

A Comprehensive Study of the Signal Handshake Protocol: Bundled Authenticated Key Exchange

Keitaro Hashimoto¹, Shuichi Katsumata^{1,2}, Guilhem Niot^{2,3}, Ida Tucker², and Thom Wiggers²

¹ AIST

² PQShield

³ Univ Rennes, CNRS, IRISA

Abstract The Signal protocol relies on a special handshake protocol, formerly X3DH and now PQXDH, to set up secure conversations. One of its privacy properties, of value to Signal, is *deniability*, allowing users to deny participation in communications. Prior analyses of these protocols (or proposals for post-quantum alternatives) have all used highly tailored models to the individual protocols and generally made ad-hoc adaptations to “standard” AKE definitions, making the concrete security attained unclear and hard to compare between similar protocols. Indeed, we observe that some natural Signal handshake protocols cannot be handled by these tailored models. In this work, we introduce *Bundled Authenticated Key Exchange* (BAKE), a concrete treatment of the Signal handshake protocol. We formally model prekey *bundles* and states, enabling us to define various levels of security in a unified model, along with a framework for analyzing deniability. We analyze Signal’s classically secure X3DH and *harvest-now-decrypt-later*-secure PQXDH, and show that they do not achieve what we call *optimal* security (as is documented). Regarding deniability, we show that PQXDH is deniable against harvest-now-*judge*-later attacks, where a quantum judge retrospectively assesses the participation of classical users. Next, we introduce RingXKEM, a fully post-quantum Signal handshake protocol achieving optimal security; as RingXKEM shares states among many prekey bundles, it could not have been captured by prior models. Motivated by our deniability analysis of RingXKEM we introduce a novel metric inspired by differential privacy, providing relaxed, pragmatic guarantees for deniability. We also use this metric to define *deniability* for RS, a relaxation of anonymity, allowing us to build an efficient RS from NIST-standardized Falcon (and MAYO), which is not anonymous, but is provably deniable. Lastly, we provide security, deniability and efficiency comparisons of X3DH, PQXDH, and RingXKEM.

1 Introduction

The Signal protocol [MP16; PM16] does not just power the Signal app, it also underpins messaging apps such as WhatsApp [Wha23], Google RCS [Goo22], and Facebook Messenger [Met23], collectively serving billions of users. To initiate a conversation, Signal users perform a handshake protocol to establish a shared key, which is then used for encrypted communication via the Double Ratchet protocol [PM16]. This handshake was originally implemented as X3DH [MP16], based on Triple Diffie-Hellman [KP05]. In late 2023, as a step towards fully post-quantum (PQ) security, X3DH was replaced with PQXDH [KS23], offering protection against “harvest-now-decrypt-later” (HNDL) attacks. There also exist several proposals for fully PQ Signal handshake protocols [Bre+22; Col+24; Has+22].

Until this work, analysis of Signal handshake protocols were performed in ad-hoc security models, sometimes deviating from the way they will be implemented in practice, e.g., assuming one-time prekey bundles never deplete. This made the concrete security properties attained unclear and hindered comparisons of the strengths and weaknesses of different protocols. We propose *Bundled Authenticated Key Exchange* (BAKE) protocols, allowing to analyze existing Signal handshake protocols in a unified manner. This is a modification to the standard AKE definition, with a focus on a more general and formal handling of *prekey bundles*; a distinct component of Signal handshake protocols, allowing users to upload batches of key materials onto the server so that any sender can establish communication even when recipients are offline. We define a security model for BAKE, treating the properties of key indistinguishability, authentication and deniability. This framework leads us to present a fully post-quantum Signal handshake protocol called RingXKEM, relying on ring signatures (RS) and Merkle trees, which could not have been captured in previous models.

1.1 Contributions

In this work, we provide a concrete treatment of the Signal handshake protocol. We formally model prekey bundles and their states, enabling us to capture new Signal handshake protocols while establishing various levels of security within a unified framework. We further propose a framework for analyzing deniability of BAKE protocols. We compare both the security and deniability properties, and performance of X3DH, PQXDH, and our new proposal RingXKEM in this

framework. Our deniability analysis of said protocols proves for the first time that PQXDH is deniable even against *quantum* distinguishers. Indeed, transcripts stored now could later be exploited when quantum capabilities become available, a risk we term “harvest-now-judge-later” (HNJL).⁴ Additionally, we provide instantiations of ring signatures based on NIST-standardized signatures, to encourage adoption of RingXKEM.

Unique features of our model. By defining the BAKE formalism, we can capture the lifecycle of prekey bundles. Each batch of prekey bundles contains a number of *one-time* prekey bundles, and a single *last-resort* prekey bundle. One-time prekey bundles are deleted after use. The last-resort prekey bundle ensures recipient availability even if they are offline for extended amounts of time; it is only used if all one-time prekey bundles are used up, and is only deleted when a new batch of prekeys is uploaded [KS23; MP16]. Our model distinguishes security guarantees based on whether one-time prekeys are depleted, revealing interesting separations. For example, in X3DH and PQXDH, using last-resort prekeys downgrades key indistinguishability security, and the attained deniability.

Moreover, when a batch of prekey bundles is generated, a *user state* is generated — a unique feature of the BAKE formalism, allowing the secret information associated to each prekey bundle to be correlated. This state is then updated after each key exchange. For an example, in X3DH and PQXDH, the secret associated to the one-time prekey bundle is deleted after the receiver completes a key exchange. Leakage of a users’ updated state may thus reveal how many handshake protocols it has executed as a receiver. We require that said leakage does not expose the sender identities for those handshakes, preserving deniability with respect to the communicating parties. As deniability under standard AKE formalism do not track persistent user states, this subtlety is absent in prior definitions.

A Post-Quantum Signal Handshake Protocol. In Section 5, we present a fully post-quantum Signal handshake protocol called RingXKEM. This protocol relies on post-quantum ring signatures for deniable post-quantum authentication and post-quantum KEM key exchange for post-quantum secrecy, and was inspired by prior proposals [Bre+22; Has+21; Has+22]. We optimize prekey bundle storage by authenticating them using a Merkle tree, the root of which is signed by the identity key. This way, the server needs to store only a single large post-quantum signature instead of one per prekey bundle. This reduces the cost of uploading prekey bundles and the deployment of post-quantum authentication at the central server. It is worth highlighting that as RingXKEM shares states across many prekey bundles, it could not have been captured in previous models. Lastly, RingXKEM achieves optimal security, and solid deniability guarantees in our BAKE model against fully quantum adversaries. In Section 5 of [Kat+25], we also examine SignXKEM, a RingXKEM variant suggested in [Has+21], which replaces RSs by (plain) signatures, and show that it offers some level of deniability under limited leakage and disclosure, highlighting the precise nature of our model in capturing weaker notions of deniability.

A pragmatic metric for deniability and new ring signature instantiations. In Section 6.5, we introduce a novel measure for deniability inspired by concepts in differential privacy and differential indistinguishability [Bac+15; Dwo+06; Mir+09]. In contrast to prior works, this new metric captures the intuition that the accused user only needs to prove that a simulator *could have* generated the evidence, not that the simulator outputs evidence with the same probability as the accused user.

We observe that, thanks to our new metric for measuring the deniability of BAKE protocols, a weaker notion than anonymity, coined *deniability* for RS, is sufficient for constructing deniable BAKE protocols. This relaxation enables us to design PQ RS schemes for small ring sizes, based on the standardized signature Falcon [Pre+22], and the signature candidate MAYO [Beu+24]. These RS schemes are as compact as the state-of-the-art. We provide implementations outperforming previous works by factors 32–66× for signing, and 146–1025× for verification.

Instantiations and efficiency comparison. We compare X3DH, PQXDH, and RingXKEM when instantiated with cryptographic primitives. For X3DH and PQXDH, we base the numbers on the deployed protocols. For RingXKEM, we base our numbers on the Gandalf RS scheme [GJK24b], as well as our new RSs from Falcon and MAYO. These results show that RingXKEM can be deployed at a cost comparable to PQXDH, especially when considering the cost of storage of prekey bundles.

