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

Keitaro Hashimoto[1,2], Shuichi Katsumata[2,3], Kris Kwiatkowski[3], Thomas Prest[4]

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

September 15, 2022

Abstract

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

In this work, we cast the X3DH protocol as a specific type of authenticated key exchange (AKE) protocol, which we call a *Signal-conforming AKE* protocol, and formally define its security model based on the vast prior works on AKE protocols. We then provide the first efficient generic construction of a Signal-conforming AKE protocol based on standard cryptographic primitives such as key encapsulation mechanisms (KEM) and signature schemes. This results in the first post-quantum secure replacement of the X3DH protocol based on well-established assumptions. Combined with a post-quantum secure double ratchet protocol, this leads to the first post-quantum secure Signal protocol.

While our first protocol already satisfies a weak flavor of deniability, we show how to progressively strengthen it using ring signatures in a second protocol, then by adding non-interactive zero-knowledge proof systems in a third protocol. Finally, we provide a full-fledged, generic C implementation of our first protocol. We instantiate it with the currently selected NIST PQC standards (Kyber, Dilithium, Falcon and SPHINCS$^+$) and compare the resulting bandwidth and computation performances. Our implementation is publicly available in Github.

1 Introduction

Secure instant messaging (SIM) ensures privacy and security by making sure that only the person you are sending the message to can read the message, a.k.a. end-to-end encryption. With the ever-growing awareness of mass-surveillance of communications, people have become more privacy-aware and the demand for SIM has been steadily increasing. While there has been a range of SIM protocols, the Signal protocol [2] is widely regarded as the gold standard. Not only is it used by the Signal app[1], the Signal protocol is also used by WhatsApp, Skype, Facebook Messenger among many others, where the number of active users is well over 2 billion. One of the reasons for such popularity is due to the simplicity and the strong security properties it provides, such as forward secrecy and post-compromise secrecy, while simultaneously allowing for the same user experience as any (non-cryptographically secure) instant messaging app.

The Signal protocol consists of two sub-protocols: the X3DH protocol [56] and the Double Ratchet (DR) protocol [55]. The former protocol can be viewed as a type of key exchange protocol allowing two parties to exchange a secure initial session key. The latter protocol is executed after the X3DH protocol and it allows two parties to perform a secure back-and-forth message delivery. Below, we briefly recall the current state of these two protocols.

The Double Ratchet Protocol. The first attempt at a full security analysis of the Signal protocol was made by Cohn-Gordon et al. [22, 23]. They considered the Signal protocol as a monolithic protocol. Since the DR 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 DR protocol by viewing it as a stand-alone protocol [9, 61, 3, 31, 44, 45]. Notably, Alwen et al. [3] fully abstracted the complex Diffie-Hellman based DR 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* DR protocol.

The X3DH Protocol. In contrast, other than the white paper offered by Signal [56] and those indirectly considered by Cohn-Gordon et al. [22, 23], works focusing on the X3DH protocol are limited. As far as we are aware, there is one recent work that studies the formalization [17] and a few papers that study one of the appealing security properties, known as (off-line) *deniability*, claimed by the X3DH protocol [66, 64, 65].

Brendel et al. [17] abstract the X3DH protocol and provide the first generic construction based on a new primitive they call a *split key encapsulation mechanism* (split KEM). However, so far, instantiations of split KEMs with strong security guarantees required for the X3DH protocol are limited to Diffie-Hellman style assumptions. In fact, the recent result of Guo et al. [38] 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. [66] study one of the security guarantees widely assumed for the X3DH protocol called (off-line) deniability [56, Section 4.4] and showed that a strong knowledge-type assumption would be necessary to formally prove it. Unger and Goldberg [64, 65] construct several protocols that can be used as drop-in replacements of the X3DH protocol that achieve a strong flavor of (on-line) deniability from standard assumptions, albeit by making a noticeable sacrifice in the security against key-compromise attacks: a type of attack that exploits leaked secret information of a party. For instance, while the X3DH protocol is secure against key-compromise impersonation (KCI) attacks [13],[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 [56, 22, 23], 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 with similar protocols without a clearly defined security model.

- The X3DH protocol is so far only instantiable from Diffie-Hellman style assumptions [17] and it is unclear whether such assumptions are inherent to the Signal protocol.

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

[2]Although [56, 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 [13], then the X3DH protocol is secure.

[3]Being vulnerable against KCI attacks seems to be intrinsic to on-line deniability [64, 65, 56].

- Ideally, similarly to what Alwen et al. [3] did for the DR 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 DR protocol is, we cannot completely secure the Signal protocol if the initial X3DH protocol is the weakest link in the chain (e.g., insecure against state-leakage and only offering security against classical adversaries).

1.1 Our Contribution

In this work, we cast the X3DH protocol (see Figure 1) as a specific type of authenticated key exchange (AKE) protocol, which we call a *Signal-conforming AKE* protocol, and define its security model based on the vast prior work on AKE protocols (see Section 2). We then provide an efficient generic construction of a Signal-conforming AKE protocol based on standard cryptographic primitives: an (IND-CCA secure) KEM, a signature scheme, and a pseudorandom function (PRF) (see Section 3). Similar to the X3DH protocol, our Signal-conforming AKE protocol offers a strong flavor of key-compromise security. Borrowing terminologies from AKE-related literature, our protocol is proven secure in the strong Canetti-Krawczyk (CK) type security models [19, 47, 35, 51], where the exchanged session key remains secure even if all the non-trivial combinations of the long-term secrets and session-specific secrets of the parties are compromised. In fact, our protocol is more secure than the X3DH protocol since it is even secure against KCI-attacks where the parties' session-specific secrets are compromised (see Footnote 2).[4] We believe the level of security offered by our Signal-conforming AKE protocol aligns with the level of security guaranteed by the DR protocol where (a specific notion of) security still holds even when such secrets are compromised.

We then provide details on how to recast our Signal-conforming AKE protocol into a key agreement protocol similar to what is used in the Signal protocol. We call our replacement of the X3DH protocol as the *Signal handshake protocol* (see Section 4). Unlike standard AKE protocols, the Signal handshake protocol, similarly to the X3DH protocol, makes several different design choices for efficiency reasons. The most prominent difference is that informally, the Signal handshake protocol reuses the same first message of the AKE protocol for a certain period of time. While this reduces communication and computation complexity and the storage size required by the server, this negatively affects the level of forward secrecy of the underlying Signal-conforming AKE protocol. We discuss in detail the trade-off between security and efficiency incurred when transforming our Signal-conforming AKE protocol into the Signal handshake protocol in Section 4.2.

In addition, we implement our post-quantum Signal handshake protocol in C, building on the open source libraries PQClean and LibTomCrypt (see Section 5). Our implementation [49] is fully generic and can thus be instantiated with a wide range of KEMs and signature schemes.We instantiate it with the currently selected NIST PQC standards (Kyber, Dilithium, Falcon and SPHINCS$^+$), and compare the bandwidth and computation costs that result from these choices. Our experiment shows that using Falcon or, to a lesser extent, Dilithium, is an order of magnitude more bandwidth-efficient than using SPHINCS$^+$.

Finally, while our basic protocol already satisfies a weak notion of deniability, we show how to progressively strengthen it by using a ring signature instead of a signature scheme and/or adding a non-interactive zero-knowledge proof system (NIZK) (see Section 6). The first protocol only relies on ring signature and the overhead to the basic protocol is small. This protocol is deniable as long as the communicating party is semi-honest (i.e., follows the protocol description but tries to infer information from what it observes). The second protocol adds an NIZK on top of this protocol to make it deniable even against communicating parties that are acting maliciously. While our construction seemingly offers the strongest form of (off-line) deniability, the formal proof relies on a strong knowledge-type assumption. The same holds for the X3DH protocol [66], and in fact, relying on such assumptions seems unavoidable [27, 70, 66]. We briefly discuss the efficiency of our deniable Signal-conforming AKE protocol using ring signatures in Remark 6.2.

[4]Although the X3DH protocol can naturally be made secure against leakage of session-specific secrets (including randomness generated within the session) by using the generic NAXOS trick, e.g., [51, 35, 48, 69], it typically requires additional computation. Since this negatively affects efficiency, we target AKE protocols without using the NAXOS trick.

1.2 Technical Overview

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

Recap on the X3DH Protocol. At a high level, the X3DH protocol allows for an asynchronous key exchange where two parties, say Alice and Bob, exchange a session key without having to be online at the same time. Even more, the party, say Bob, that wishes to send a secure message to Alice can do so without Alice even knowing Bob. For instance, imagine the scenario where you send a friend request and a message at the same time before being accepted as a friend. At first glance, it seems what we require is a non-interactive key exchange (NIKE) since Bob needs to exchange a key with Alice who is offline, while Alice does not yet know that Bob is trying to communicate with her. Unfortunately, solutions based on NIKEs are undesirable since they either provide weaker guarantees than standard (interactive) AKE or exhibit inefficient constructions [10, 21, 34, 62].

The X3DH protocol circumvents this issue by considering an *untrusted server* (e.g., the Signal server) to sit in the middle between Alice and Bob to serve as a public bulletin board. That is, the parties can store and retrieve information from the server while the server is not assumed to act honestly. A simplified description of the X3DH protocol, which still satisfies our purpose, based on the classical Diffie-Hellman (DH) key exchange is provided in Figure 1.[5] As the first step, Alice sends her DH component $g^x \in \mathbb{G}$ and its signature σ_A[6] to the server and then possibly goes offline. We point out that Alice does *not* need to know who she will be communicating with at this point. Bob, who may ad-hocly decide to communicate with Alice, then fetches Alice's first message from the server and uploads its DH component g^y to the server. As in a typical DH key exchange, Bob computes the session key $\mathsf{k_B}$ 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 DR protocol without waiting for Alice to come online.[7] Whenever Alice comes online, she can fetch whatever message Bob sent from the server.

