b = b'] - \frac{1}{2}\right|$.

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

3.3 Security Properties

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

The freshness clauses Items 1 and 2 imply that we only exclude the reveal of session keys for the tested oracle and its partner oracles. This captures *key independence*; if the revealed keys are different from the tested oracle's key, then such keys must not enable computing the session key. Note that key independence implies resilience to "no-match attacks" presented by Li and Schäge [43]. 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 [12]; a successful UKS attack is a specific type of attack that breaks implicit authentication where two parties compute the same session key but have different views on whom they are communicating with.

The freshness clauses Items 3 to 5 indicate that the game allows the adversary to reveal any subset of the four secret information—the long-term secret keys and the session-states of the two parties (where one party being the party defined by the tested oracle and the other its peer)—except for the combination where both the long-term secret key and session-state of one of the party is revealed. These clauses capture *weak forward secrecy* [39]: the adversary can obtain the

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

long-term secret keys of both parties if it has been passive in the protocol run of the two oracles. Another property captured by our model is resistance to *key-compromise impersonation* (KCI) attacks [11]. 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* [15,30,39,42] 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 [20,32] that do not allow the adversary to learn sessions-states. More discussion on state leakage is provided in Sect. 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 [45] used by the Signal protocol [1]. As explained in Sect. 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 become 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 a substitute for the X3DH protocol.

We abstract this crucial yet implicit property achieved by the X3DH protocol as *receiver obliviousness.*[12]

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 [22,38]) can be checked to be receiver oblivious. In contrast, known generic AKE protocols such as [30,31,34,54,55] are not receiver oblivious since the first message requires the knowledge of the receiver's long-term public key.

[12] This property has also been called as *post-specified peers* [16] in the context of Internet Key Exchange (IKE) protocols.

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

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

3.5 Relation to Other Security Models

In the literature of AKE protocols, many security models have been proposed: the Bellare-Rogaway (BR) model [7], the Canetti-Krawczyk (CK) model [15], the CK+ model [30,39], the extended CK (eCK) model [42], and variants therein [3, 20,21,32,34,35]. Although many of these security models are built based on similar motivations, there are subtle differences. (A comparison between our model and the models listed above can be found in the full version.)

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

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

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

- $\Pi_{\mathsf{KEM}} = (\mathsf{KEM.Setup}, \mathsf{KEM.KeyGen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ is a KEM scheme that is $\mathsf{IND\text{-}CCA}$ secure and assume we have $(1 - \delta_{\mathsf{KEM}})$-correctness.[13]

[13] To prove the security of $\Pi_{\mathsf{SC\text{-}AKE}}$, we require Π_{KEM} and Π_{wKEM} to have high min-entropy of the encapsulation key and the ciphertext.

- $\Pi_{\mathsf{wKEM}} = (\mathsf{wKEM.Setup}, \mathsf{wKEM.KeyGen}, \mathsf{wKEM.Encap}, \mathsf{wKEM.Decap})$ is a KEM schemes that is $\mathsf{IND\text{-}CPA}$ secure (and not $\mathsf{IND\text{-}CCA}$ secure) and assume we have $(1 - \delta_{\mathsf{wKEM}})$-correctness.
- $\Pi_{\mathsf{SIG}} = (\mathsf{SIG.Setup}, \mathsf{SIG.KeyGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ is a signature scheme that is $\mathsf{EUF\text{-}CMA}$ secure and $(1 - \delta_{\mathsf{SIG}})$-correctness. We denote d as the bit length of the signature generated by $\mathsf{SIG.Sign}$.
- $\mathsf{F} : \mathcal{FK} \times \{0,1\}^* \to \{0,1\}^{\kappa+d}$ is a pseudo-random function family with key space $\mathcal{FK}$.
- $\mathsf{Ext} : \mathcal{S} \times \mathcal{KS} \to \mathcal{FK}$ is a strong randomness extractor.