1.2 Full versions and additional resources

Due to page limitations, this work is a shortened, and merged, version of previous works [HKW25; Kat+25]. The initial [HKW25], accepted in the first cycle of USENIX’25, introduces the BAKE syntax, along with associated correctness,

⁴ To avoid confusion with the acronym of harvest-now-decrypt-later, we chose the term *judge*⁸ as opposed to *distinguish*.

key-indistinguishability and authentication properties. It also introduces the RingXKEM protocol. Then, [Kat+25], accepted in the second cycle of USENIX'25, provides a unified framework for analyzing the deniability of BAKE protocols, and introduces the aforementioned ring signature schemes from Falcon and MAYO. Implementations of our Ring Signature schemes accompanying this paper can be found at doi.org/10.5281/zenodo.15571694.

2 Syntax of Bundled Authenticated Key Exchange

In this section, we define the syntax for a (two-round) *bundled authenticated key exchange* (BAKE) protocol. This definition is tailored to the semantics and flow of Signal handshake protocols like X3DH. While we build on prior approaches (e.g., [Bre+22; Coh+17; Coh+20; Col+24; FG24; Has+21; Has+22]), our concrete modeling of the uploading of prekey bundles and the users' state allow a more formal modeling of forward secrecy and state reuse.

We give our syntax for BAKE protocols in [Definition 2.1](#). Signal protocols pre-generate and publish a number of so-called *prekey bundles* to the central server, which can be viewed as the first message in standard AKE. We model this through the BAKE.PreKeyBundleGen function, which is the most significant difference to prior models; prior work typically treated prekey bundles individually. This function explicitly returns a single state that contains all (private) information for the prekey bundles. We use this to model attacks on the ephemeral keys stored by the users. In the second round of the key agreement, the person that wants to start a conversation, whom we refer to as *sender*, downloads a prekey bundle and uses it to complete the cryptographic handshake and obtain a shared secret to encrypt their message with. This is modeled by the BAKE.Send function. Finally, the *receiver* (whose previously uploaded prekey bundle was used by the sender) takes this generated message and its current state to complete the handshake in BAKE.Receive.

Definition 2.1. A two-round bundled authenticated key exchange protocol BAKE consists of the following four PPT algorithms, where $L \in \text{poly}(\lambda)$.

BAKE.IdKeyGen(1^λ) $\xrightarrow{\$}$ (ik, isk): The identity key generation algorithm takes as input the security parameter 1^λ and outputs an identity public key ik and secret key isk.

BAKE.PreKeyBundleGen(isk_u) $\xrightarrow{\$}$ ($\vec{\text{prek}}_u, \text{st}_u$): The prekey bundle generation algorithm takes a user u 's identity secret key as input and outputs a number of prekey bundles $\vec{\text{prek}}_u = (\text{prek}_{u,t})_{t \in [L] \cup \{\perp\}}$, and a user state st_u . Prekey bundles with $t \neq \perp$ are called one-time prekey bundles and the special prekey bundle $\text{prek}_{u,\perp}$ is called the last-resort prekey bundle. The state may for example include the associated (ephemeral) secret keys to public keys in $\vec{\text{prek}}_u$.

BAKE.Send(isk_s, ik_r, $\text{prek}_{r,t}$) $\xrightarrow{\$}$ (K, ρ): The sender algorithm takes as input a sender's identity secret key isk_s and the intended receiver r 's identity key ik_r and a particular prekey bundle $\text{prek}_{r,t}$, and outputs a session key K and a handshake message ρ.

BAKE.Receive(isk_r, st_r, ik_s, t, ρ) \rightarrow (K', st_r): The (deterministic) receiver algorithm takes as input a receiver r 's identity secret key isk_r and state st_r, a sender's identity key ik_s, along with the identifier of the used prekey bundle $t \in [L] \cup \{\perp\}$, and the initial message ρ. It then outputs a key K' and a possibly updated state st_r. Key agreement may fail, in which case $K' = \perp$ is returned, and the state is rolled back to before running the algorithm.

A Single State for Prekey Bundles. BAKE protocols use a single state for all prekey bundles uploaded by a single BAKE.PreKeyBundleGen call. We use this state to model forward secrecy and deniability properties related to state compromises leaking the private keys for prekey bundles that have not been used and deleted. The singular shared state is one of the functionalities missing in prior formalization. Looking ahead, our fully post-quantum Signal handshake protocol RingXKEM could not have been captured by prior work as prekey bundles were treated independently. Running the BAKE.PreKeyBundleGen algorithm refreshes all prekey bundles and the state. Signal clients call this function frequently, to ensure enough prekey bundles are available at the server, and to rotate last-resort prekey bundles, which we describe in the next paragraph. In our security model, described in [Section 3.3 of \[HKW25\]](#), we use *epochs* to track the expiration of state secret key material.

Availability Versus Ephemeral Keys. If each prekey bundle were single use, the number of prekey bundles uploaded would pose a limit on the number of Signal handshakes that can be completed. Thus, to ensure availability of the recipient even if they are offline for extended amounts of time, so-called *last-resort prekey bundles* are used if the list of *one-time prekey bundles* is depleted. The last-resort prekey bundle is a specially designated prekey bundle which, when used, is

not deleted from the list of available prekeys at the server, and its associated secrets are not deleted from the receiver’s state. Because of this, exchanges that use the last-resort prekey bundle are vulnerable to state compromise even after the handshake completes, until the next call of `BAKE.PreKeyBundleGen`, which replaces the last-resort prekey bundle and the receiver’s state. For bookkeeping in our models, we use the label \perp to refer to a last-resort prekey bundle. In protocol execution, the server will first distribute all one-time prekey bundles, and only once these are exhausted will the last-resort prekey bundle $\text{prek}_{u,\perp}$ be used.

3 Correctness and Security of Bundled AKE

In Section 3 of [HKW25], we formally define the correctness and security of a BAKE protocol. We hereafter highlight some novelties of this security model.

A security model for BAKE. Based on our BAKE syntax, we define a game-based security model that treats key indistinguishability and authentication properties separately. For key indistinguishability, the adversary can reveal both the long-term identity secret keys and states associated to the prekey bundles. However, allowing it to reveal secrets without restrictions leads to an *unavoidable attack* on key indistinguishability. We thus exclude the minimum set of all such unavoidable attacks that any BAKE protocol is vulnerable against, and define the *optimal* confidentiality properties of a BAKE protocol. If a specific protocol has further (accepted) weaknesses, we can include them as additional unavoidable attacks⁵. By comparing the unavoidable attacks for different protocols, we get an immediate means of comparing their achieved security properties.

Explicit treatment of authentication. During the development of PQXDH, Bhargavan et al. discovered that the protocol is vulnerable to so-called “KEM re-encapsulation attacks” if the encapsulation key is not bound to the key exchange [Bha+24]. This attack forces two users to establish the same key, unknown to the adversary, while disagreeing on the encapsulation key being used. This was previously considered an implicit attack on key indistinguishability, though not immediately clear why key indistinguishability should fail. Another subtle attack is the potential replaying of messages, which the documentation mentions as a possibility and defers the analysis to be beyond the scope of the document [MP16, Sec. 4.2]. While Signal implements a countermeasure, replays seemingly were not covered by prior game-based security models as they do not break key indistinguishability and are very specific to the treatment of the so-called *last-resort*⁶ prekey bundles. (The one exception is [KBB17], which covers this using symbolic analysis.) In our work, we treat authentication as a primary goal, making it possible to capture both attacks as explicit breaks of authentication.

Classic, harvest-now-decrypt-later, and quantum adversaries. We capture not only classical and quantum adversaries to key indistinguishability and authentication, but also the intermediate HNDL adversary. Indeed, the adversary’s powers are adjusted depending on the attempted attack: while certain attacks are unavoidable if the adversary is quantum from the outset of the security game, they may become avoidable assuming the adversary is classical up to some point. To our knowledge, ours is the first work to formally model what it means for a general Signal handshake protocol to be HNDL secure. Such a fine-grained security model is essential to formally proving security of PQXDH. We note that while there are some works [Bha+24; FG24] showing (a slight variant of) HNDL security of PQXDH, the security model is highly tailored to PQXDH and is non-reusable for general protocols.