Alice: $(\mathsf{lpk_A} = (g^a, \mathsf{vk_A}), \mathsf{lsk_A} = (a, \mathsf{sk_A}))$		Server		Bob: $(\mathsf{lpk_B} = (g^b, \mathsf{vk_B}), \mathsf{lsk_B} = (b, \mathsf{sk_B}))$
$x \leftarrow\$ \mathbb{Z}_p$ $\sigma_\mathsf{A} \leftarrow \mathsf{Sign}(\mathsf{sk_A}, g^x)$ Store x Upload g^x, σ_A to server `-- go offline --`	$\xrightarrow{g^x, \sigma_\mathsf{A}}$	Store $(\text{Alice}, g^x, \sigma_\mathsf{A})$	$\xrightarrow{g^x, \sigma_\mathsf{A}}$	Fetch $(\text{Alice}, g^x, \sigma_\mathsf{A})$ $\mathsf{Verify}(\mathsf{vk_A}, g^x, \sigma_\mathsf{A}) \overset{?}{=} 1$ $y \leftarrow\$ \mathbb{Z}_p$ $\mathsf{k_B} := \mathsf{KDF}((g^x)^b, (g^a)^y, (g^x)^y)$ Upload g^y to server Erase y
`-- come online --` Fetch ((Alice, Bob), g^y) $\mathsf{k_A} := \mathsf{KDF}((g^b)^x, (g^y)^a, (g^y)^x)$	$\xleftarrow{g^y}$	Store $((\text{Alice}, \text{Bob}), g^y)$	$\xleftarrow{g^y}$	

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

Casting the X3DH Protocol as an AKE Protocol. It is not difficult to see that the X3DH protocol

[5]We assume Alice and Bob know each other's long-term key. In practice, this can be enforced by "out-of-bound" authentications (see [56, Section 4.1]).

[6]In the actual protocol [56, 60], XEdDSA is used as the signature scheme, and the same long-term key (a, g^a) is used for both key exchange and signing.

[7]In practice, Bob may initiate the DR protocol using $\mathsf{k_B}$ and send his message to Alice along with g^y to the server before Alice responds.

can be cast as a specific type of AKE protocol. In particular, we can think of the server as an adversary that tries to mount a person-in-the-middle attack in a standard AKE protocol. Viewing the server as a malicious adversary, rather than some semi-honest entity, has two benefits: the parties do not need to put trust in the server since the protocol is supposed to be secure even against a malicious server, while the server or the company providing the app is relieved from having to "prove" that it is behaving honestly. One distinguishing feature required by the X3DH protocol when viewed as an AKE protocol is that it needs to be a two-round protocol where the initiator message is generated *independently* from the responder. That is, Alice needs to be able to store her first message to the server without knowing who she will be communicating with. In this work, we define an AKE protocol with this property as a *Signal-conforming* AKE protocol.

Regarding the security model for a Signal-conforming AKE protocol, we base it on the vast prior works on AKE protocols. Specifically, we build on the recent formalizations of [37, 24] 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 DR 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 [19, 47, 35, 51], which is a strengthening of the Bellare-Rogaway security model [7] considered by [37, 24]. A detailed discussion and comparison between our model and the numerous other security models of AKE protocols are provided in Section 2.

Lack of Signal-Conforming AKE Protocol. The main feature of a Signal-conforming AKE protocol is that the initiator's message is *independent* of the responder. Although this seems like a very natural feature considering DH-type AKE protocols, it turns out that they are quite unique (see Brendel et al. [17] 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. [25] and Kawashima et al. [46]. Considering the community is still in the process of assessing the concrete parameter selection for *standard* CSIDH [15, 59], it would be desirable to base the X3DH protocol on more well-established and versatile assumptions. On the other hand, when we turn our eyes to known generic constructions of AKE protocols [35, 36, 48, 69, 68, 43, 67] that can be instantiated from versatile assumptions, including post-quantum ones, we observe that they are either not Signal-conforming or require the NAXOS trick [51] (see Section 1.4) to be made secure against leakage of session-specific secrets.

Our Construction. To this end, in this work, we provide a new practical generic construction of a Signal-conforming AKE protocol from an (IND-CCA secure) KEM and a signature scheme. We believe this may be of independent interest in other protocols that require a flavor of "receiver obliviousness".[8]

The construction is simple: let us assume Alice and Bob's long-term keys consist of KEM key pairs $(\mathsf{ek}_\mathsf{A}, \mathsf{dk}_\mathsf{A})$ and $(\mathsf{ek}_\mathsf{B}, \mathsf{dk}_\mathsf{B})$ and signature key pairs $(\mathsf{vk}_\mathsf{A}, \mathsf{sk}_\mathsf{A})$ and $(\mathsf{vk}_\mathsf{B}, \mathsf{sk}_\mathsf{B})$, respectively. The Signal-conforming AKE protocol then starts by Alice (i.e., the initiator) generating a session-specific KEM key $(\mathsf{ek}_T, \mathsf{dk}_T)$, creating a signature $\sigma_\mathsf{A} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_\mathsf{A}, \mathsf{ek}_T)$, and sending $(\mathsf{ek}_T, \sigma_\mathsf{A})$ to Bob (i.e., the responder). Here, observe that Alice's message does not depend on who she will be communicating with.

Upon reception, Bob verifies the signature and then constructs two ciphertexts: one using Alice's long-term key $(\mathsf{K}_\mathsf{A}, \mathsf{C}_\mathsf{A}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_\mathsf{A})$ and another using the session-specific key $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_T)$. He then signs these ciphertext $\mathsf{M} := (\mathsf{C}_\mathsf{A}, \mathsf{C}_T)$ as $\sigma_\mathsf{B} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_\mathsf{B}, \mathsf{M})$, where we include other session-specific components in M in the actual construction. Since sending σ_B in the clear may serve as public evidence that Bob communicated with Alice, Bob will hide this. To this end, he derives two keys, a session key k_AKE and a one-time pad key k_OTP, by running a key derivation function on input the random KEM keys $(\mathsf{K}_\mathsf{A}, \mathsf{K}_T)$. Bob then sends $(\mathsf{C}_\mathsf{A}, \mathsf{C}_T, \mathsf{c} := \sigma_\mathsf{B} \oplus \mathsf{k}_\mathsf{OTP})$ to Alice and sets the session key as k_AKE. Here, note that we do not require Alice to hide her signature σ_A since this can only reveal that she was using the Signal app, unlike σ_B that may reveal who Bob was talking to.

Once Alice receives the message from Bob, she decrypts the ciphertexts $(\mathsf{C}_\mathsf{A}, \mathsf{C}_T)$, derives the two keys $(\mathsf{k}_\mathsf{AKE}, \mathsf{k}_\mathsf{OPT})$, and checks if $\sigma := \mathsf{c} \oplus \mathsf{k}_\mathsf{OTP}$ is a valid signature of Bob's. If so, she sets the session key as k_AKE.

[8]This property has also been called as *post-specified peers* [20] in the context of Internet Key Exchange (IKE) protocols.

We provide a formal proof and show that our protocol satisfies a strong flavor of security where the shared session key remains pseudorandom even to an adversary that can obtain any non-trivial combinations of the long-term private keys (i.e., $\mathsf{dk}_\mathsf{A}, \mathsf{dk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, \mathsf{sk}_\mathsf{B}$) and session-specific secret keys (i.e., dk_T). Notably, our protocol satisfies a stronger notion of security compared to the X3DH protocol since it prevents an adversary to impersonate Alice even if her session-specific secret key is compromised [56, Section 4.6].

Our first 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.[9] This is in contrast to the X3DH protocol that does not allow such a deniability attack.

In a second protocol, we show that we can protect Bob from such attacks by replacing the signature scheme with a *ring* signature scheme. In particular, Alice now further sends a session-specific ring signature verification key vk_T, and Bob signs to the ring $\{\mathsf{vk}_T, \mathsf{vk}_\mathsf{B}\}$. Effectively, when Alice outputs a signature from Bob $\sigma_{\mathsf{B},T}$, she cannot fully convince a third-party whether it originates from Bob since she could have signed $\sigma_{\mathsf{B},T}$ using her signing key sk_T corresponding to vk_T. Since we only require a ring of two users, we can use existing efficient post-quantum ring signatures to instantiate this idea. For example, targeting NIST security level 1, we have 2.5 KiB for Raptor [53] (based on NTRU), 4.4 KiB for DualRing [71] (based on M-LWE/SIS), and 3.5 KiB for Calamari [11] (based on CSIDH).

Although the intuition is clear, it turns out that turning this into a formal proof is quite difficult and we observe that for some practical ring signature schemes, this method only provides deniability against *semi-honest* adversaries, which are types of adversaries that follow the protocol description honestly. We provide a concrete attack where a malicious Alice that registers malformed key packages to the server can later (informally) prove to a third-party that Bob was trying to communicate with her even if Bob used a ring signature to sign. We thus propose a third protocol that additionally uses non-interactive zero-knowledge proof systems to make it secure even against malicious adversaries. Similar to all previous works on AKE protocols satisfying a strong flavor of key-compromise security [27, 70] (including the X3DH protocol [66]), the proof of deniability against malicious adversaries relies on a strong knowledge-type assumption.

1.3 Full version and additional resources

Due to page limitations, this work is a shortened version of previous work [39, 41]. The initial version, including the core protocols, was presented at PKC 2021 [39]. A more complete version, which includes a more detailed comparison with concurrent work, has been published at the Journal of Cryptology [41], and is available on IACR ePrint [40]. In addition, the following online resources may be of interest:

- Presentation slides by Keitaro Hashimoto:
  `https://kaminomisosiru.github.io/assets/pdf/HKKP-PKC2021.pdf`
- Presentation slides by Thomas Prest:
  `https://tprest.github.io/pdf/slides/x3dh-pqnet-2021.pdf`
- Presentation video by Keitaro Hashimoto:
  `https://www.youtube.com/watch?v=VO4fwOUHdMI`
- Proof of concept implementation by Kris Kwiatkowski:
  `https://github.com/post-quantum-cryptography/post-quantum-state-leakage-secure-ake`

[9]Note that Bob is *only* non-deniable of the fact that he tried to contact Alice. In particular, exchanged messages will still be deniable. This is because the Signal-conforming AKE protocol only concerns the exchanged key and the subsequent double ratchet protocol (which is deniable) concerns the exchanging of messages.