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

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

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

1. Party P_i sets $\mathsf{Pid}_i^s := j$ and $\mathsf{role}_i^s := \mathtt{init}$. P_i computes $(\mathsf{dk}_T, \mathsf{ek}_T) \leftarrow \mathsf{wKEM.KeyGen}(\mathsf{pp}_{\mathsf{wKEM}})$ and sends ek_T to party P_j. P_i stores the ephemeral decapsulation key dk_T as the session-state i.e., $\mathsf{state}_i^s := \mathsf{dk}_T$.[14]
2. Party P_j sets $\mathsf{Pid}_j^t := i$ and $\mathsf{role}_j^t := \mathtt{resp}$. Upon receiving ek_T, P_j first computes $(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_i)$ and $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{wKEM.Encap}(\mathsf{ek}_T)$. Then P_j derives two PRF keys $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K})$, $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$. It then defines the session-identifier as $\mathsf{sid}_j^t := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{C} \| \mathsf{C}_T$ and computes $\mathsf{k}_j \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_j) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_j)$, where $\mathsf{k}_j \in \{0,1\}^\kappa$ and $\tilde{k} \in \{0,1\}^d$, and sets the session key as $\mathsf{k}_j^t := \mathsf{k}_j$. P_j then signs $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_j, \mathsf{sid}_j^t)$ and encrypts it as $\mathsf{c} \leftarrow \sigma \oplus \tilde{k}$. Finally, it sends $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ to P_i and sets $\Psi_j := \mathtt{accept}$. Here, note that P_j does not require to store any session-state, i.e., $\mathsf{state}_j^t = \perp$.
3. Upon receiving $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$, P_i first decrypts $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{dk}_i, \mathsf{C})$ and $\mathsf{K}_T \leftarrow \mathsf{wKEM.Decap}(\mathsf{dk}_T, \mathsf{C}_T)$, and derives two PRF keys $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K})$ and $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$. It then sets the session-identifier as $\mathsf{sid}_i^s := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{C} \| \mathsf{C}_T$ and computes $\mathsf{k}_i \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_i) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_i)$, where $\mathsf{k}_j \in \{0,1\}^\kappa$ and $\tilde{k} \in \{0,1\}^d$. P_i then decrypts $\sigma \leftarrow \mathsf{c} \oplus \tilde{k}$ and checks whether $\mathsf{SIG.Verify}(\mathsf{vk}_j, \mathsf{sid}_i^s, \sigma) = 1$ holds. If not, P_i sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{reject}, \perp, \perp)$ and stops. Otherwise, it sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{accept}, \mathsf{k}_i, \perp)$. Here, note that P_i deletes the session-state $\mathsf{state}_i^s = \mathsf{dk}_T$ at the end of the key exchange.

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

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

Remark 4.2 (Signed Prekeys). In the X3DH protocol, the initiator sends the first message with a signature attached called *signed prekey*. Informally, this allows Bob to *explicitly* authenticate Alice, while otherwise without the signature, Bob can only *implicitly* authenticate Alice. Moreover, this signature enhances the X3DH protocol to be *perfect* forward secret rather than being only *weak* forward secret, where the former allows the adversary to be active in the protocol run of the two oracles. Indeed, according to [45], the X3DH is considered to have perfect forward secrecy. We observe that adding such signature in our protocol has the same effect as long as the added signature is not included in the session-identifier. This is due to Li and Schäge [43, Appendix D], who showed that adding new messages to an already secure protocol cannot lower the security as long as the derived session keys and the session-identifiers remain the same as the original protocol. Here, note the latter implies that the partnering relation remains the same. Similarly, Cremers and Feltz [21] show that adding a signature to the exchanged messages can enhance weak forward secrecy to perfect forward secrecy for natural classes of AKE protocols.

Security. Correctness holds by a routine check. The following establishes the security or $\Pi_{\mathsf{SC\text{-}AKE}}$. We provide a proof overview and refer the full proof to the full version.

Theorem 4.1 (Security of $\Pi_{\mathsf{SC\text{-}AKE}}$). *Assume Π_{wKEM} is $\mathsf{IND\text{-}CPA}$ secure, Π_{KEM} is $\mathsf{IND\text{-}CCA}$ secure, Π_{SIG} is $\mathsf{EUF\text{-}CMA}$ secure, and F is secure PRF. Then $\Pi_{\mathsf{SC\text{-}AKE}}$ is secure AKE protocol with respect to Definition 3.3.*

Proof Sketch. Let $\mathcal{A}$ be an adversary that plays the security game $G_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mu, \ell)$. We distinguish between all possible strategies that can be taken by $\mathcal{A}$. Specifically, $\mathcal{A}$'s strategy can be divided into the eight types of strategies listed in Table 1.