4 Signal’s X3DH and PQXDH as BAKE protocols

The X3DH protocol [MP16] was proposed in 2016 by Marlinspike and Perrin based on the Triple Diffie–Hellman AKE protocol [KP05]. In 2023, Signal introduced PQXDH to protect the Signal handshake protocol against HNDL attacks [KS23]. In this section, we will first describe X3DH and PQXDH as BAKE protocols, then we discuss their security.

4.1 Descriptions of X3DH and PQXDH

The descriptions of X3DH and PQXDH are given in Algs. 1 to 3. As PQXDH essentially adds a post-quantum KEM to X3DH, it is described in the same figures, marked with a gray dotted box. The key agreement in these protocols proceeds

⁵ For instance, Signal’s X3DH and PQXDH have some documented and accepted weaknesses in specific powerful compromise scenarios. We detail these in Section 3.5 of [HKW25], and exclude them from our security analysis.

⁶ Following the Signal source code and the specification for PQXDH.

```

1: function  $\overline{\text{PQX3DH}}$ .IdKeyGen( $1^\lambda$ )
2:    $\overline{\text{isk}} \xleftarrow{\$} \mathbb{Z}_p^s$ ;  $\overline{\text{ik}} := [\overline{\text{isk}}]G$ 
3:    $(\text{vk}, \text{sk}) \leftarrow \text{Sig.KeyGen}(1^\lambda)$ 
4:   return  $(\text{ik} := (\overline{\text{ik}}, \text{vk}), \text{isk} := (\overline{\text{isk}}, \text{sk}))$ 
5: function  $\overline{\text{PQX3DH}}$ .PreKeyBundleGen( $\text{isk}_u$ )
6:    $(\overline{\text{isk}}_u, \text{sk}_u) \leftarrow \text{isk}_u$ 
7:    $D_{\text{prek}}, D_{\rho_\perp} := \emptyset$   $\triangleright$  Initialize empty lists
8:    $\triangleright$  Generate what Signal calls the signed prekey
9:    $\text{spksec}_u \xleftarrow{\$} \mathbb{Z}_p$ ;  $\text{spk}_u := [\text{spksec}]G$ 
10:   $\sigma_{\text{spk}_u} \leftarrow \text{Sig.Sign}(\text{sk}_u, \text{spk}_u)$ 
11:   $\triangleright$  Create the  $L$  one-time prekey bundles
12:  for  $t \in [L]$  do
13:     $\text{osk}_{u,t} \xleftarrow{\$} \mathbb{Z}_p$ ;  $\text{opk}_{u,t} := [\text{osk}_{u,t}]G$ 
14:     $(\text{ek}_{u,t}, \text{dk}_{u,t}) \xleftarrow{\$} \text{KEM.KeyGen}(1^\lambda)$ 
15:     $\sigma_{\text{ek}_{u,t}} \leftarrow \text{Sig.Sign}(\text{sk}_u, \text{ek}_{u,t})$ 
16:     $\text{prek}_{u,t} := (\text{spk}_u, \sigma_{\text{spk}_u}, \text{opk}_{u,t}, (\text{ek}_{u,t}, \sigma_{\text{ek}_{u,t}}))$ 
17:     $D_{\text{prek}}[t] \leftarrow (\text{prek}_{u,t}, (\text{spksec}_u, \text{osk}_{u,t}, \text{dk}_{u,t}))$ 
18:   $\triangleright$  Set up the last-resort prekey bundle
19:   $(\text{ek}_{u,\perp}, \text{dk}_{u,\perp}) \xleftarrow{\$} \text{KEM.KeyGen}(1^\lambda)$ 
20:   $\sigma_{\text{ek}_{u,\perp}} \leftarrow \text{Sig.Sign}(\text{sk}_u, \text{ek}_{u,\perp})$ 
21:   $\text{prek}_{u,\perp} := (\text{spk}_u, \sigma_{\text{spk}_u}, \perp, (\text{ek}_{u,\perp}, \sigma_{\text{ek}_{u,\perp}}))$ 
22:   $D_{\text{prek}}[\perp] \leftarrow (\text{prek}_{u,\perp}, (\text{spksec}_u, \perp, \text{dk}_{u,\perp}))$ 
23:  return  $(\overline{\text{prek}}_u, \text{st}_u := (D_{\text{prek}}, D_{\rho_\perp}))$ 

```

Algorithm 1: $\overline{\text{PQX3DH}}$ identity key and prekey bundle generation algorithms.

roughly as follows. The identity keys of both users are Diffie–Hellman (DH) values. The prekey bundle contains a signed DH key, and, if it is a one-time prekey bundle, an ephemeral DH key. Finally, the sender generates an ephemeral key. These keys are used pairwise in DH computations before combining them into a shared secret ss (c.f. [Alg. 2, Lines 6 to 13](#)).

While our description of X3DH and PQXDH closely follow Signal’s documentation [[KS23](#); [MPI16](#)], we incorporated several minor modifications based on discussions with Signal developers that may be included in future updates [[Sch24](#)]. We detail these modifications in [Section 4.1 of \[HKW25\]](#).

4.2 HNDL-Security for PQXDH

PQXDH only attempts to give post-quantum security against HNDL attacks, and thus still relies on elliptic curve cryptography for authentication. While the identity keys are the same as X3DH, signed post-quantum KEM keys are added to the prekey bundles. In the functions $\overline{\text{PQX3DH}}$.Send and $\overline{\text{PQX3DH}}$.Receive one can see how these additional KEM keys are used to inject a KEM-encapsulated quantum-safe shared secret into the key returned by the handshake.

Note that although the Signal specification and implementation of PQXDH supports prekey bundles without KEM prekeys (as this gives backwards compatibility with X3DH), we do not model this.⁷ Classic security of PQXDH without KEM prekeys follows directly from X3DH.

Downgrade resilience of PQXDH. As long as PQXDH clients do not enforce the usage of KEM prekeys, i.e., run in “compatibility mode”, a network attacker or malicious server may omit them from prekey bundles and force a classically-secure session. This is because the prekey bundle’s composition is not authenticated. Though it appears receivers might notice that prekey bundle prek_t contained a KEM prekey when it was generated, in the Signal implementation, prekey bundles are actually assembled piece-wise on the server and the DH and KEM (one-time) prekeys are individually identified (i.e., in practice identifier t can be considered a tuple $(t_{\text{DH}}, t_{\text{KEM}})$). The protocols do not try to authenticate protocol version or algorithms supported by the sender or receiver, as, e.g., the TLS 1.3 handshake does [[Res18](#)]. That means that the sender and receiver will each assume the other only supported X3DH if the KEM prekeys are just omitted. As X3DH was not designed with negotiation in mind, this issue can seemingly not be prevented without sacrificing backwards compatibility.

4.3 Security Overview

The correctness of X3DH and PQXDH follows from construction. Below, we state the key indistinguishability of PQXDH, and explain how X3DH differs in the full version.