1.4 Subsequent Work

After the proceedings version of our paper [39] appeared, Brendel et al. [16] posted on ePrint a post-quantum key exchange protocol that can be used in place of the X3DH protocol based on designated verifier signatures. Since (their definition of) designated verifier signature can be shown to be identical to ring signatures, their result provides an alternative perspective on our second construction. They also introduce a new definition for deniability that is incomparable to ours and provide evidence that our second construction provides useful notion of deniability depending on the scenario. Very recently, Dobson and Galbraith [29] proposed an X3DH-style protocol tailored to SIDH (SI-X3DH).

2 Security Model for Signal-Conforming AKE Protocols

In this section, we define a security model for a *Signal-conforming* authenticated key exchange (AKE) protocol: AKE protocols that can be used as a drop-in replacement of the X3DH protocol. We first provide in Sections 2.1 to 2.3 a game-based security model building on the recent formalization of [37, 24] targeting general AKE protocols. We then discuss in Section 2.4 the modifications needed to make it Signal-conforming.

2.1 Execution Environment

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

- rand_i^s is the randomness hard-wired to π_i^s;
- sid_i^s ("session identifier") stores the identity of the session as specified by the protocol;
- Pid_i^s ("peer id") stores the identity of the intended communication partner;
- $\Psi_i^s \in \{\perp, \mathtt{accept}, \mathtt{reject}\}$ 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;
- $\mathsf{role}_i^s \in \{\perp, \mathtt{init}, \mathtt{resp}\}$ indicates π_i^s's role during the protocol execution.

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

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 [19, 26].

Definition 2.1 (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 say that an AKE is $(1 - \delta)$-correct if a faithful execution succeeds with probability at least $(1 - \delta)$. We defer the formal definition to Definition A.11.

2.2 Security Game

We define the security of an AKE protocol Π_{AKE} via a game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. We consider two slightly different variants, each denoted as $G^{\mathsf{weakFS}}_{\Pi_{\mathsf{AKE}}}(\mu, \ell)$ and $G^{\mathsf{FS}}_{\Pi_{\mathsf{AKE}}}(\mu, \ell)$. They capture a *weakly* and *perfect* forward secure AKE protocol, respectively. Roughly, when the long-term secret key is exposed, the former only ensures the security of past sessions where the adversary did not modify the exchanged messages. In contrast, the latter ensures the security of all past sessions regardless of the adversary actively modifying the exchanged messages. Further details on the difference are provided in Section 2.3.

More formally, the security game is parameterized by two integers μ (the number of honest parties) and ℓ (the maximum number of protocol executions per party), and proceeds as follows, where the freshness clauses Item 5a and Item 5b is used to define $G^{\mathsf{FS}}_{\Pi_{\mathsf{AKE}}}(\mu, \ell)$ and $G^{\mathsf{weakFS}}_{\Pi_{\mathsf{AKE}}}(\mu, \ell)$, respectively:

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

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

- $\mathsf{Send}(i, s, m)$: This query allows $\mathcal{A}$ to send an arbitrary message m to oracle π_i^s. The oracle will respond according to the protocol specification and its current internal state. To start a new oracle, the message m takes a special form:
  $\langle \mathtt{START} : \mathsf{role}, j\rangle$; $\mathcal{C}$ initializes π_i^s in the role role, having party P_j as its peer, that is, $\mathcal{C}$ sets $\mathsf{Pid}_i^s := j$ and $\mathsf{role}_i^s := \mathsf{role}$. If π_i^s is an initiator (i.e., $\mathsf{role} = \mathtt{init}$), then $\mathcal{C}$ returns the first message of the protocol.[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 random key depending on the challenge bit b.

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

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

[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 Section 2.4 for more details.

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

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. At this point, the tested oracle must be *fresh*. Here, an oracle π_i^s with $\mathsf{Pid}_i^s = j$[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
   (a) in game $G^{\mathsf{FS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$, P_j is corrupted only after π_i^s finishes the protocol execution.
   (b) in game $G^{\mathsf{weakFS}}_{\Pi_{\mathsf{AKE}}}(\mu,\ell)$, P_j is not corrupted.

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

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

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

Definition 2.2 (Security of AKE Protocol). *An AKE protocol Π_{AKE} is secure with perfect (resp. weak) forward secrecy if $\mathsf{Adv}^{\mathsf{AKE\text{-}FS}}_{\Pi_{\mathsf{AKE}}}(\mathcal{A})$ (resp. $\mathsf{Adv}^{\mathsf{AKE\text{-}weakFS}}_{\Pi_{\mathsf{AKE}}}(\mathcal{A})$) is negligible for any QPT adversary $\mathcal{A}$.*

2.3 Security Properties

In this section, we explain the security properties captured by our security model.

The freshness clauses Items 1 and 2 imply that we only exclude the reveal of session keys for the tested oracle π^T and its partner oracles. These capture *key independence*: if the revealed session keys differ from the key of π^T, then they must not enable computing the session key of π^T. Note that key independence implies resilience to "no-match attacks" presented by Li and Schäge [52]. Moreover, the two items capture *implicit authentication* between the involved parties. This is because an oracle π that computes the same session key as π^T but disagrees on the peer would not be a partner of π^T, and hence, an adversary can obtain the key of π^T by querying the session key computed by π. Specifically, our model captures resistance to *unknown key-share* (UKS) attacks [14], a specific class of attack that makes two parties compute the same session key but have different views on whom they are communicating with.

The freshness clauses Items 3 to 5 indicate that the game allows the adversary to reveal any subset of the four secret items of information — the long-term secret keys and the session-states of the two parties (where one party is the party defined by the tested oracle and the other its peer) — except for the combination where both the long-term secret key and session-state of one of the party is revealed. In particular, Item 5a captures *perfect forward secrecy* [28, 19]: the adversary can obtain the long-term secret keys of both parties once the tested oracle finishes the protocol and generates a session key. On the other hand, Item 5b captures *weak forward secrecy* [47]: the adversary can obtain the long-term secret keys of both parties only when it has been passive during the protocol run of the tested oracle.

Our model also captures resistance to *key-compromise impersonation* (KCI) attacks [13]. 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 with no restriction. Most importantly, our model captures resistance to *state leakage* [19, 47, 51, 35] where an adversary is allowed to obtain session-states of both parties. Note that our security model is strictly stronger than recent models [37, 24] that prevent the adversary from learning sessions-states.

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

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

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

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

2.4 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 [56] used by the Signal protocol [2]. As explained in Section 1.2, the X3DH protocol can be viewed as a special type of AKE protocol where the Signal server acts as an (untrusted) bulletin board, where parties can store and retrieve information from. More specifically, the Signal server can be viewed as an adversary for an AKE protocol that controls the communication channel between the parties. When casting the X3DH protocol as an AKE protocol, one crucial property is that the first message of the initiator is generated *independently* of the communication partner. This is because, in secure messaging, parties are often *offline* during the key agreement so if the first message depended on the communication partner, then we must wait until they come online to complete the key agreement. Since we cannot send messages without agreeing on a session key, such an AKE protocol where the first message depends on the communication partner cannot be used as an alternative to the X3DH protocol.

We abstract this crucial yet implicit property achieved by the X3DH protocol as *receiver obliviousness*. As noted in Footnote 8, this property has also been called as *post-specified peers* [20] in the context of Internet Key Exchange (IKE) protocols.

Definition 2.3 (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 [25, 46]) can be checked to be receiver oblivious.

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

In this section, we propose a Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ that can be used to construct a Signal's initial key agreement (Signal handshake) protocol such as the X3DH protocol. Unlike the X3DH protocol, our protocol can be instantiated from post-quantum assumptions, and moreover, it also provides stronger security against state leakage. The protocol description is presented in Figure 2. Details follow.

Building Blocks. Our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ use the following building blocks. Formal definitions for the syntax and properties of the building blocks can be found in Appendix A.

- $\Pi_{\mathsf{KEM}} = (\mathsf{KEM.Setup}, \mathsf{KEM.KeyGen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ is a KEM scheme that is $\mathsf{IND\text{-}CCA}$ secure and assume we have $(1-\delta_{\mathsf{KEM}})$-correctness, ν_{KEM}-high encapsulation key min-entropy and χ_{KEM}-high ciphertext min-entropy.
- $\Pi_{\mathsf{wKEM}} = (\mathsf{wKEM.Setup}, \mathsf{wKEM.KeyGen}, \mathsf{wKEM.Encap}, \mathsf{wKEM.Decap})$ is a KEM schemes that is $\mathsf{IND\text{-}CPA}$ secure (and not $\mathsf{IND\text{-}CCA}$ secure) and assume we have $(1-\delta_{\mathsf{wKEM}})$-correctness, ν_{wKEM}-high encapsulation key min-entropy, and χ_{wKEM}-high ciphertext min-entropy. In the following, for simplicity of presentation and without loss of generality, we assume $\delta_{\mathsf{wKEM}} = \delta_{\mathsf{KEM}}$, $\nu_{\mathsf{wKEM}} = \nu_{\mathsf{KEM}}$, $\chi_{\mathsf{wKEM}} = \chi_{\mathsf{KEM}}$.
- $\Pi_{\mathsf{SIG}} = (\mathsf{SIG.Setup}, \mathsf{SIG.KeyGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ is a signature scheme that is $\mathsf{EUF\text{-}CMA}$ secure and $(1-\delta_{\mathsf{SIG}})$-correctness. We denote d as the bit length of the signature generated by $\mathsf{SIG.Sign}$.
- $\mathsf{F} : \mathcal{FK} \times \{0,1\}^* \to \{0,1\}^{\kappa+d}$ is a pseudo-random function family with key space $\mathcal{FK}$.
- $\mathsf{Ext} : \mathcal{S} \times \mathcal{KS} \to \mathcal{FK}$ is a strong $(\gamma_{\mathsf{KEM}}, \varepsilon_{\mathsf{Ext}})$-extractor.