Here, each strategy is mutually independent and covers all possible (non-trivial) strategies. We point out that for our specific AKE construction we have $\mathsf{state}_{\mathsf{resp}} := \perp$ since the responder does not maintain any states (see Remark 4.1). Therefore, the Type-1 (resp. Type-3, Type-7) strategy is strictly stronger than the Type-2 (resp. Type-4, Type-8) strategy. Concretely, for our proof, we only need to consider the following four cases and to show that $\mathcal{A}$ has no advantage in each cases: (a) $\mathcal{A}$ uses the Type-1 strategy; (b) $\mathcal{A}$ uses the Type-3 strategy; (c) $\mathcal{A}$ uses the Type-5 or Type-6 strategy; (d) $\mathcal{A}$ uses the Type-7 strategy.

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.

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	✗	–	✗	✓

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

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

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

In this section, we present the implementation details of our post-quantum Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$. We take existing implementations of post-quantum KEMs and signature schemes submitted for the NIST PQC standardization. To instantiate our Signal-conforming AKE we pair variants of KEMs and signature schemes corresponding to the same security level. We consider security levels 1, 3 and 5 as defined by NIST for the PQC standardization. With more than 30 variants of KEM and 13 variants of signature schemes, we can create at least 128 different instantiations of post-quantum Signal-conforming AKE protocols. The provided implementation simulates post-quantum, weakly deniable authenticated key exchange between two entities. We study the efficiency of our instantiations through two metrics—the total amount of data exchanged between parties and run-time performance. Our implementation is available at the URL [41].

5.1 Instantiation Details

Our implementation is instantiated with the following building blocks:

- s: (pseudo)-randomly generated 32 bytes of data calculated at session initialization phase,
- Ext_s: uses HMAC-SHA256 as a strong randomness extractor. As an input message we use a key K_T prepended with byte `0x02` which works as a domain separator (since we also use HMAC-SHA256 as a PRF). Security of using HMAC as a strong randomness extractor is studied in [28],
- 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 [4],
- b: depends on the security level of the underlying post-quantum KEM scheme, where $b \in \{128, 192, 256\}$,
- d: depends on the byte length of the signature generated by the post-quantum signature scheme $\varPi_{\mathsf{SIG}}$,
- $\varPi_{\mathsf{KEM}}$, $\varPi_{\mathsf{wKEM}}$, $\varPi_{\mathsf{SIG}}$: to instantiate $\varPi_{\mathsf{SC\text{-}AKE}}$, implementation uses pairs of KEM and signature schemes. List of the schemes used can be found in the table (Table 2) below. We always use the same KEM scheme for $\varPi_{\mathsf{KEM}}$ and $\varPi_{\mathsf{wKEM}}$.

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

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

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

Finally, we use an implementation of post-quantum algorithms that can be found in libOQS[15]. We also use LibTomCrypt[16] which provides an implementation of the building blocks HMAC, HKDF and SHA-256.

5.2 Efficiency Analysis

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

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

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

For benchmarking, we modeled a scenario in which two parties try to establish a session key. Alice generates and makes her long-term public key $\mathsf{lpk}_{\mathsf{A}}$ and ephemeral KEM key ek_T publicly available. Bob retrieves the pair $(\mathsf{lpk}_{\mathsf{A}}, \mathsf{ek}_T)$ and uses it to perform his part of the session establishment. Namely, Bob generates the triple $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ and sends it to Alice along with its long-term public key $\mathsf{lpk}_{\mathsf{B}}$. Upon receipt, Alice finalizes the process by computing the session key on her side. We note that in the case of the Signal protocol, both parties communicate with a server (e.g., the Signal server), and not directly. For simplicity, we abstract this fact out of our scenario. Further note that in the Signal protocol, the long-term public keys lpk must be fetched from the server as the parties do not store the keys lpk corresponding to those that they have not communicated with before[17]

Table 3 provides the results for Round 3 candidates of the NIST PQC standardization process.[18] The **CPU cycles** column is related to the computational

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

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

[17] The X3DH protocol assumes the parties authenticate the long-term public keys through some authenticated channel [45, Section 4.1].

[18] The results for all 128 instantiations can be found at the URL [41].