In [Theorem 2 of \[HKW25\]](#) we show that PQXDH is key indistinguishable against a harvest-now-decrypt-later adversary with respect to the set of “avoidable” attacks, which translates to all the allowed adversary attack strategies

⁷ This DH-only mode will eventually be disabled [[Sch24](#)].

```

1: function PQX3DH.Send( $\text{isk}_s, \text{ik}_r, \text{prek}_r$ )
2:  $(\overline{\text{isk}}_s, \text{sk}_r) \leftarrow \text{isk}_s; (\overline{\text{ik}}_r, \text{vk}_r) \leftarrow \text{ik}_r$ 
3:  $(\text{spk}_r, \sigma_{\text{spk}_r}, \text{opk}_r, \overline{\text{ek}}_r, \sigma_{\overline{\text{ek}}_r}) \leftarrow \text{prek}_r \triangleright \text{opk}_r = \perp$ 
   if  $\text{prek}_r$  is a last-resort key bundle
4: require  $\llbracket \text{Sig.Verify}(\text{vk}_r, \text{spk}_r, \sigma_{\text{spk}_r}) = 1 \rrbracket$ 
5: require  $\llbracket \text{Sig.Verify}(\text{vk}_r, \overline{\text{ek}}_r, \sigma_{\overline{\text{ek}}_r}) = 1 \rrbracket$ 
6:  $\text{esk} \leftarrow \mathbb{Z}_p, \text{epk} := [\text{esk}]_G$ 
7:  $\text{ss}_1 := [\overline{\text{isk}}_s]_{\text{spk}_r}$ 
8:  $\text{ss}_2 := [\text{esk}]_{\text{ik}_r}$ 
9:  $\text{ss}_3 := [\text{esk}]_{\text{spk}_r}$ 
10:  $\text{ss} := \text{ss}_1 \parallel \text{ss}_2 \parallel \text{ss}_3$ 
11: if  $\llbracket \text{opk}_r \neq \perp \rrbracket$  then  $\triangleright$  One-time prekey bundle
12:    $\text{ss}_4 := [\text{esk}]_{\text{opk}_r}$ 
13:    $\text{ss} := \text{ss}_1 \parallel \text{ss}_2 \parallel \text{ss}_3 \parallel \text{ss}_4$ 
14:  $(\text{ss}_{\text{KEM}}, \text{ct}) \leftarrow \text{KEM.Encaps}(\overline{\text{ek}}_r)$ 
15:  $\text{content} := \text{isk}_s \parallel \text{ik}_r \parallel \text{prek}_r \parallel \text{epk} \parallel \text{ct}$ 
16:  $K \parallel \tau_{\text{conf}} := \text{KDF}(\text{ss}_1 \parallel \text{ss}_{\text{KEM}}, \text{content})$ 
17:  $\rho := (\text{epk}, \text{ct}, \tau_{\text{conf}})$ 
18: return  $(K, \rho)$ 

```

Algorithm 2: PQX3DH sender algorithms. prek is not indexed by $t \in [L] \cup \{\perp\}$ as they are oblivious to the sender.

```

1: function PQX3DH.Receive( $\text{isk}_r, \text{st}_r, \text{ik}_s, t, \rho$ )
2:  $(\overline{\text{isk}}_r, \text{sk}_r) \leftarrow \text{isk}_r; (\overline{\text{ik}}_s, \text{vk}_s) \leftarrow \text{ik}_s$ 
3:  $(D_{\text{prek}}, D_{\rho_\perp}) \leftarrow \text{st}_r$ 
4: if  $\llbracket t \neq \perp \rrbracket$  then  $\triangleright$  One-time prekey bundle
5:   require  $\llbracket D_{\text{prek}}[t] \neq \perp \rrbracket \triangleright$  Check if unused.
6:    $(\text{prek}_{r,t}, (\text{spksec}_r, \text{osk}_{r,t}, \overline{\text{dk}}_{r,t})) \leftarrow D_{\text{prek}}[t]$ 
7: else  $\triangleright$  Last-resort prekey bundle (i.e.,  $t = \perp$ )
8:   require  $\llbracket \rho \notin D_{\rho_\perp} \rrbracket \triangleright$  Check  $\rho$  is not replayed.
9:    $D_{\rho_\perp} \leftarrow D_{\rho_\perp} \cup \{\rho\}$ 
10:   $(\text{prek}_{r,t}, (\text{spksec}_r, \perp, \overline{\text{dk}}_{r,t})) \leftarrow D_{\text{prek}}[t]$ 
11:   $(\text{epk}, \text{ct}, \tau_{\text{conf}}) \leftarrow \rho$ 
12:   $\text{ss}_1 := [\text{spksec}_r]_{\overline{\text{ik}}_s}; \text{ss}_2 := [\overline{\text{isk}}_r]_{\text{epk}}$ 
13:   $\text{ss}_3 := [\text{spksec}_r]_{\text{epk}}; \text{ss} := \text{ss}_1 \parallel \text{ss}_2 \parallel \text{ss}_3$ 
14:  if  $\llbracket t \neq \perp \rrbracket$  then  $\triangleright$  One-time prekey bundle
15:     $\text{ss}_4 := [\text{osk}_{r,t}]_{\text{epk}}; \text{ss} := \text{ss}_1 \parallel \text{ss}_2 \parallel \text{ss}_3 \parallel \text{ss}_4$ 
16:   $\text{ss}_{\text{KEM}} \leftarrow \text{KEM.Decaps}(\overline{\text{dk}}_{r,t}, \text{ct})$ 
17:   $\text{content} := \text{isk}_s \parallel \text{ik}_r \parallel \text{prek}_{r,t} \parallel \text{epk} \parallel \text{ct}$ 
18:   $K \parallel \tau'_{\text{conf}} := \text{KDF}(\text{ss}_1 \parallel \text{ss}_{\text{KEM}}, \text{content})$ 
19:  require  $\llbracket \tau_{\text{conf}} = \tau'_{\text{conf}} \rrbracket$ 
20:   $\triangleright$  Delete prekey bundle if not last-resort
21:  if  $\llbracket t \neq \perp \rrbracket$  then  $D_{\text{prek}}[t] \leftarrow \perp$ 
22:   $\text{st}_r \leftarrow (D_{\text{prek}}, D_{\rho_\perp})$ 
23:  return  $(K, \text{st}_r)$ 

```

Algorithm 3: PQX3DH receiver algorithms.

(cf. Table 4 of [HKW25]). We thus prove the advantage is negligible for each of these strategies. In particular, though PQXDH offers security against a class of HNDL adversaries, if the classical adversary compromises the post-quantum KEM prekeys, then it cannot offer HNDL security as all the remaining security comes from classical primitives.

5 Our Post-Quantum RingXKEM

In this section, we propose a post-quantum BAKE protocol RingXKEM which meets the optimal confidentiality properties of a BAKE protocol. Its' core design is inspired from the *deniable* AKE protocol by Hashimoto et al. [Has+21; Has+22] based on ring signatures. We extend it to meet the BAKE syntax and optimize it using Merkle trees to save on receiver bandwidth and server storage. Regarding security, we provide formal statements and proofs in Appendix D of [HKW25].

5.1 Description of RingXKEM

The description of RingXKEM is given in Algs. 4 to 6. The construction is based on a KDF, Merkle tree, KEM, and an RS. If we ignore the Merkle tree for a moment, used only for optimization purposes, the construction is quite simple. The t^{th} ($t \in [L] \cup \{\perp\}$) prekey bundle consists of a KEM public key $\overline{\text{ek}}_t$, a (ring) signature on the $\overline{\text{ek}}_t$, and a ring signature verification key rvk . Here, rvk is shared by all $L + 1$ prekey bundles and the associated signing key rsk is discarded. A sender, after checking validity of $\overline{\text{ek}}_t$, will generate two KEM ciphertexts ct and $\widehat{\text{ct}}$: one associated to ek included in the receiver's identity key and the other to $\overline{\text{ek}}_t$. It then generates a ring signature σ with the ring $\{\text{rvk}_s, \text{rvk}\}$, where the message is ct and $\widehat{\text{ct}}$ along with additional public information. Lastly, the sender derives a session key K and an SKE key K_{ske} from the KEM session keys ss and $\widehat{\text{ss}}$, encrypts σ using K_{ske} as ct_{ske} , and sends the handshake message $\rho = (\text{ct}, \widehat{\text{ct}}, \text{ct}_{\text{ske}})$. The receiver can process ρ using the KEM secret keys.