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

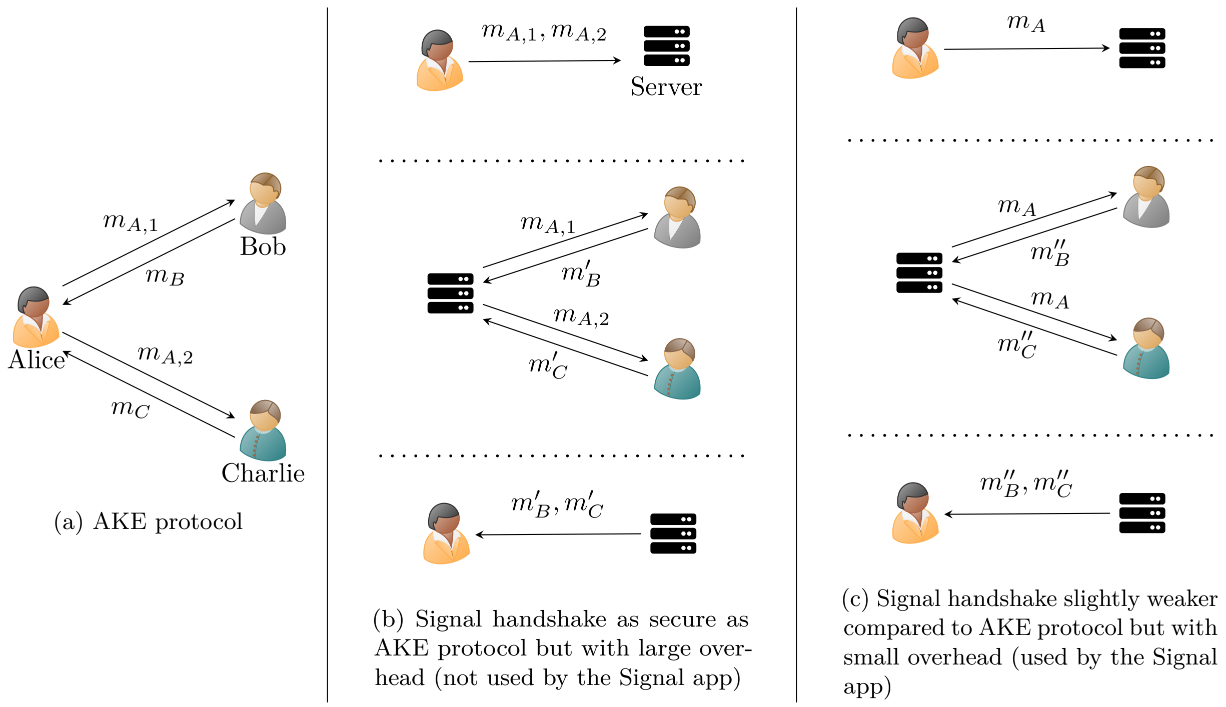
Long-Term Public and Secret Keys. Each party P_i runs $(\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{KEM.KeyGen}(\mathsf{pp}_{\mathsf{KEM}})$ and $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{SIG.KeyGen}(\mathsf{pp}_{\mathsf{SIG}})$. Party P_i's long-term public key and secret key are set as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

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

1. Party P_i sets $\mathsf{Pid}_i^s := j$ and $\mathsf{role}_i^s := \mathtt{init}$. P_i computes $(\mathsf{dk}_T, \mathsf{ek}_T) \leftarrow \mathsf{wKEM.KeyGen}(\mathsf{pp}_{\mathsf{wKEM}})$ and $\sigma_i \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_i, \mathsf{ek}_T)$. Then it sends $(\mathsf{ek}_T, \sigma_i)$ to party P_j. P_i stores the ephemeral decapsulation key dk_T as the session-state, i.e., $\mathsf{state}_i^s := \mathsf{dk}_T$.[12]
2. Party P_j sets $\mathsf{Pid}_j^t := i$ and $\mathsf{role}_j^t := \mathtt{resp}$. Upon receiving $(\mathsf{ek}_T, \sigma_i)$, P_j first checks whether $\mathsf{SIG.Verify}(\mathsf{vk}_i, \mathsf{ek}_T, \sigma_i) = 1$ holds. If not, P_j sets $(\Psi_j, \mathsf{k}_j^t, \mathsf{state}_j) := (\mathtt{reject}, \bot, \bot)$ and stops. Otherwise, it computes $(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek}_i)$ and $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{wKEM.Encap}(\mathsf{ek}_T)$. Then P_j derives two PRF keys $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K})$ and $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$. It then defines the session-identifier as $\mathsf{sid}_j^t := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{C} \| \mathsf{C}_T$ and computes $\mathsf{k}_j \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_j) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_j)$, where $\mathsf{k}_j \in \{0,1\}^\kappa$ and $\tilde{k} \in \{0,1\}^d$, and sets the session key as $\mathsf{k}_j^t := \mathsf{k}_j$. P_j then signs $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_j, \mathsf{sid}_j^t)$ and encrypts it as $\mathsf{c} \leftarrow \sigma \oplus \tilde{k}$. Finally, it sends $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ to P_i and sets $\Psi_j := \mathtt{accept}$. Here, note that P_j does not require to store any session-state, i.e., $\mathsf{state}_j^t = \bot$.
3. Upon receiving $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$, P_i first decrypts $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{dk}_i, \mathsf{C})$ and $\mathsf{K}_T \leftarrow \mathsf{wKEM.Decap}(\mathsf{dk}_T, \mathsf{C}_T)$, and derives two PRF keys $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K})$ and $\mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$. It then sets the session-identifier as $\mathsf{sid}_i^s := P_i \| P_j \| \mathsf{lpk}_i \| \mathsf{lpk}_j \| \mathsf{ek}_T \| \mathsf{C} \| \mathsf{C}_T$ and computes $\mathsf{k}_i \| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}_i) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid}_i)$, where $\mathsf{k}_i \in \{0,1\}^\kappa$ and $\tilde{k} \in \{0,1\}^d$. P_i then decrypts $\sigma \leftarrow \mathsf{c} \oplus \tilde{k}$ and checks whether $\mathsf{SIG.Verify}(\mathsf{vk}_j, \mathsf{sid}_i^s, \sigma) = 1$ holds. If not, P_i sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{reject}, \bot, \bot)$ and stops. Otherwise, it sets $(\Psi_i, \mathsf{k}_i^s, \mathsf{state}_i) := (\mathtt{accept}, \mathsf{k}_i, \bot)$. Here, note that P_i deletes the session-state $\mathsf{state}_i^s = \mathsf{dk}_T$ at the end of the key exchange.

Remark 3.1 (A Note on Session-State). The session-state of the initiator P_i contains the ephemeral decryption key dk_T, and P_i must store it until the peer responds. Any other information that is computed after receiving the message from the peer is erased after the session key is established. In contrast, the responder P_j has no session-state because the responder directly computes the session key after receiving the initiator's message and does not need to store any session-specific information. That is, all states can be erased as soon as the session key is computed.

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



(a) AKE protocol

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

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

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

Security. The following theorems establish the correctness and security of our protocol $\Pi_{\mathsf{SC\text{-}AKE}}$. Proofs can be found in the full version of this work, see Section 1.3.

Theorem 3.2 (Correctness of $\Pi_{\mathsf{SC\text{-}AKE}}$). *Assume Π_{KEM} and Π_{wKEM} are $(1-\delta_{\mathsf{KEM}})$-correct and Π_{SIG} is $(1-\delta_{\mathsf{SIG}})$-correct. Then, $\Pi_{\mathsf{SC\text{-}AKE}}$ is $(1-\mu\ell(\delta_{\mathsf{SIG}}+2\delta_{\mathsf{KEM}})/2)$-correct.*

Theorem 3.3 (Security of $\Pi_{\mathsf{SC\text{-}AKE}}$). *For any QPT adversary $\mathcal{A}$ that plays the game $G^{\mathsf{FS}}_{\Pi_{\mathsf{SC\text{-}AKE}}}(\mu,\ell)$ with μ parties that establishes at most ℓ sessions per party, there exist QPT algorithms $\mathcal{B}_1$ breaking the $\mathsf{IND\text{-}CPA}$ security of Π_{wKEM}, $\mathcal{B}_2$ breaking the $\mathsf{IND\text{-}CCA}$ security of Π_{KEM}, $\mathcal{B}_3$ breaking the $\mathsf{EUF\text{-}CMA}$ security of Π_{SIG}, and $\mathcal{D}_1$ and $\mathcal{D}_2$ breaking the security of PRF F such that*

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

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

4 Post-Quantum Signal Handshake

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

4.1 Signal Handshake From Signal-conforming AKE protocol

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

In contrast, in secure messaging, parties are not always online, and the communicating partner may be unknown at the time of registration. Therefore, we need a mechanism for initiating communication even when the partners are offline and undefined. Signal handshake realizes this by using a possibly untrusted server (see Figure 3b). In its most secure version (which is not used by the Signal app), if Alice anticipates communicating with at most two parties, she first uploads two first messages $m_{A,1}$ and $m_{A,2}$ of the AKE protocol to the server and goes offline. She will replenish the first messages on the server periodically so that the server does not use up all the first messages. Since Alice does not know who will use the first messages in this case, only Signal-conforming AKE protocols (i.e., two-round and the first message can be generated independently of the responder) can be used for the Signal handshake.

Next, if Bob wants to share a session key with Alice, he accesses the server and receives the unused $m_{A,1}$. Then, Bob computes the session key and uploads the second message m'_B. If Charlie also exchanges a session key with Alice, he executes the protocol in the same way as Bob, using a different unused first message $m_{A,2}$.