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

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

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

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

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

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

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

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

Depending on the scheme used, increased packet loss may negatively impact session establishment time (see [47]). For example, a scheme instantiated with `Falcon512/NTRU hps2048509` requires exchange of $npacks = 7$ packets over the network, where instantiation with `SPHINCS-SHAKE256-128f-simple/Saber Light` requires 16. Assuming increased packet rate loss of 5%, the probability of losing a packet in the former case is $1 - (1 - rate)^{npacks} = 30\%$, where in the latter it is 56%. In the latter case, at the median, every other session key establishment will experience packet retransmission and hence a delay.

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

In this section, we provide a theory-oriented discussion on the deniability aspect of our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$. In the following, we first informally show that $\Pi_{\mathsf{SC\text{-}AKE}}$ already has a very weak form of deniability that may be acceptable in some applications. We then show that we can slightly modify $\Pi_{\mathsf{SC\text{-}AKE}}$ to satisfy a more stronger notion of deniability. As it is common with all deniable AKE protocols secure against key-compromise attacks [24,53,57], we prove deniability by relying on strong knowledge-type assumptions , including a variant of the *plaintext-awareness* (PA) for the KEM scheme [5,6,8].

Weak Deniability of $\Pi_{\mathsf{SC\text{-}AKE}}$. Our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ already satisfies a weak notion of deniability, where the communication transcript does not leave a trace of the two parties if both parties honestly executed the AKE protocol. Namely, an adversary that is passively collecting the communication transcript cannot convince a third party that communication between two parties took place. Informally, this can be observed by checking that all the contents in the transcript can be simulated by the adversary on its own. We discuss a stronger notion of deniability next.

6.1 Definition of Deniability and Tool Preparation

We follow a simplified definition of deniability for AKE protocols introduced by Di Raimondo et al. [24]. Discussion on the simplification is provided in Remark 6.2. Let Π be an AKE protocol and KeyGen be the key generation algorithm. That is, for any integer $\mu = \mu(\kappa)$ representing the number of parties in the system, define $\mathsf{KeyGen}(1^\kappa, \mu) \rightarrow (\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$, where pp is the public parameter used by the system and $\overrightarrow{\mathsf{lpk}} := \{\mathsf{lpk}_i \mid i \in [\mu]\}$ and $\overrightarrow{\mathsf{lsk}} := \{\mathsf{lsk}_i \mid i \in [\mu]\}$ are the corresponding long-term public and secret keys of the μ parties, respectively.

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

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

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

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

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

Remark 6.1 (Including Public Information and Session Keys). It is crucial that the two distributions $\mathcal{F}_{\mathsf{Real}}$ and $\mathcal{F}_{\mathsf{Sim}}$ include the public information $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}})$. Otherwise, $\mathsf{SIM}_{\mathcal{M}}$ can simply create its own set of $(\mathsf{pp}', \overrightarrow{\mathsf{lpk}'}, \overrightarrow{\mathsf{lsk}'})$ and simulate the view to $\mathcal{M}$. However, this does not correctly capture deniability in the real-world since $\mathcal{M}$ would not be able to convince anybody with such a view using public information that it cooked up on its own. In addition, it is essential that the value of the session key is part of the output of $\mathsf{SIM}_{\mathcal{M}}$. This guarantees that the contents of the sessions authenticated by the session key can also be denied.

Remark 6.2 (Comparison between Prior Definition). Our definition is weaker than the deniability notion originally proposed by Di Raimondo et al. [24]. 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 $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$.[19] For instance, this allows to capture information that $\mathcal{M}$ may have obtained by eavesdropping conversations between honest parties (which is not modeled by $\mathsf{View}_{\mathcal{M}}$). Since our goal is to provide a minimal presentation on the deniability of our protocol, we only focus on the weaker definition where $\mathcal{M}$ does not obtain such auxiliary information. We leave it as future work to prove our protocol deniable in the sense of Di Raimondo et al. [24]. We also note that stronger forms of deniability are known and formalized in the universally composable (UC) model [25,51,52], however, AKE protocols satisfying such a strong deniability notion are known to achieve weaker security guarantees. For instance, as noted in [52], an AKE protocol cannot be on-line deniable while also being secure against KCI attacks.