This vanilla construction requires users to upload $L + 1$ (ring) signatures to the server. While this is also the case for PQXDH, it becomes problematic in RingXKEM when targeting post-quantum security, where signatures can be an

```

1: function RingXKEM.IdKeyGen( $1^\lambda$ )
2:    $(ek, dk) \xleftarrow{\$} \text{KEM.KeyGen}(1^\lambda)$ 
3:    $(rvk, rsk) \xleftarrow{\$} \text{RS.KeyGen}(1^\lambda)$ 
4:   return  $(ik := (ek, rvk), isk := (dk, rsk))$ 
5: function RingXKEM.PreKeyBundleGen( $isk_u$ )
6:    $(dk_u, rsk_u) \leftarrow isk_u$ 
7:    $D_{kem}, D_{\rho_\perp} := \emptyset \triangleright$  Initialize empty lists
8:   for  $t \in [L] \cup \{\perp\}$  do
9:      $(\widehat{ek}_{u,t}, \widehat{dk}_{u,t}) \xleftarrow{\$} \text{KEM.KeyGen}(1^\lambda)$ 
10:     $\triangleright$  Create and sign Merkle tree
11:     $(root_u, tree_u) \leftarrow \text{MerkleTree}(\{\widehat{ek}_{u,t}\}_{t \in [L] \cup \{\perp\}})$ 
12:     $\sigma_{u,root} \xleftarrow{\$} \text{RS.Sign}(rsk_u, root_u, \{rvk_u\})$ 
13:     $(rvk, \_) \xleftarrow{\$} \text{RS.KeyGen}(1^\lambda) \triangleright$  Discard rsk
14:    for  $t \in [L]$  do  $\triangleright$  One-time prekey bundles
15:       $path_{u,t} \leftarrow \text{getMerklePath}(tree_u, t)$ 
16:       $prek_{u,t} := (\widehat{ek}_{u,t}, path_{u,t}, root_u, \sigma_{u,root}, rvk)$ 
17:       $D_{kem}[t] \leftarrow (prek_{u,t}, \widehat{dk}_{u,t})$ 
18:     $\triangleright$  Last-resort prekey bundle  $t = \perp$ 
19:     $path_{u,\perp} \leftarrow \text{getMerklePath}(tree_u, L + 1)$ 
20:     $prek_{u,\perp} := (\widehat{ek}_{u,\perp}, path_{u,\perp}, root_u, \sigma_{u,root}, rvk)$ 
21:     $D_{kem}[t] \leftarrow (prek_{u,\perp}, \widehat{dk}_{u,\perp})$ 
22:   return  $(\vec{prek}_u := (prek_{u,t})_{t \in [L] \cup \{\perp\}},$ 
    $st_u := (D_{kem}, rvk, D_{\rho_\perp}))$ 

```

Algorithm 4: RingXKEM’s identity key and prekey bundle generation algorithms.

order of magnitude larger than in the classical setting, making prekey bundles very large. The Merkle tree optimization allows to only upload a single signature: users accumulate all the KEM public keys $(\widehat{ek}_t)_{t \in [L] \cup \{\perp\}}$ and only sign the digest root. Note that this Merkle tree optimization is made possible owing to our new definition of BAKE protocols. Previous works on Signal’s handshake protocols, e.g., [Bre+22; Coh+17; Coh+20; Col+24; FG24; Has+21; Has+22], are not able to handle such optimization as each prekey bundle $prek_t$ was assumed to be generated *independently*.

One downside of our optimization is that prekey bundles become slightly larger. In particular, a sender is now required to download an extra Merkle tree $path_t$ proving that \widehat{ek}_t was accumulated in root. Notice that in our construction, the users explicitly include $path_t$ in each prekey bundle $prek_t$. However, in practice, we can simply let the server reconstruct them using the uploaded $(\widehat{ek}_t)_{t \in [L] \cup \{\perp\}}$ without harming security. Namely, when a sender retrieves u ’s prekey bundle from the server, the server can compute $path_t$ on the fly. Importantly, due to binding of the Merkle tree, the server cannot inject a prekey that u did not accumulate in the hash digest.

Lastly, we note that the usage of ring signatures and an SKE to encrypt the ring signature is purely for deniability reasons, similarly to what is done in the standard AKE protocol by Hashimoto et al.

6 Deniability of BAKE

Deniability is a privacy property ensuring that the transcript of a communication session cannot serve as evidence that a user participated in said communication, even if another party attempts to frame them. This is particularly relevant in scenarios involving, e.g., oppressive regimes or whistleblowers, where participation alone can be incriminating. For Signal, deniability is integral to their protocol’s design [KS23; MP16]. Selecting the most suitable protocols thus requires not only considering key indistinguishability and authentication guarantees, but also placing equal emphasis on deniability.

In this section, we formally capture the deniability of BAKE protocols, building upon prior work on the deniability of AKE [CF11; Dag+13; DGK06; UG15; UG18], and adaptations made for specific handshake protocols [Bre+22; Col+24; FJ24; Has+22; Vat+20].

6.1 Entities and Roles

Following the general approach common to existing simulation-based definitions, we define an *accuser*, who collects evidence which is provided to a *distinguisher*⁸, who, in turn, decides if the evidence could have been simulated by the accuser. Precisely, the entities involved in deniability are as follows.

Accused users: The set of honest users, denoted as \mathcal{H} , whose goal is to deny their involvement in the BAKE protocol. An accused user can either be a sender or a receiver.

⁸ Prior work has also used the term “Judge”; we prefer “distinguisher”, which better reflects its algorithmic (as opposed to human) nature.

```

1: function RingXKEM.Send(isks, ikr, prekr)
2:   (dks, rsks) ← isks; (ekr, rvkr) ← ikr
3:   (ekr, pathr, rootr, σr,root, rvk) ← prekr
4:   require [ReconstructRoot(ekr, pathr) = rootr]
5:   require [RS.Verify({rvkr}, ekr, σr,root) = 1]
6:   (ssr, ctr)  $\xleftarrow{\$}$  KEM.Encaps(ekr)
7:   (s̄sr, c̄tr) ← KEM.Encaps(ekr)
8:   content := iks || ikr || prekr || ctr || c̄tr
9:   K || Kske := KDF(ssr || s̄sr, content)
10:  σ ← RS.Sign(rsks, content, {rvks, rvk})
11:  ctske ← SKE.Enc(Kske, σ) ▷ Mask ring signature
12:  ρ := (ctr, c̄tr, ctske)
13:  return (K, ρ)

```

Algorithm 5: RingXKEM’s sender algorithm. The prekey bundle index t is oblivious to the sender.

```

1: function RingXKEM.Receive(iskr, str, iks, t, ρ)
2:   (dkr, rskr) ← iskr; (eks, rvks) ← iks
3:   (Dkem, rvk, Dρ⊥) ← str
4:   (ctr, c̄tr, ctske) ← ρ
5:   ▷ Check  $t^{\text{th}}$  prekey bundle was not deleted. ◁
6:   require [Dkem[t] ≠ ⊥]
7:   if [t = ⊥] then
8:     require [(ctr, c̄tr) ∉ Dρ⊥] ▷ Check not re-
9:     played.
10:    Dρ⊥ ← Dρ⊥ ∪ {(ctr, c̄tr)}
11:    (prekr,t, dkr,t) ← Dkem[t]
12:    ssr := KEM.Decaps(dkr,t, ctr)
13:    s̄sr := KEM.Decaps(dkr,t, c̄tr)
14:    content := iks || ikr || prekr,t || ctr || c̄tr
15:    K || Kske := KDF(ssr || s̄sr, content)
16:    σ := SKE.Dec(Kske, ctske) ▷ Unmask signature
17:    require [RS.Verify({rvks, rvk}, content, σ) =
18:    1]
19:    if [t ≠ ⊥] then
20:      Dkem[t] ← ⊥ ▷ Delete prekey bundle
21:      str ← (Dkem, rvk, Dρ⊥)
22:    return (K, str)

```

Algorithm 6: RingXKEM’s receiver algorithm.

(Insider) accusers: The set of corrupted users, denoted as C , that communicate⁹ with an accused user. Their goal is to prove that the accused user ran a BAKE protocol with them. We consider *honest-but-curious* accusers and *malicious* accusers. The former honestly follow the protocol description but may collect as much information as possible to accuse their peer. In contrast, the latter considers much stronger accusers that can execute arbitrary code; capturing, e.g., devices running a modified secure messaging application.

(Outsider) accuser: An adversary aiming to prove that a pair of honest users communicated. This could be, e.g., the server or another user of the secure messaging application.

Distinguisher: An entity outputting a *verdict* on whether a user participated in a BAKE protocol.