Finally, when Alice comes online, she downloads the second messages m'_B and m'_C from the server and derives the session keys between Bob and Charlie. In this way, parties can exchange session keys asynchronously. Moreover, the Signal handshake in Figure 3b can be shown to be as secure as the underlying AKE protocol. This is because we can view the server as a person-in-the-middle adversary in AKE protocols, and AKE protocols are defined to be secure against such adversaries.

Although secure, this approach requires Alice to upload as many first messages as the number of parties she anticipates communicating with. To reduce storage overhead, the Signal protocol reuses the first message for multiple key exchange sessions (see Figure 3c). Alice uploads one first message m_A to the server, and periodically updates it, e.g., once a week or once a month. Bob and Charlie exchange session keys with Alice using the same first message m_A. Reusing the first message provides a trade-off between security guarantees and communication cost. For example, if an adversary obtains the randomness used to generate m_A, depending on the underlying AKE protocol, it may expose both session keys exchanged between Bob and Charlie. In contrast, without re-use, the security of the AKE protocol guarantees that an adversary can recover only the session key that used m_A. A more detailed discussion can be found at the end of Section 4.2.

4.2 Details of Our Post-Quantum Signal Handshake

We provide the details of our post-quantum Signal handshake based on our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$. Unlike the X3DH protocol, our protocol can be made post-quantum by choosing appropriate post-quantum building blocks. The protocol description is presented in Figure 4. As explained in Section 4.1, parties communicate with each other with the help of the server and reuse the first messages of the AKE protocol for a certain interval (see Figure 3c).

Registration phase. Alice and Bob generate their long-term keys lpk_A and lpk_B, respectively, and register them to the server (see the top of Figure 4). Note that parties upload their long-term keys only once. As in the Signal *app*, we assume the validity of the long-term keys are checked between the parties by some "out-of-bound" authentication mechanism (see [56, Section 4.1]).

Initiator. Next, Alice uploads a first message of the AKE protocol: an encapsulation key ek_T (called *signed pre-key* in the Signal white paper [56]) along with a signature σ_A (called *pre-key signature* in the Signal white paper [56]) to the server (see the center of Figure 4)[13]. The signed pre-key is reused for multiple key exchange sessions and is updated at some interval (e.g. once a week or once a month).

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

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

Alice (Initiator)		**Server**		**Bob** (Responder)
$\mathsf{lsk_A} = (\mathsf{dk_A}, \mathsf{sk_A})$, $\mathsf{lpk_A} = (\mathsf{ek_A}, \mathsf{vk_A})$ Upload $\mathsf{lpk_A}$ to server	$\xrightarrow[\textit{long-term key}]{\mathsf{lpk_A}}$		$\xleftarrow[\textit{long-term key}]{\mathsf{lpk_B}}$	$\mathsf{lsk_B} = (\mathsf{dk_B}, \mathsf{sk_B})$, $\mathsf{lpk_B} = (\mathsf{ek_B}, \mathsf{vk_B})$ Upload $\mathsf{lpk_B}$ to server
$(\mathsf{ek}_T, \mathsf{dk}_T) \leftarrow$ $\mathsf{wKEM.KeyGen}(\mathsf{pp_{wKEM}})$ $\sigma_\mathsf{A} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk_A}, \mathsf{ek}_T)$ Store $\mathsf{ek}_T, \mathsf{dk}_T$ Upload $\mathsf{ek}_T, \sigma_\mathsf{A}$ to server	$\xrightarrow{\mathsf{ek}_T, \sigma_\mathsf{A}}$ *signed pre-key* and *pre-key signature*	Store (Alice, $(\mathsf{lpk_A}, \mathsf{ek}_T, \sigma_\mathsf{A})$)		
Fetch ((Alice, Bob), $(\mathsf{lpk_B}, \mathsf{C}, \mathsf{C}_T, \mathsf{c})$) $\mathsf{K} \leftarrow \mathsf{KEM.Decap}(\mathsf{dk_A}, \mathsf{C})$ $\mathsf{K}_T \leftarrow \mathsf{wKEM.Decap}(\mathsf{dk}_T, \mathsf{C}_T)$ $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K}); \mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$ $\mathsf{k_A} \\| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid})$ $\sigma \leftarrow \mathsf{c} \oplus \tilde{k}$ $\mathsf{SIG.Verify}(\mathsf{vk_B}, \mathsf{sid}, \sigma) \overset{?}{=} 1$ Output the session key $\mathsf{k_A}$	$\xleftarrow{\mathsf{lpk_B}, \mathsf{C}, \mathsf{C}_T, \mathsf{c}}$	Store ((Alice, Bob), $(\mathsf{lpk_B}, \mathsf{C}, \mathsf{C}_T, \mathsf{c})$)	$\xrightarrow{\mathsf{lpk_A}, \mathsf{ek}_T, \sigma_\mathsf{A}}$ $\xleftarrow{\mathsf{C}, \mathsf{C}_T, \mathsf{c}}$	Fetch (Alice, $(\mathsf{lpk_A}, \mathsf{ek}_T, \sigma_\mathsf{A})$) $\mathsf{SIG.Verify}(\mathsf{vk_A}, \mathsf{ek}_T, \sigma_\mathsf{A}) \overset{?}{=} 1$ $(\mathsf{K}, \mathsf{C}) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek_A})$ $(\mathsf{K}_T, \mathsf{C}_T) \leftarrow \mathsf{wKEM.Encap}(\mathsf{ek}_T)$ $\mathsf{K}_1 \leftarrow \mathsf{Ext}_s(\mathsf{K}); \mathsf{K}_2 \leftarrow \mathsf{Ext}_s(\mathsf{K}_T)$ $\mathsf{k_B} \\| \tilde{k} \leftarrow \mathsf{F}_{\mathsf{K}_1}(\mathsf{sid}) \oplus \mathsf{F}_{\mathsf{K}_2}(\mathsf{sid})$ $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk_B}, \mathsf{sid})$ $\mathsf{c} \leftarrow \sigma \oplus \tilde{k}$ Upload $\mathsf{C}, \mathsf{C}_T, \mathsf{c}$ to server Output the session key $\mathsf{k_B}$

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

Responder. Finally, when Bob wants to communicate with Alice, he accesses the server and downloads Alice's long-term key and signed pre-key $(\mathsf{lpk}_\mathsf{A}, \mathsf{ek}_T, \sigma_\mathsf{A})$. Then, he runs the AKE protocol of the responder's part and uploads the response message $(\mathsf{C}, \mathsf{C}_T, \mathsf{c})$ to the server. Finally, when Alice comes online, she downloads the long-term key and the response message from Bob and derives the shared initial secret (see the bottom of Figure 4). It is clear that if the signed pre-keys are not reused, then the protocol is secure as the underlying Signal-conforming AKE protocol.

Let us discuss the security implication of reusing signed pre-keys. It is clear that it *seems* insecure to reuse the signed pre-keys compared to not reusing them. The question is, to what extent are they insecure? In our post-quantum Signal handshake, multiple session keys are exchanged using the same pair of long-term key and signed pre-key. We first argue that, if either the long-term key or the signed pre-key are not leaked, then the exchanged keys remain secure. Our security model (defined in Section 2) guarantees that the session key of oracle π is secure even if an adversary forwards the first message, i.e., a signed pre-key and pre-key signature, generated by oracle π, to multiple responder oracles and obtains at most one of the two KEM keys include in the long-term key or the signed pre-key used by π. This attack captures the scenario where a party sends the same first message to multiple responders, i.e., reuses its signed pre-key. Therefore, our post-quantum Signal handshake is as secure as a variant that never reuses a signed pre-key (i.e., our Signal-conforming AKE protocol) as long as one of the long-term key or the signed pre-key are not leaked. The distinction between our post-quantum Signal handshake and our Signal-conforming AKE protocol becomes clear when both the long-term key and the signed pre-key are leaked. Observe that once both keys are leaked, the adversary can compute *all* session keys that were exchanged using them. In other words, the number of session keys that are compromised is the same as the number of times the signed pre-key was reused. Thus, the Signal handshake is less secure than the underlying AKE protocol since the number of session keys that are exposed is larger when both the long-term key and the signed pre-key are leaked. To mitigate the number of exposed session keys, those parties looking for better security can use signed pre-keys only once or add a so-called *one-time pre-key* in the first message (i.e., an additional non-signed one-time KEM key). This is exactly what the Signal app does. In this case, even if all the KEM keys used in a specific session are leaked, an adversary can not compute the session keys of the other sessions since a different KEM key is used for each session. Thus, this mitigation can be used to enhance the security in a scenario where the long-term key and the signed pre-key are both exposed but the one-time pre-key is not.

5 Instantiating Post-Quantum Signal Handshake

In this section, we present the implementation details of our post-quantum Signal handshake protocol presented in Figure 4. We take existing implementations of post-quantum KEMs and signature schemes submitted for the NIST PQC standardization. To instantiate our Signal handshake, we pair variants of KEMs and signature schemes corresponding to the same security level. Due to the scope of the conference, we limited ourselves to schemes currently selected by NIST for standardization: Kyber, Dilithium, Falcon and SPHINCS$^+$. The full version of this paper [41] presents more instantiations with Round 3 candidates. We study the efficiency of our instantiations through two metrics — the total amount of data exchanged between parties and run-time performance. Our implementation [49] is available in the form of open-source software.

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/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 [33],

- 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, 5, 42],
- b: equals the security level of the underlying post-quantum KEM scheme, where $b \in \{128, 192, 256\}$,
- d: equals the byte length of the signature generated by the post-quantum signature scheme Π_{SIG},
- Π_{KEM}, Π_{wKEM}, Π_{SIG}: the implementation uses pairs of KEM and signature schemes. We always use the same KEM scheme for Π_{KEM} and Π_{wKEM}.