Remark 6.3 (Extending to Malicious Quantum Adversaries). We only consider classical deniability above. Although we can show deniability for semi-honest quantum adversaries, we were not able to do so for malicious quantum adversaries. This is mainly due to the fact that to prove deniability against malicious classical adversaries, we require a strong knowledge type assumption (i.e., plaintext-awareness for KEM) that assumes an extractor can invoke the adversary multiple of times on the *same* randomness. We leave it as an interesting problem to formally define a set of tools that allow to show deniability even against malicious quantum adversaries.

Required Tools. To argue deniability in the following section we rely on the following tools: ring signature, plaintext-aware (PA-1) secure KEM scheme, and

[19] Although in [24, Definition 2], aux is defined as fixed information that $\mathcal{M}$ cannot adaptively choose, we observe that in their proof they implicitly assume that aux is sampled adaptively from some distribution dependent on $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}})$. Such a definition of aux is necessary to invoke PA-2 security of the underlying encryption scheme.

a non-interactive zero-knowledge (NIZK) argument.[20] We use standard notions of ring signatures and NIZK arguments. On the other hand, we use a slightly stronger variant of $\mathsf{PA\text{-}1}$ secure KEM schemes than those originally defined in [5,6,8]. Informally, a KEM scheme is $\mathsf{PA\text{-}1}$ secure if for any adversary $\mathcal{M}$ that outputs a valid ciphertext C, there is an extractor $\mathsf{Ext}_{\mathcal{M}}$ that outputs the matching session key K. In our work, we require $\mathsf{PA\text{-}1}$ security to hold even when $\mathcal{M}$ is given multiple public keys rather than a single public key [46]. We note that although Di Raimondo et al. [24] considered the standard notion of $\mathsf{PA\text{-}1}$ security, we observe that their proof only works in the case where multiple public keys are considered. Finally, we further require the extractor $\mathsf{Ext}_{\mathcal{M}}$ to be efficiently computable given $\mathcal{M}$.

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

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

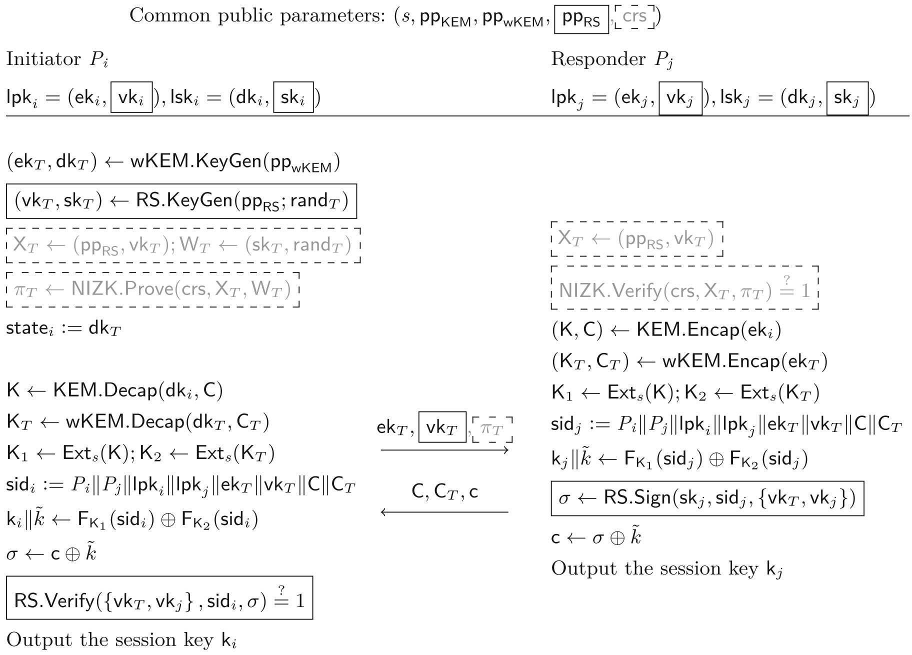
Building Blocks. Our deniable Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ against semi-honest adversaries consists of the following building blocks.