6.2 Distinguisher Capabilities

The distinguisher D determines whether an accused user participated in a BAKE protocol based on a transcript (i.e., prekey bundles and handshake message) and the session key K . It is important that K also be deniable since it is used to exchange the actual payload of the secure messaging protocol [DGK06; Vat+20]. To decide, D may further be provided information through so-called *leakage* and *disclosure* functions. The former dictates how much information of the accused user leaks to D , whereas the latter dictates how much information the accusing user discloses to D . A BAKE protocol is more deniable if it allows leaking and disclosing more information to the distinguisher.

Leakage function for accused users. The information leaked from an accused user is formalized via a function $\mathcal{L}_{\text{leak}}$. We consider three levels of leakage: for $\text{leak} = \text{low}$, no leakage occurs; $\text{leak} = \text{med}$ leaks the identity secret key; and $\text{leak} = \text{high}$ leaks all the accused user’s secret information.

Definition 6.1. The leakage function $\mathcal{L}_{\text{leak}}$ for the set \mathcal{H} of accused users of a BAKE protocol is defined as:

$$\mathcal{L}_{\text{leak}}((\text{isk}_u, \text{st}_u)_{u \in \mathcal{H}}) := \begin{cases} (\perp, \perp) & \text{if } \text{leak} = \text{low} \\ ((\text{isk}_u)_{u \in \mathcal{H}}, \perp) & \text{if } \text{leak} = \text{med} \\ ((\text{isk}_u, \text{st}_u)_{u \in \mathcal{H}}) & \text{if } \text{leak} = \text{high} \end{cases}$$

⁹ Throughout the paper, for readability, we may say users u and v *communicated* with each other to mean that u and v participated in a BAKE protocol.

Disclosure function for honest-but-curious accusers. The information disclosed by honest-but-curious (insider) accusers is formalized using a function $\mathcal{D}_{\text{disc}}$. We again consider three levels of disclosure: $\text{disc} = \text{low}$ is the weakest setting where the accuser only discloses its identity secret key; $\text{disc} = \text{med}$ additionally discloses its *current* state; lastly, $\text{disc} = \text{high}$ further discloses the *initial* state output by algorithm `BAKE.PreKeyBundleGen`. The third setting models an honest-but-curious accuser that follows the protocol description, but may store information without deleting it.¹⁰

Definition 6.2. The disclosure function $\mathcal{D}_{\text{disc}}$ for the set C of honest-but-curious (insider) accusers of a BAKE protocol is defined as:

$$\mathcal{D}_{\text{disc}}\left(\left(\text{isk}_u, \text{st}_u, \text{st}_u^{\text{init}}\right)_{u \in C}\right) := \begin{cases} \left(\left(\text{isk}_u\right)_{u \in C}, \perp, \perp\right) & \text{if } \text{disc} = \text{low} \\ \left(\left(\text{isk}_u, \text{st}_u\right)_{u \in C}, \perp\right) & \text{if } \text{disc} = \text{med} \\ \left(\left(\text{isk}_u, \text{st}_u, \text{st}_u^{\text{init}}\right)_{u \in C}\right) & \text{if } \text{disc} = \text{high}. \end{cases}$$

Disclosure function for malicious accusers. Our model accounts for varying adversarial capabilities, distinguishing between honest-but-curious accusers (*standard* deniability) and malicious accusers (*strong* deniability), as did [Has+22]. Note that malicious accusers can deviate arbitrarily from the protocol, such as by maliciously generating prekeys, possibly without knowing the associated secrets. We thus assume the malicious accuser to always disclose their entire state $\text{st}_{\mathcal{A}}$. Note that despite the general unpredictability of malicious accusers, we require that $\text{st}_{\mathcal{A}}$ includes their identity secret keys, based on the assumption that they register a valid identity public key (cf. Section 6.4).

6.3 Scopes of Deniability

Lastly, we consider two scopes of deniability: one focused on protecting individual users and the other on shielding the conversation between two users as a whole.

Local deniability: Allows an accused user, either a sender or a receiver, to deny participating in a BAKE protocol.

Local deniability considers only insider accusers

Global deniability: Further allows a pair of communicating accused users to *simultaneously* deny participating in a BAKE protocol. This implicitly accounts for outsider accusers.

Formalizing what it means to “deny participating” is one of our main technical contributions, which we explain in Section 6.5. At a high level, in a locally deniable protocol, a distinguisher may be convinced that either one of the users participated in the BAKE protocol, but cannot determine which one. Local deniability thus suffices when the accused user seeks only individual protection. In contrast, when both communicating users simultaneously seek deniability (e.g., a journalist communicating with a source) global deniability is required. In such cases, the distinguisher cannot exclude the possibility that the communication was generated by an unrelated third party.

6.4 Modeling Choices and Simplifications

Firstly, similarly to other works on the deniability of Signal handshake protocols [Bre+22; Jia+22; Vat+20], we assume that users *honestly* generate their identity keys. Secondly, we assume that honest-but-curious accusers disclose *at least* as much as the information leaked from accused users, e.g., if the accused leak updated states, then so do the accusers. Finally, we focus on the deniability of *messages*. This means that like prior work (e.g., [Bre+22; Col+24]), we do not consider the deniability of registering or uploading prekey bundles. Our focus is the deniability of the handshake message and the computation of the resulting session key by both sender and receiver, even if there is evidence of registration.

6.5 Deniability of Bundled AKE Protocols

We define the *deniability* of a BAKE protocol. The main novelty in our definitions is a new pragmatic metric for deniability, which may be of an independent interest. Looking ahead, our definition allows constructing a post-quantum BAKE protocol using a new efficient “deniable” ring signature based on the NIST-standardized Falcon.

¹⁰ Considering initial state leakage for the accused user would imply that their device was tampered with. We do not consider such settings as deniability can be trivially lost through other means.

Overview of Our Deniability Definition Say an accused user wants to deny participating in a certain protocol, or, more formally, the user wants to prove that any *evidence* the distinguisher holds could have come from somebody else. In the context of a BAKE protocol, evidence is the protocol transcript (prekey bundles and handshake message) and the session key, cf. [Section 6.2](#). A standard way to formalize this is to construct a *simulator* that can output simulated evidence indistinguishable from real evidence to a distinguisher [[DNS98](#)]. This has been used to define deniable AKEs, e.g., [[DGK06](#)]. This simulator must not only exist but also be constructible in the real world [[Pas03](#)]. An accused user then convinces the distinguisher by showing that such a simulator could have been actually used by the accuser.

We also use simulators to define deniability of a BAKE protocol, but with a twist, inspired by concepts in differential privacy and differential indistinguishability [[Bac+15](#); [Dwo+06](#); [Mir+09](#)]. Prior works required the real evidence π_{real} and simulated evidence π_{sim} to be *indistinguishable* by a distinguisher D . That is, they (informally) required the statistical distance to be close: $|\Pr[D(\pi_{\text{real}}) = 0] - \Pr[D(\pi_{\text{sim}}) = 0]| = \delta(\lambda)$ for some negligible function δ . While sufficient, we observe this level of deniability to be overly conservative. In practice, the accused user only needs to prove that a simulator *could have* generated the evidence, not that the simulator outputs evidence with the same probability as the accused user. As a concrete example, assume the evidence includes a ring signature σ where the ring consists of two users: the accused u and the accuser v . Roughly, prior definitions require that the probability of u and v outputting σ is identical. We relax this so that there could be a higher chance that u outputs σ , as long as v could have output σ . Put differently, while σ is more likely to have come from u , we cannot *deny* the possibility that it came from v . This is sufficient for u to plausibly deny the conversation.