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

The concrete implementation of post-quantum schemes is provided by *PQ Crypto Catalog* library [50], which is a collection of implementations submitted to NIST PQC standardization process. We also use *LibTomCrypt* library [1] which provides an implementation of the building blocks HMAC, HKDF and SHA-256. We note that we use portable `C` code implementations of schemes, which do not include platform specific optimizations. There are two reasons for such a choice. First, our goal was to show the expected results on a broad number of platforms. Second, the *PQ Crypto Catalog* library does not provide hardware-assisted optimizations for all schemes, hence selectively enabling would result in unfair comparison.

5.2 Efficiency Analysis

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

To properly assess the cost, we modeled a scenario according to Figure 4. Note that in the Signal protocol, long-term public keys lpk are fetched from the server. Parties do not store the keys lpk corresponding to those that they have not communicated with before.[14] We provide two metrics:

- **Data transfer cost:** the amount of data exchanged when two parties establish a session key.
- **Computational cost:** CPU cycles spent in computation during session establishment by both parties.

Cost analysis for each metric is provided separately.

Data Transfer Cost. Table 1 provides the selected results for schemes selected by NIST for standardization.[15] The lpk column contains the byte size of a long-term key. The following four columns contain the byte size of the data exchanged by the initiator, the server and the responder during a session key establishment (as per Figure 4). Finally, the column **Total** contains the total size of data exchanged between Alice and Bob.

From Table 1, we can conclude that using Falcon or, to a lesser extent, Dilithium, is an order of magnitude more bandwidth-efficient than using SPHINCS$^+$.

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

[14]The X3DH protocol assumes the parties authenticate the long-term public keys through some authenticated channel [56, Section 4.1].

[15]Results for a more exhaustive combination of instantiations can be found in the repository containing the implementation [49].

Remark 5.1 (Note on Low Quality Network Links). We anticipate the Signal handshake to be used with handheld devices and areas with a poor quality network connection. In such cases, larger key, ciphertext and signature sizes generated may negatively impact the quality of the connection. Network packet loss is an additional factor that should be considered when choosing schemes for concrete instantiation.

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

Depending on the scheme used, an increased packet loss may negatively impact session establishment time (see [58]). For example, pairing Kyber512 with Falcon512 requires exchange of $npacks = 7$ packets over the network, whereas pairing Kyber512 with SPHINCS-SHAKE256-128f-simple requires 27 packets. Assuming increased packet rate loss of 2%, the probability of losing a packet in the former case is $1-(1-rate)^{npacks} = 13\%$, where in the latter it is 42%. In the latter case, at the median, every third session key establishment will experience packet retransmission and hence a delay.

Scheme	lpk	**I→S**	**S→R**	**R→S**	**S→I**	**Total**
NIST security level 1						
Falcon512/Kyber512	1697	1490	3187	2226	3923	10826
Dilithium2/Kyber512	2112	3220	5332	3956	6068	18576
SPHINCS-SHAKE256-128f-s/Kyber512	832	17888	18720	18624	19456	74688
NIST security level 3						
Dilithium3/Kyber768	3136	4477	7613	5469	8605	26164
SPHINCS-SHAKE256-192f-s/Kyber768	1232	36848	38080	37840	39072	151840
NIST security level 5						
Falcon1024/Kyber1024	3361	2898	6259	4466	7827	21450
Dilithium5/Kyber1024	4160	6163	10323	7731	11891	36108
SPHINCS-SHAKE256-256f-s/Kyber1024	1632	51424	53056	52992	54624	212096

Table 1: Data transfer cost in bytes of Figure 4 instantiated with the algorithms selected by NIST for standardization. We use the following abbreviations: I = Initiator, S = Server, R = Responder. Note that:
(a) (S→R) − (I→S) = (S→I) − (R→S) = lpk.
(b) Total := (I→S) + (S→R) + (R→S) + (S→I).

Computational Cost. The most expensive operations in our protocol are those performed by asymmetric primitives. Our post-quantum Signal handshake performs 9 such operations during a session agreement: the initiator runs a KEM key generation, two KEM decapsulations, one signing and one verification, and the responder performs two KEM encapsulations, one signing and one verification.

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

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

Table 2 contains the number of CPU cycles spent during session establishment for selected post-quantum schemes. The **3-way handshake** column shows the cost of the whole session establishment, the **Finish** column contains the final part of the handshake, performed by the initiator. Presented results exclude the cost of long-term key (lpk) generation. We note that the computational cost is a less absolute metric than the bandwidth cost as it depends on the concrete implementation of the post-quantum schemes.

Scheme	**3-way handshake**	**Finish**
NIST security level 1		
Falcon512/Kyber512	3901892	807969
Dilithium2/Kyber512	4996125	1060589
SPHINCS-SHAKE256-128f-s/Kyber512	445636240	12041223
NIST security level 3		
Dilithium3/Kyber768	8247125	1652879
SPHINCS-SHAKE256-192f-s/Kyber768	710632430	17476256
NIST security level 5		
Falcon1024/Kyber1024	8063636	1661382
Dilithium5/Kyber1024	10576515	1672158
SPHINCS-SHAKE256-256f-s/Kyber1024	1344750414	17978590

Table 2: Computational cost in CPU cycles of Figure 4 instantiated with schemes selected by NIST for standardization. Benchmarking run on the Intel Xeon E3-1220v3 @3.1GhZ with Turbo Boost disabled.

6 Two Deniable Variants of Our Basic Signal-Conforming AKE

In this section, we show how to modify our Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ in order to satisfy a progressively stronger notion of deniability. Due to page limitations, we only briefly present our two new protocols and their security properties (deniability against semi-honest and malicious adversaries, respectively). A more comprehensive description of the protocols and their properties can be found in the full version of the paper (see Section 1.3).

We recall the definition of deniability in Definition 6.1. Informally, it means a protocol participant cannot use the protocol transcript after the fact to incriminate other participants.

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*.*

Building Blocks. In addition to the building blocks of $\Pi_{\mathsf{SC\text{-}AKE}}$ (Section 3), our deniable Signal-conforming AKE protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ against *semi-honest* adversaries uses a ring signature scheme.

- $\Pi_{\mathsf{RS}} = (\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Verify})$ is a ring signature scheme that is anonymous and unforgeable and assume we have $(1 - \delta_{\mathsf{RS}})$-correctness.

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

Initiator P_i

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Output the session key k_i

Responder P_j

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

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

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

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

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

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

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

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

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

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

Output the session key k_j

Figure 5: Deniable Signal-conforming AKE protocols $\Pi_{\mathsf{SC\text{-}DAKE}}$ (includes all operations except for those in gray text inside dotted boxes) and $\Pi'_{\mathsf{SC\text{-}DAKE}}$ (includes all operations). The initiator no longer signs the first message, and the other components that differ from the non-deniable protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ are indicated by a box. $\Pi_{\mathsf{SC\text{-}DAKE}}$ (resp. $\Pi'_{\mathsf{SC\text{-}DAKE}}$) satisfies deniability against malicious (resp. semi-honest) adversaries.

In addition to the building blocks of $\Pi_{\mathsf{SC\text{-}DAKE}}$, our deniable Signal-conforming AKE protocol $\Pi'_{\mathsf{SC\text{-}DAKE}}$ against *malicious* adversaries requires a KEM with a plaintext-awareness property, and a NIZK argument system. Formal definitions for the syntax and properties of these building blocks can be found in Appendix B.

- $\Pi_{\mathsf{KEM}} = (\mathsf{KEM.Setup}, \mathsf{KEM.KeyGen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ is an $\mathsf{IND\text{-}CCA}$ secure KEM scheme as in the previous section that additionally satisfies PA_μ-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 $(\mathsf{vk}, \mathsf{sk}) = \mathsf{RS.KeyGen}(\mathsf{pp}; \mathsf{rand})$.

Figure 5 presents two Signal-conforming AKE protocols: $\Pi_{\mathsf{SC\text{-}DAKE}}$ and $\Pi'_{\mathsf{SC\text{-}DAKE}}$ achieve deniability against semi-honest and malicious adversaries, respectively. Compared to our first protocol $\Pi_{\mathsf{SC\text{-}AKE}}$, the main additions of $\Pi_{\mathsf{SC\text{-}DAKE}}$ and $\Pi'_{\mathsf{SC\text{-}DAKE}}$ are emphasized in Figure 5 in (solid and dotted) boxes.

Remark 6.2 (Efficiency of $\Pi_{\mathsf{SC\text{-}DAKE}}$). We evaluate the bandwidth overhead of $\Pi_{\mathsf{SC\text{-}DAKE}}$ compared to $\Pi_{\mathsf{SC\text{-}AKE}}$. The first message of $\Pi_{\mathsf{SC\text{-}DAKE}}$ contains the additional ring signature verification key vk_T. Thus, the size of the first message increases by the amount of vk_T compared to $\Pi_{\mathsf{SC\text{-}AKE}}$.[16] The second message of $\Pi_{\mathsf{SC\text{-}DAKE}}$ contains the ring signature for a ring of two users instead of the standard signature. Examples of post-quantum ring signatures sizes for a ring of two users at the NIST security level 1 are 2.5 KiB (Raptor [53], a variant of

[16] To be fair, we compare $\Pi_{\mathsf{SC\text{-}DAKE}}$ with a variant of $\Pi_{\mathsf{SC\text{-}AKE}}$ who not sign the first message. Presented in [39], such variant is as secure as $\Pi_{\mathsf{SC\text{-}DAKE}}$ (modulo the difference between weak and perfect forward secrecy), and the main difference of the two schemes is deniability.

Falcon, based on NTRU), 4.4 KiB (DualRing [71], based on M-LWE/SIS), or 3.5 KiB (Calamari [11], based on CSIDH). On the other hand, examples of standard signatures sizes at the same security level are 0.6 KiB (Falcon [63], based on NTRU), 2.3 KiB (Dilithium [54], based on M-LWE/SIS) or 0.26 KiB (CSI-FiSh [12], based on CSIDH). More examples of post-quantum ring signatures can be found in a recent survey [18].