- $\Pi_{\mathsf{KEM}} = (\mathsf{KEM.Setup}, \mathsf{KEM.KeyGen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ is a KEM scheme that is $\mathsf{IND\text{-}CCA}$ secure and assume we have $(1-\delta_{\mathsf{KEM}})$-correctness.[21]
- $\Pi_{\mathsf{wKEM}} = (\mathsf{wKEM.Setup}, \mathsf{wKEM.KeyGen}, \mathsf{wKEM.Encap}, \mathsf{wKEM.Decap})$ is a KEM schemes that is $\mathsf{IND\text{-}CPA}$ secure (and not $\mathsf{IND\text{-}CCA}$ secure) and assume we have $(1-\delta_{\mathsf{wKEM}})$-correctness.
- $\Pi_{\mathsf{RS}} = (\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Verify})$ 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 $\mathsf{RS.Sign}$.
- $\mathsf{F} : \mathcal{FK} \times \{0,1\}^* \rightarrow \{0,1\}^{\kappa+d}$ is a pseudo-random function family with key space $\mathcal{FK}$.
- $\mathsf{Ext} : \mathcal{S} \times \mathcal{KS} \rightarrow \mathcal{FK}$ is a strong randomness 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 $\mathsf{X.Setup}$.

[20] Due to the page limitation, the formal definitions of these tools are provided in the full version.

[21] Similar to $\Pi_{\mathsf{SC\text{-}AKE}}$, to prove the security of $\Pi_{\mathsf{SC\text{-}DAKE}}$, we require Π_{KEM} and Π_{wKEM} to have high min-entropy of the encapsulation key and the ciphertext.



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

Initiator P_i

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

Responder P_j

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

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

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

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

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

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

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

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

$\mathsf{NIZK.Verify}(\mathsf{crs}, \mathsf{X}_T, \pi_T) \stackrel{?}{=} 1$

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

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

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

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

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

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

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

Output the session key k_j

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

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

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

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

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

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

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

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

Output the session key k_i

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

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 Fig. 2. More formally, we have the following.

1. Party P_i sets $\mathsf{Pid}_i^s := j$ and $\mathsf{role}_i^s := \texttt{init}$. 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., $\mathsf{state}_i^s := \mathsf{dk}_T$.[22]
2. Party P_j sets $\mathsf{Pid}_j^t := i$ and $\mathsf{role}_j^t := \texttt{resp}$. 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

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

$\mathsf{k}_j \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_j) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_j)$, 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 the full version. The main difference from the security proof of $\Pi_{\mathsf{SC\text{-}AKE}}$ is that we have to make sure that using a ring signature instead of a standard signature does not allow the adversary to mount a key-compromise impersonation (KCI) attack (see Sect. 3.3 for the explanation on KCI attacks).

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

Theorem 6.1 (Deniability of $\Pi_{\mathsf{SC\text{-}DAKE}}$ against Semi-Honest Adversaries). *Assume Π_{RS} is anonymous. Then, the Signal-conforming protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ is deniable against semi-honest adversaries.*

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

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

Combining everything together, we conclude the proof. □

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

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

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

- $\Pi_{\mathsf{KEM}} = (\mathsf{KEM.Setup}, \mathsf{KEM.KeyGen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ is an $\mathsf{IND\text{-}CCA}$ secure KEM scheme as in the previous section that additionally satisfies $\mathsf{PA}_{\mu}\text{-}1$ security with an efficiently constructible extractor, where μ is the number of parties in the system.
- $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ is a NIZK argument system for the relation $\mathcal{R}_{\mathsf{RS}}$ where $(\mathsf{X}, \mathsf{W}) \in \mathcal{R}_{\mathsf{RS}}$ if and only if the statement $\mathsf{X} = (\mathsf{pp}, \mathsf{vk})$ and witness $\mathsf{W} = (\mathsf{sk}, \mathsf{rand})$ satisfy $(\mathsf{vk}, \mathsf{sk}) = \mathsf{RS.KeyGen}(\mathsf{pp}; \mathsf{rand})$.

Additional Assumption. We require a knowledge-type assumption to prove deniability against malicious adversaries. Considering that all of the previous AKE protocols satisfying a strong form of security and deniability require such knowledge-type assumptions [24,53,57], this seems unavoidable. On the other hand, there are protocols achieving a strong form of deniability from standard assumptions [25,51,52], 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. [24] 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[23] like those provide by Fischlin's NIZK [27]. We leave it to future works to investigate the credibility of the following assumption and those required to prove deniability of the X3DH protocol [53].