In order to formalize this idea, we introduce a *multiplicative slack* $\mu(\lambda)$, and only require a relaxed condition: $\Pr[D(\pi_{\text{real}}) = 0] \leq \mu(\lambda) \cdot \Pr[D(\pi_{\text{sim}}) = 0] + \delta(\lambda)$. Technically, this means the two distributions are close in terms of the *hockey-stick divergence* [[SV16](#)]. While the original definition is recovered by setting $\mu(\lambda) = 1 + \text{negl}(\lambda)$, we obtain a relaxed definition by setting, say $\mu(\lambda) = 1 + 0.1$. This indicates that while the evidence is (roughly) 10% more likely to have come from the accused user, we cannot ignore the high possibility that the accuser outputs it by running the simulator.¹¹

The benefit of relaxing the definition becomes clear when we later construct a post-quantum BAKE protocol based on ring signatures. We notice that while a ring signature based on the same parameter sets as Falcon [[Pre+22](#)] is insufficient for the standard notion of deniability, it is sufficient for the relaxed definition with a multiplicative slack $\mu(\lambda) = 1 + 2^{-27}$. See [Sections 8](#) and [9.1](#) for details.

Note that in our formal definitions, we allow the multiplicative slack μ to depend on the number of queries D makes. This allows for tighter analysis using a Falcon-based ring signature as Falcon also assumes an upper-bound on the number of signatures (i.e., $Q = 2^{64}$).

Deniability Against Honest-but-Curious Accusers We first give an overview of local and global deniability against honest-but-curious accusers,¹² formally defined through a game in [Alg. 9](#) of [Appendix A](#). The distinguisher D is given the set of identity keys and prekey bundles, and can adaptively query transcripts exchanged between users. At the end of the game, D is given the leakage and disclosed information of the accused and accusing users. It then outputs some bit b (e.g., a verdict of the judge). As discussed above, we require that the probability D outputs b in the real world (i.e., $\text{mode} = \text{real}$) is overwhelmingly likely to be within some multiplicative slack μ of the probability D outputs b in the simulated world (i.e., $\text{mode} = \text{sim}$). Importantly, our definition captures the deniability of last-resort prekey bundles as well.

The simulated world is defined using a simulator algorithm, which first simulates the prekey bundles of all users. Notably, we allow simulation of the *honest* users' prekey bundles as these are not necessarily tied to the identity key, which we assume to be honestly registered (see [Section 6.4](#)). As an example, assume a prekey bundle consisting of two public keys where one key is signed using the identity key, while the other is not. Generating the latter key can then be plausibly denied, as it could have been injected by an accuser. Then, the simulator must simulate the honest user's transcripts and session keys using only the accuser's secret information. In the local setting, the simulator is given the secrets of the insider accuser taking part in the communication. Conversely, for global deniability, the simulator has no access to either sender or receiver secrets, since the accuser is an outsider.

¹¹ Note that we do not require the other direction: $\Pr[D(\pi_{\text{sim}}) = 0] \leq \mu'(\lambda) \cdot \Pr[D(\pi_{\text{real}}) = 0] + \delta'(\lambda)$. We only care that if some verdict was made using a real evidence, then the same verdict could have been made on a simulated evidence.

¹² As our definition implicitly captures outsider accusers, accusers will mean insider accusers in this section unless otherwise stated (cf. [Section 6.1](#)).

Lastly, we must simulate the honest and corrupt user’s state, which may be leaked and disclosed to the distinguisher, respectively.¹³ As the state is only used by the `BAKE.Receive` algorithm, we only need to simulate the state update of honest *receivers* r . Importantly, the simulator algorithm performing this update should have no information on the sender s . This is because if receiver r ’s updated state depended on the sender s , the state may prove to the distinguisher that r was communicating with s . On the other hand, we allow it to take the identity secret key isk_r as input since, by definition of the `BAKE.PreKeyBundleGen` algorithm, the state can depend on isk_r . The formal definition of local and global deniability is given in [Appendix A](#).

Remark 6.1. For simplicity, our model does not explicitly capture asymmetric deniability, but can be extended to do so. For instance, deniability for accused users who are *always receivers* can be modeled by restricting the distinguisher to query transcripts exchanged between users where the receiver r must be honest. This scenario applies when r never initiates conversations with unknown users. For some protocols discussed in this work, we demonstrate stronger deniability guarantees for accused receivers.

Deniability Against Malicious Accusers In [Section 3.3](#) of [\[Kat+25\]](#), we further define *strong* local and global deniability, that is deniability against malicious accusers.

7 Deniability of X3DH and PQXDH

In this section, we summarize the levels of deniability satisfied by X3DH and PQXDH. [Table 1](#) provides an overview of the deniability of X3DH, PQXDH and RingXKEM.

Table 1: Signal key exchange protocols and their deniability and security properties

Signal handshake protocol deniability properties										Legend			
Protocol:	X3DH		PQXDH		PQXDH			RingXKEM		Last-resort prekey:			
	Classic \mathcal{A}/D		Classic \mathcal{A}/D		Classic \mathcal{A}		Classic or Quantum D		Classic or Quantum \mathcal{A}/D		Icon	No	Yes
Deniability Level	Leakage leak	disc	Leakage leak	disc	Leakage leak	disc	QROM	Leakage leak	disc	QROM		leak or disc	leak or disc
local	●	●	●	●	●	●	✓	●	●	✓	●	high	high
global	●	●	●	●	●	●	✓	●	●	✓	●	high	med
strong-	local	●†	●	●†	●	● ^{SO}	?	● ^{SO}	●	?	?	Open problem	
	global	●†	●	●†	●	● ^{SO}	?	● ^{SO}	●	?	● ^{SO}	Accusers \mathcal{A} restricted to being senders, no deniability otherwise.	
Security [HKW25]	Classical		Harvest-Now Decrypt-Later				Fully post-quantum				†	Proof using GGM.	

Example: RingXKEM is local deniable with leak = high and disc = high even if a last-resort prekey bundle was used.

Remark: For strong deniability, we always set disc = high, since we have no control over the information revealed by malicious accusers.

Local deniability. Both protocols achieve the highest level of local deniability (i.e., (leak, disc) = (high, high)). For X3DH, this matches our intuition since the sender and receiver hold a symmetric role in the generation of the session key K . This is also the case for PQXDH: the honest sender remains deniable since `KEM.Encaps` can be run publicly; the honest receiver remains deniable since, assuming an honest-but-curious accusing sender, (informally) the accuser knows the outcome of `KEM.Decaps`. Importantly, we show that local deniability of PQXDH holds even if the accuser is classical but the distinguisher is *quantum* (i.e., HNJL security).

¹³ Put differently, we can ignore these simulators if leak \in { low, med } or disc = low (cf. [Section 6.2](#)).

Global deniability. Global deniability is more nuanced. If the accused users never deplete their one-time prekey bundles, both protocols achieve the highest level of global deniability (i.e., $(\text{leak}, \text{disc}) = (\text{high}, \text{high})$). However, if the accused user used their last-resort prekey bundle, it can only support global deniability with $(\text{leak}, \text{disc}) = (\text{med}, \text{high})$, i.e., the accused user *cannot* deny if its user state st_u leaks to the distinguisher.

At a high level, to argue X3DH is globally deniable, the session key and handshake message (K, ρ) must be simulatable without relying on any secret of the sender and receiver. In case the sender’s one-time prekey bundle is used, this follows from K being KDF-derived from a DH agreement between a one-time prekey opk_r and an ephemeral key epk (cf. [Alg. 2, Line 13](#)). Namely, since the secrets of opk_r and epk are deleted from the receiver and sender states respectively, a distinguisher cannot distinguish between a correctly generated K and a randomly sampled K by an outside accuser, assuming the hardness of DDH. In contrast, when a last-resort prekey bundle is used, key K can be derived by a distinguisher holding both the receiver’s isk_r and st_r , which includes spksec_r (see [Alg. 3, Lines 12 and 13](#)). Hence, accused users can only leak their identity secret key (i.e., $\text{leak} = \text{med}$).

Global deniability of PQXDH is shown quite differently and relies on the IND-CPA security of the KEM. This is because against a *quantum* distinguisher, the argument used above fails; DDH is no longer hard.¹⁴ We show deniability by the session key K being KDF-derived from KEM key ss_{KEM} (cf. [Alg. 2, Line 15](#)). As long as the distinguisher cannot decrypt the ciphertext ct included in ρ , K remains indistinguishable.