In conclusion, using ring signatures to turn our protocol $\Pi_{\mathsf{SC\text{-}AKE}}$ into a deniable (against semi-honest adversaries) protocol $\Pi_{\mathsf{SC\text{-}DAKE}}$ has a minimal impact on the bandwidth efficiency.

The following theorems state the deniability of our protocols illustrated in Figure 5. Proofs can be found in the full version of this work, see Section 1.3.

Theorem 6.3 (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.*[17]

Theorem 6.4 (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,*[18] *and the key-awareness assumption in Assumption 6.5 holds. Then, the Signal-conforming protocol $\Pi'_{\mathsf{SC\text{-}DAKE}}$ with μ parties is deniable against malicious adversaries.*

Additional Assumption for Deniability Against Malicious Adversaries. We require a knowledge-type assumption to prove deniability of $\Pi'_{\mathsf{SC\text{-}DAKE}}$ against malicious adversaries. Considering that all previous AKE protocols satisfying a strong form of security and deniability require such knowledge-type assumptions [27, 70, 66], this seems unavoidable. On the other hand, there are protocols achieving a strong form of deniability from standard assumptions [30, 64, 65], however, they make a significant compromise in the security such as being vulnerable to KCI attacks and state leakages.

Assumption 6.5 is defined similarly in spirit to the assumptions of Di Raimondo et al. [27] 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[19] like those provide by Fischlin's NIZK [32]. We leave it to future works to investigate the credibility of Assumption 6.5 and those required to prove deniability of the X3DH protocol [66]. We also believe defining a more relaxed notion of deniability that still suffices in practice while also being satisfiable from standard assumptions to be of great importance.

Assumption 6.5 (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 $\mathsf{pp}_{\mathsf{RS}} \in \mathsf{RS.Setup}(1^\kappa)$.*

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).

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

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

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

References

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

[2] Signal protocol: Technical documentation. https://signal.org/docs/. 2, 10

[3] J. Alwen, S. Coretti, and Y. Dodis. The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 129–158. Springer, Heidelberg, May 2019. 2, 3

[4] M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Heidelberg, Aug. 2006. 16

[5] M. Bellare. New proofs for NMAC and HMAC: Security without collision resistance. *Journal of Cryptology*, 28(4):844–878, Oct. 2015. 16

[6] M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 48–62. Springer, Heidelberg, Dec. 2004. 25, 28

[7] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, Aug. 1994. 5

[8] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995. 25, 28

[9] M. Bellare, A. C. Singh, J. Jaeger, M. Nyayapati, and I. Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 619–650. Springer, Heidelberg, Aug. 2017. 2

[10] D. J. Bernstein. Curve25519: New Diffie-Hellman speed records. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer, Heidelberg, Apr. 2006. 4

[11] W. Beullens, S. Katsumata, and F. Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 464–492. Springer, Heidelberg, Dec. 2020. 6, 20

[12] W. Beullens, T. Kleinjung, and F. Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, Dec. 2019. 20

[13] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer, Heidelberg, Dec. 1997. 2, 9

[14] S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In H. Imai and Y. Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 154–170. Springer, Heidelberg, Mar. 1999. 9

[15] X. Bonnetain and A. Schrottenloher. Quantum security analysis of CSIDH. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 493–522. Springer, Heidelberg, May 2020. 5

[16] J. Brendel, R. Fiedler, F. Günther, C. Janson, and D. Stebila. Post-quantum asynchronous deniable key exchange and the signal handshake. Cryptology ePrint Archive, Report 2021/769, 2021. 7

[17] J. Brendel, M. Fischlin, F. Günther, C. Janson, and D. Stebila. Towards post-quantum security for signal's X3DH handshake. In O. Dunkelman, M. J. Jacobson, Jr., and C. O'Flynn, editors, *Selected Areas in Cryptography*, pages 404–430, Cham, 2020. Springer International Publishing. 2, 5

[18] M. Buser, R. Dowsley, M. F. Esgin, C. Gritti, S. K. Kermanshahi, V. Kuchta, J. T. LeGrow, J. K. Liu, R. C.-W. Phan, A. Sakzad, R. Steinfeld, and J. Yu. A survey on exotic signatures for post-quantum blockchain: Challenges & research directions. Cryptology ePrint Archive, Paper 2022/1151, 2022. https://eprint.iacr.org/2022/1151. 20

[19] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, Heidelberg, May 2001. 3, 5, 7, 9

[20] R. Canetti and H. Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 143–161. Springer, Heidelberg, Aug. 2002. https://eprint.iacr.org/2002/120/. 5, 10

[21] D. Cash, E. Kiltz, and V. Shoup. The twin Diffie-Hellman problem and applications. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 127–145. Springer, Heidelberg, Apr. 2008. 4

[22] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the signal messaging protocol. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 451–466, 2017. 2

[23] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the signal messaging protocol. *Journal of Cryptology*, 33(4):1914–1983, 2020. 2

[24] K. Cohn-Gordon, C. Cremers, K. Gjøsteen, H. Jacobsen, and T. Jager. Highly efficient key exchange protocols with optimal tightness. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 767–797. Springer, Heidelberg, Aug. 2019. 5, 7, 9

[25] B. de Kock, K. Gjøsteen, and M. Veroni. Practical isogeny-based key-exchange with optimal tightness. In O. Dunkelman, M. J. Jacobson, Jr., and C. O'Flynn, editors, *Selected Areas in Cryptography*, pages 451–479, Cham, 2020. Springer International Publishing. 5, 10

[26] C. de Saint Guilhem, M. Fischlin, and B. Warinschi. Authentication in key-exchange: Definitions, relations and composition. In L. Jia and R. Küsters, editors, *CSF 2020 Computer Security Foundations Symposium*, pages 288–303. IEEE Computer Society Press, 2020. 7

[27] M. Di Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 400–409. ACM Press, Oct. / Nov. 2006. 3, 6, 20, 25, 28

[28] W. Diffie, P. C. Van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and cryptography*, 2(2):107–125, 1992. 9

[29] S. Dobson and S. D. Galbraith. Post-quantum signal key agreement with SIDH. Cryptology ePrint Archive, Report 2021/1187, 2021. `https://ia.cr/2021/1187`. 7

[30] Y. Dodis, J. Katz, A. Smith, and S. Walfish. Composability and on-line deniability of authentication. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 146–162. Springer, Heidelberg, Mar. 2009. 20

[31] F. B. Durak and S. Vaudenay. Bidirectional asynchronous ratcheted key agreement with linear complexity. In N. Attrapadung and T. Yagi, editors, *IWSEC 19*, volume 11689 of *LNCS*, pages 343–362. Springer, Heidelberg, Aug. 2019. 2

[32] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, Aug. 2005. 20

[33] P.-A. Fouque, D. Pointcheval, and S. Zimmer. HMAC is a randomness extractor and applications to TLS. In M. Abe and V. Gligor, editors, *ASIACCS 08*, pages 21–32. ACM Press, Mar. 2008. 15

[34] E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Heidelberg, Feb. / Mar. 2013. 4

[35] A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 467–484. Springer, Heidelberg, May 2012. 3, 5, 9

[36] A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama. Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In K. Chen, Q. Xie, W. Qiu, N. Li, and W.-G. Tzeng, editors, *ASIACCS 13*, pages 83–94. ACM Press, May 2013. 5

[37] K. Gjøsteen and T. Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 95–125. Springer, Heidelberg, Aug. 2018. 5, 7, 9

[38] S. Guo, P. Kamath, A. Rosen, and K. Sotiraki. Limits on the efficiency of (ring) LWE based non-interactive key exchange. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 374–395. Springer, Heidelberg, May 2020. 2

[39] K. Hashimoto, S. Katsumata, K. Kwiatkowski, and T. Prest. An efficient and generic construction for signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. In J. Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 410–440. Springer, Heidelberg, May 2021. 6, 7, 19

[40] K. Hashimoto, S. Katsumata, K. Kwiatkowski, and T. Prest. An efficient and generic construction for signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. Cryptology ePrint Archive, Report 2021/616, 2021. `https://eprint.iacr.org/2021/616`. 6

[41] K. Hashimoto, S. Katsumata, K. Kwiatkowski, and T. Prest. An efficient and generic construction for signal's handshake (X3DH): post-quantum, state leakage secure, and deniable. *J. Cryptol.*, 35(3):17, 2022. 6, 15

[42] A. Hosoyamada and T. Iwata. On tight quantum security of HMAC and NMAC in the quantum random oracle model. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 585–615, Virtual Event, Aug. 2021. Springer, Heidelberg. 16

[43] K. Hövelmanns, E. Kiltz, S. Schäge, and D. Unruh. Generic authenticated key exchange in the quantum random oracle model. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 389–422. Springer, Heidelberg, May 2020. 5

[44] D. Jost, U. Maurer, and M. Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 159–188. Springer, Heidelberg, May 2019. 2

[45] D. Jost, U. Maurer, and M. Mularczyk. A unified and composable take on ratcheting. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 180–210. Springer, Heidelberg, Dec. 2019. 2

[46] T. Kawashima, K. Takashima, Y. Aikawa, and T. Takagi. An efficient authenticated key exchange from random self-reducibility on CSIDH. In D. Hong, editor, *ICISC 20*, volume 12593 of *LNCS*, pages 58–84. Springer, Heidelberg, Dec. 2020. 5, 10

[47] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, Heidelberg, Aug. 2005. 3, 5, 9

[48] K. Kurosawa and J. Furukawa. 2-pass key exchange protocols from CPA-secure KEM. In J. Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 385–401. Springer, Heidelberg, Feb. 2014. 3, 5

[49] K. Kwiatkowski. An efficient and generic construction for signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. proof of concept implementation, December 2020. https://github.com/post-quantum-cryptography/post-quantum-state-leakage-secure-ake. 3, 15, 16

[50] K. Kwiatkowski. PQ Crypto Catalog, December 2020. https://github.com/kriskwiatkowski/pqc. 16