Assumption 6.2 (Key-Awareness Assumption for $\Pi'_{\mathsf{SC\text{-}DAKE}}$). *We say that* $\Pi'_{\mathsf{SC\text{-}DAKE}}$ *has the* key-awareness property *if for all PPT adversaries* $\mathcal{M}$ *interacting with a real protocol execution in the deniability game as in Definition 6.1, there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{M}}$ *such that for any choice of* $(\mathsf{pp}, \overrightarrow{\mathsf{lpk}}, \overrightarrow{\mathsf{lsk}}) \in \mathsf{KeyGen}(1^\kappa, \mu)$*, whenever* $\mathcal{M}$ *outputs a ring signature verification key* vk *and a* NIZK *proof* π *for the language* $\mathcal{L}_{\mathsf{RS}}$*, then* $\mathsf{Ext}_{\mathcal{M}}$ *taking input the same input as* $\mathcal{M}$ *(including its randomness) outputs a signing key* sk *such that* $(\mathsf{vk}, \mathsf{sk}) \in \mathsf{RS.KeyGen}(\mathsf{pp}_{\mathsf{RS}})$ *for any* $\mathsf{pp}_{\mathsf{RS}} \in \mathsf{RS.Setup}(1^\kappa)$*.*

With the added building blocks along with the key-awareness assumption, we prove the following theorem. The high-level approach is similar to the previous proof against semi-honest adversaries but the concrete proof requires 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 non-uniform adversary. The proof is presented in the full version.

Theorem 6.3 (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,*[24] *and the key-awareness assumption in Assumption 6.2 holds. Then, the Signal-conforming protocol* $\Pi'_{\mathsf{SC\text{-}DAKE}}$ *with* μ *parties is deniable against malicious adversaries.*

Finally, we show $\Pi'_{\mathsf{SC\text{-}DAKE}}$ is correct and secure as a standard Signal-conforming AKE protocol in the full version.

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).

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

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

References

1. Signal protocol: Technical documentation. https://signal.org/docs/
2. Alwen, J., Coretti, S., Dodis, Y.: the double ratchet: security notions, proofs, and modularization for the signal protocol. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 129–158. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_5
3. Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 629–658. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_26
4. Bellare, M.: New proofs for NMAC and HMAC: security without collision-resistance. Cryptology ePrint Archive, Report 2006/043
5. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055718
6. Bellare, M., Palacio, A.: Towards plaintext-aware public-key encryption without random oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30539-2_4
7. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_21
8. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0053428
9. Bellare, M., Singh, A.C., Jaeger, J., Nyayapati, M., Stepanovs, I.: Ratcheted encryption and key exchange: the security of messaging. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 619–650. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_21
10. Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006). https://doi.org/10.1007/11745853_14
11. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 30–45. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0024447
12. Blake-Wilson, S., Menezes, A.: Unknown key-share attacks on the station-to-station (STS) protocol. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 154–170. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49162-7_12
13. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 493–522. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_17
14. Brendel, J., Fischlin, M., Günther, F., Janson, C., Stebila, D.: Towards post-quantum security for signal's X3DH handshake. In: SAC 2020
15. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_28

16. Canetti, R., Krawczyk, H.: Security analysis of IKE's signature-based key-exchange protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_10
17. Cash, D., Kiltz, E., Shoup, V.: The Twin Diffie-Hellman problem and applications. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_8
18. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. In: IEEE European Symposium on Security and Privacy (EuroS&P), pp. 451–466
19. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. J. Cryptol. 1–70
20. Cohn-Gordon, K., Cremers, C., Gjøsteen, K., Jacobsen, H., Jager, T.: Highly efficient key exchange protocols with optimal tightness. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 767–797. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_25
21. Cremers, C., Feltz, M.: Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 734–751. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33167-1_42
22. de Kock, B., Gjøsteen, K., Veroni, M.: Practical isogeny-based key-exchange with optimal tightness. In: SAC 2020
23. de Saint Guilhem, C.D., Fischlin, M., Warinschi, B.: Authentication in key-exchange: definitions, relations and composition. In: 2020 IEEE 33rd Computer Security Foundations Symposium (CSF), pp. 288–303
24. Di Raimondo, M., Gennaro, R., Krawczyk, H.: Deniable authentication and key exchange. In: ACM CCS, pp. 400–409 (2006)
25. Dodis, Y., Katz, J., Smith, A., Walfish, S.: Composability and on-line deniability of authentication. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 146–162. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_10
26. Durak, F.B., Vaudenay, S.: Bidirectional asynchronous ratcheted key agreement with linear complexity. In: Attrapadung, N., Yagi, T. (eds.) IWSEC 2019. LNCS, vol. 11689, pp. 343–362. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26834-3_20
27. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_10
28. Fouque, P.-A., Pointcheval, D., Zimmer, S.: HMAC is a randomness extractor and applications to TLS. In: ASIACCS 2008, pp. 21–32
29. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 254–271. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_17
30. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 467–484. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_28
31. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In: ASIACCS 2013, pp. 83–94

32. Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 95–125. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_4
33. Guo, S., Kamath, P., Rosen, A., Sotiraki, K.: Limits on the efficiency of (Ring) LWE based non-interactive key exchange. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 374–395. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_13
34. Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 389–422. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_14
35. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. Cryptology ePrint Archive, Report 2020/1279
36. Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: almost-optimal guarantees for secure messaging. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 159–188. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_6
37. Jost, D., Maurer, U., Mularczyk, M.: A unified and composable take on ratcheting. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 180–210. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36033-7_7
38. Kawashima, T., Takashima, K., Aikawa, Y., Takagi, T.: An efficient authenticated key exchange from random self-reducibility on CSIDH. In: Hong, D. (ed.) ICISC 2020. LNCS, vol. 12593, pp. 58–84. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-68890-5_4
39. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_33
40. Kurosawa, K., Furukawa, J.: 2-pass key exchange protocols from CPA-secure KEM. In: CT-RSA, pp. 385–401 (2014)
41. Kwiatkowski, K.: Signal-conforming AKE protocol implementation. https://github.com/post-quantum-cryptography/post-quantum-state-leakage-secure-ake
42. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75670-5_1
43. Li, Y., Schäge, S.: No-match attacks and robust partnering definitions: defining trivial attacks for security protocols is not trivial. In: ACM CCS, pp. 1343–1360 (2017)
44. Marlinspike, M., Perrin, T.: The double ratchet algorithm. https://signal.org/docs/specifications/doubleratchet/
45. Marlinspike, M., Perrin, T.: The X3DH key agreement protocol. https://signal.org/docs/specifications/x3dh/
46. Myers, S., Sergi, M., Shelat: Blackbox construction of a more than non-malleable CCA1 encryption scheme from plaintext awareness. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 149–165. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32928-9_9
47. Paquin, C., Stebila, D., Tamvada, G.: Benchmarking post-quantum cryptography in TLS. Cryptology ePrint Archive, Report 2019/1447

48. Peikert, C.: He gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 463–492. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_16
49. Poettering, B., Rösler, P.: Towards bidirectional ratcheted key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 3–32. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_1
50. Pointcheval, D., Sanders, O.: Forward secure non-interactive key exchange. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 21–39. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_2
51. Unger, N., Goldberg, I.: Deniable key exchanges for secure messaging. In: ACM CCS, pp. 1211–1223 (2015)
52. Unger, N., Goldberg, I.: Improved strongly deniable authenticated key exchanges for secure messaging. PoPETs **1**, 21–66 (2018)
53. Vatandas, N., Gennaro, R., Ithurburn, B., Krawczyk, H.: On the cryptographic deniability of the signal protocol. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 2020, Part II. LNCS, vol. 12147, pp. 188–209. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57878-7_10
54. Xue, H., Au, M.H., Yang, R., Liang, B., Jiang, H.: Compact authenticated key exchange in the quantum random oracle model. Cryptology ePrint Archive, Report 2020/1282
55. Xue, H., Lu, X., Li, B., Liang, B., He, J.: Understanding and constructing AKE via double-key key encapsulation mechanism. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 158–189. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_6
56. Yang, Z., Chen, Y., Luo, S.: Two-message key exchange with strong security from ideal lattices. In: CT-RSA, pp. 98–115 (2018)
57. Yao, A.C., Zhao, Y.: Deniable internet key exchange. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 329–348. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13708-2_20