[51] B. A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In W. Susilo, J. K. Liu, and Y. Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 1–16. Springer, Heidelberg, Nov. 2007. 3, 5, 9

[52] Y. Li and S. Schäge. No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 1343–1360. ACM Press, Oct. / Nov. 2017. 9

[53] X. Lu, M. H. Au, and Z. Zhang. Raptor: A practical lattice-based (linkable) ring signature. In R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 110–130. Springer, Heidelberg, June 2019. 6, 19

[54] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehlé, and S. Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions. 20

[55] M. Marlinspike and T. Perrin. The double ratchet algorithm, November 2016. https://signal.org/docs/specifications/doubleratchet/. 2

[56] M. Marlinspike and T. Perrin. The X3DH key agreement protocol, November 2016. https://signal.org/docs/specifications/x3dh/. 2, 4, 6, 10, 13, 16

[57] S. Myers, M. Sergi, and a. shelat. Blackbox construction of a more than non-malleable CCA1 encryption scheme from plaintext awareness. In I. Visconti and R. D. Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 149–165. Springer, Heidelberg, Sept. 2012. 25, 28

[58] C. Paquin, D. Stebila, and G. Tamvada. Benchmarking post-quantum cryptography in TLS. In J. Ding and J.-P. Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 72–91. Springer, Heidelberg, 2020. 17

[59] C. Peikert. He gives C-sieves on the CSIDH. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 463–492. Springer, Heidelberg, May 2020. 5

[60] T. Perrin. The XEdDSA and VXEdDSA signature schemes, October 2016. https://signal.org/docs/specifications/xeddsa/. 4

[61] B. Poettering and P. Rösler. Towards bidirectional ratcheted key exchange. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 3–32. Springer, Heidelberg, Aug. 2018. 2

[62] D. Pointcheval and O. Sanders. Forward secure non-interactive key exchange. In M. Abdalla and R. D. Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 21–39. Springer, Heidelberg, Sept. 2014. 4

[63] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`. 20

[64] N. Unger and I. Goldberg. Deniable key exchanges for secure messaging. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 1211–1223. ACM Press, Oct. 2015. 2, 20

[65] N. Unger and I. Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *PoPETs*, 2018(1):21–66, Jan. 2018. 2, 20

[66] N. Vatandas, R. Gennaro, B. Ithurburn, and H. Krawczyk. On the cryptographic deniability of the Signal protocol. In M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, editors, *ACNS 20, Part II*, volume 12147 of *LNCS*, pages 188–209. Springer, Heidelberg, Oct. 2020. 2, 3, 6, 20

[67] H. Xue, M. H. Au, R. Yang, B. Liang, and H. Jiang. Compact authenticated key exchange in the quantum random oracle model. Cryptology ePrint Archive, Report 2020/1282, 2020. `https://eprint.iacr.org/2020/1282`. 5

[68] H. Xue, X. Lu, B. Li, B. Liang, and J. He. Understanding and constructing AKE via double-key key encapsulation mechanism. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 158–189. Springer, Heidelberg, Dec. 2018. 5

[69] Z. Yang, Y. Chen, and S. Luo. Two-message key exchange with strong security from ideal lattices. In N. P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 98–115. Springer, Heidelberg, Apr. 2018. 3, 5

[70] A. C.-C. Yao and Y. Zhao. Deniable internet key exchange. In J. Zhou and M. Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 329–348. Springer, Heidelberg, June 2010. 3, 6, 20

[71] T. H. Yuen, M. F. Esgin, J. K. Liu, M. H. Au, and Z. Ding. DualRing: Generic construction of ring signatures with efficient instantiations. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 251–281, Virtual Event, Aug. 2021. Springer, Heidelberg. 6, 20

A Omitted Preliminaries for our First Protocol

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

A.1 Notation

The operator $\oplus$ denotes bit-wise "XOR", and $\|$ denotes string concatenation. For $n \in \mathbb{N}$, we write $[n]$ to denote the set $[n] := \{1, \dots, n\}$. For $j \in [n]$, we write $[n \backslash j]$ to denote the set $[n \backslash j] := \{1, \dots, n\} \setminus \{j\}$. 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.

A.2 Key Encapsulation Mechanisms

Definition A.1 (KEM Schemes). *A key encapsulation mechanism (KEM) scheme with session key space $\mathcal{KS}$ consists of the following four PPT algorithms $\Pi_{\mathsf{KEM}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$:*

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

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

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

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

Definition A.2 (**$(1-\delta)$-Correctness**)**.** *We say a KEM scheme Π_{KEM} is $(1-\delta)$-correct if for all $\kappa \in \mathbb{N}$ and $\mathsf{pp} \in \mathsf{Setup}(1^\kappa)$,*

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

Definition A.3 (IND-CPA **and** IND-CCA **Security**)**.** *Let κ be a security parameter, $\Pi_{\mathsf{KEM}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ be a KEM scheme and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, we define the advantage of $\mathcal{A}$ as*

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

where

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

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

Definition A.4 (Min-Entropy of KEM Encapsulation Key). *We say a KEM scheme Π_{KEM} has ν-high encapsulation key min-entropy if for all $\kappa \in \mathbb{N}$ and $\mathsf{pp} \in \mathsf{Setup}(1^\kappa)$,*

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

Definition A.5 (Min-Entropy of KEM Ciphertext). *We say a KEM scheme Π_{KEM} has χ-high* ciphertext *min-entropy if for all $\kappa \in \mathbb{N}$ and $\mathsf{pp} \in \mathsf{Setup}(1^\kappa)$,*

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

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

We define plaintext-awareness (PA) for KEM schemes [8, 6] where multiple keys are considered [57]. We observe that the standard PA security defined for a single key does not immediately imply a multi-key variant and that the original proof of deniability by Di Raimondo et al. [27, Theorem 2 and 3] crucially relies on the multi-key variant. Furthermore, below we consider strengthening of the (already strong) PA security where the efficient extractor $\mathcal{E}_{\mathcal{C}}$ for the ciphertext creator $\mathcal{C}$ can be constructed efficiently given the description of $\mathcal{C}$. This is required in our deniability proof as the simulator must construct such $\mathcal{E}_{\mathcal{C}}$ given the description of the adversary.

A.3 Digital Signatures

Definition A.6 (Signature Schemes). *A signature scheme with message space $\mathcal{M}$ consists of the following four PPT algorithms $\Pi_{\mathsf{SIG}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:*

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

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

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

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

Definition A.7 ($(1-\delta)$-Correctness). *We say a signature scheme Π_{SIG} is $(1-\delta)$-correct if for all $\kappa \in \mathbb{N}$, all messages $\mathsf{M} \in \mathcal{M}$ and all $\mathsf{pp} \in \mathsf{Setup}(1^\kappa)$,*

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

Definition A.8 ($\mathsf{EUF\text{-}CMA}$ Security). *Let κ be a security parameter, $\Pi_{\mathsf{SIG}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme and $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ as*

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

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

A.4 Pseudo-Random Functions

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

Definition A.9 (Pseudo-Random Function Family). *Let $\mathcal{A}$ be an adversary that is given oracle access to either $\mathsf{F}_\mathsf{K}(\cdot) := \mathsf{F}(\mathsf{K}, \cdot)$ for $\mathsf{K} \leftarrow\$\mathcal{FK}$ or a truly random function $\mathsf{RF} : \mathcal{D} \to \mathcal{R}$. We define the advantage of $\mathcal{A}$ as*

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

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

A.5 Strong Randomness Extractors

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

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

Definition A.10 (Strong Randomness Extractors). *Let $\mathsf{Ext} : \mathcal{S} \times \mathcal{D} \to \mathcal{R}$ be a family of efficiently computable functions with set $\mathcal{S}$, domain $\mathcal{D}$ and range $\mathcal{R}$, all with finite size. A function family Ext is a strong $(\lambda, \varepsilon_{\mathsf{Ext}})$-extractor if for any random variable X over $\mathcal{D}$ with $\Pr[X = x] \leq 2^{-\lambda}$ (i.e., X has min-entropy at least λ), if s and R are chosen uniformly at random from $\mathcal{S}$ and $\mathcal{R}$, respectively, the two distributions $(s, \mathsf{Ext}_s(X))$ and (s, R) are within statistical distance $\varepsilon_{\mathsf{Ext}}$, that is*

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

A.6 $(1-\delta)$-Correctness

Definition A.11 **($(1-\delta)$-Correctness).** *An AKE protocol Π_{AKE} is $(1-\delta)$-correct if for any set with a valid pairing $S \subseteq ([\mu] \times [\ell])^2$, when we execute the AKE protocol faithfully between all the oracle pairs included in S, it holds that*

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

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

B Omitted Preliminaries for our Deniable Protocols

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

B.1 Ring Signatures

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

B.2 Plaintext-Awareness

We define plaintext-awareness (PA) for KEM schemes [8, 6] where multiple keys are considered [57]. We observe that the standard PA security defined for a single key does not immediately imply a multi-key variant and that the original proof of deniability by Di Raimondo et al. [27, Theorem 2 and 3] crucially relies on the multi-key variant. Furthermore, below we consider strengthening of the (already strong) PA security where the efficient extractor $\mathcal{E}_{\mathcal{C}}$ for the ciphertext creator $\mathcal{C}$ can be constructed efficiently given the description of $\mathcal{C}$. This is required in the deniability proof as the simulator must construct such $\mathcal{E}_{\mathcal{C}}$ given the description of the adversary $\mathcal{M}$.

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

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

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

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

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

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

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

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

(ii) $\mathcal{C}$ *can adaptively query an index-ciphertext pair* C *polynomially many times to the challenger. When the challenger receives* (i, C)*, it returns* $\mathcal{E}_{\mathcal{C}}(\mathtt{query}, (i, \mathsf{C}), \mathsf{rand}_{\mathcal{C}})$*.*[20]

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

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

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

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

B.3 Non-Interactive Zero-Knowledge

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

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

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

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

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

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

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

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

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

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

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

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