Strong local and global deniability. Unlike previous works relying on knowledge type assumptions, we rely on the generic group model (GGM) [[Sho97](#)] to show strong (local and global) deniability. While GGM is an idealized model, it allows to concretely write down a simulator assuming a *generic* accuser, aligning better with the notion of deniability. Indeed, GGM has been used by Signal’s private group system [[CPZ20](#)], giving us more confidence in generic accusers.

We can straightforwardly prove strong local and global deniability of X3DH in the GGM with the same level of information leakage and disclosure considered by non-strong deniability. One limitation, however, is that GGM assumes a prime-order group, whereas X3DH uses the non-prime-order X25519. We can get around this by either relying on a prime-order group such as Ristretto, used by Signal’s private group system [[CPZ20](#)], or extending GGM to work over non-prime groups. We leave these considerations to future work.

Similarly, we show strong local and global deniability of PQXDH against a classical accuser and a *classical* distinguisher. For strong (local and global) HNJK deniability of PQXDH, we are only able to prove deniability for the *receiver* in the *classical* ROM; see [Section 4.2 of \[Kat+25\]](#) for a discussion on why the deniability of the *sender* breaks.

Remark 7.1. In our BAKE model, all elements of a pre-key bundle run out simultaneously. However, in the actual specification of PQXDH, last resort KEM keys may be used before one runs out of one time prekeys $\text{opk}_{r,t}$ and vice versa. As we detail in [Appendix E.6 of \[Kat+25\]](#), this does not harm our deniability results for PQXDH.

8 Deniability of RingXKEM

In this section, we see how the RSs in RingXKEM simultaneously ensure authentication and deniability.

8.1 Deniable Ring Signatures

We first recall the syntax of ring signatures. Standard notions of correctness and unforgeability are provided in [Appendix A.4 of \[Kat+25\]](#).

Definition 8.1 (Ring Signatures). A ring signature (RS) scheme consists of three PPT algorithms:

$\text{RS.KeyGen}(1^\lambda) \xrightarrow{\$} (\text{rvk}, \text{rsk})$: On input the security parameter 1^λ , it outputs a pair of keys (rvk, rsk) .

$\text{RS.Sign}(\text{rsk}, M, \text{RL}) \xrightarrow{\$} \text{sig}$: On input a secret key rsk , a message M , and a list of public keys equipped with some canonical ordering, i.e., a ring, $\text{RL} = \{\text{rvk}_1, \dots, \text{rvk}_N\}$, it outputs a signature sig .

$\text{RS.Verify}(\text{RL}, M, \text{sig}) \rightarrow 1/0$: On input a ring $\text{RL} = \{\text{rvk}_1, \dots, \text{rvk}_N\}$, a message M , and a signature sig , it outputs 1 if the signature is valid and 0 otherwise.

¹⁴ Against classical D, global deniability of PQXDH follows from X3DH.

```

1: function GameRS,b,ℳDeny(1λ, Q)
2:   for user  $u \in [N]$  do (rvku, rsku)  $\xleftarrow{\$}$  RS.KeyGen(1λ)
3:    $q := 0 \triangleright$  Number of queries made
4:   return  $b \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\text{RS,Sign}}(\cdot, \cdot, \cdot)}((\text{rvk}_u, \text{rsk}_u)_{u \in [N]})$ 
5:   function  $\mathcal{O}_{\text{RS,Sign}}(M, u_0, u_1)$ 
6:     require  $[(u_0, u_1) \in [N] \times [N]] \wedge [q < Q]$ 
7:      $q \leftarrow q + 1$ 
8:     return  $\sigma \xleftarrow{\$} \text{RS.Sign}(\text{rsk}_{u_b}, M, \{\text{rvk}_{u_0}, \text{rvk}_{u_1}\})$ 

```

Algorithm 7: Game for the deniability of RS.

Deniability: weakening anonymity. The standard notion of *anonymity* guarantees that signatures produced using two secret keys are indistinguishable. While sufficient, we observe that, thanks to our new metric for measuring the deniability of BAKEs in terms of the hockey-stick divergence, we can relax anonymity (cf. Section 6.5). Informally, we only care that signatures remain *deniable*; the signatures do not provide hard evidence about which secret key was used to sign a message.

Definition 8.2 (Deniability). Let $\mu : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}^+$ be a positive-valued function and $Q = \text{poly}(\lambda)$ an upper bound on the number of signing queries. A ring signature scheme is (μ, δ) -deniable if for any $N = \text{poly}(\lambda)$, and efficient adversary \mathcal{A} ,

$$\Pr \left[\text{Game}_{\text{RS},0,\mathcal{A}}^{\text{Deny}}(1^\lambda, Q) = 0 \right] \leq \mu(\lambda, Q) \cdot \Pr \left[\text{Game}_{\text{RS},1,\mathcal{A}}^{\text{Deny}}(1^\lambda, Q) = 0 \right] + \delta,$$

with $\delta = \text{negl}(\lambda)$, and where $\text{Game}_{\text{RS},b,\mathcal{A}}^{\text{Deny}}$ is defined in Alg. 7.

As alluded to in Section 6.5, the main benefit of our new definition is that it provides justification to formally use “reasonably” anonymous RSs. Indeed, we show that an RS based on the NIST-standardized Falcon [Pre+22] is deniable with a multiplicative slack $\mu = 1 + 2^{-27}$; using the standard notion of anonymity, this translates to a mere 27-bits of security. This indicates that deniability is a more fine-grained notion than anonymity, allowing to split the distinguishing probability δ_{anon} of anonymity into μ and δ of deniability. What we uncover is that if δ_{anon} can be “absorbed” into μ , we can maintain a negligibly small δ , sufficient for deniability applications. We believe our new definition to be of an independent interest.

8.2 Deniability of RingXKEM

We summarize the level of deniability that RingXKEM satisfies. See Table 1 for a complete overview. The formal statements are provided in Appendix G of [Kat+25].

Local and global deniability. RingXKEM achieves the highest level of local deniability. This matches our intuition since, due to the deniability of RS, the signature included in the senders’ handshake message, which authenticates the sender, does not reveal *which* key (among rvk_s and $\widehat{\text{rvk}}_r$) was used to sign, so that either the sender or the receiver could have produced it.

We further have global deniability, thanks to the RS key $\widehat{\text{rvk}}$ in the prekey bundle not being authenticated. Specifically, the simulator, given prekey bundles of honest users, can substitute the verification key $\widehat{\text{rvk}}$ with one for which it knows the associated secret key. It can then easily compute the required ring signature and successfully simulate the communication of two honest users. However, if the receiver’s state — which contains the (original) receiver verification key — leaks, the simulation can easily be detected by the distinguisher. Hence, global deniability only holds if the accused receiver’s state does not leak. Both local and global deniability of RingXKEM hold even if both the accuser and the distinguisher are *quantum*, so long as RS’s deniability holds against quantum adversaries.

Strong local and global deniability. The situation for strong (local and global) deniability is less clear. Firstly, we are only able to prove deniability for the *receiver* (cf. Remark 6.1); see Appendix G.3.1 of [Kat+25] for a discussion on why the deniability of the *sender* breaks. In fact, we can only prove strong deniability for the receiver in the *classical* ROM, while still enabling quantum capability to the malicious accuser and distinguisher. In the classical ROM, the proof is quite natural using the observation that the simulator need only simulate when the (honest) receiver accepts, which we

expand on in Section 3.3 of [Kat+25]. Indeed, for the receiver to accept, the KDF, modeled as a random oracle, must have output keys $K \| K_{\text{ske}}$ for which the ciphertext ct_{ske} in the (possibly maliciously generated) handshake message ρ decrypts correctly, yielding a valid signature of content for the ring $\{\text{rvk}_s, \widehat{\text{rvk}}_r\}$. If such keys exist, the simulator simply outputs K .

Unfortunately, it is unclear how to extend our proof to work in the *quantum* ROM. Our proof hinges on the simulator observing the input/output of the random oracle, however, in the QROM, such measurement may affect the malicious accuser’s quantum state, leading to a different behavior from the real world. We leave it as an open problem to prove fully post-quantum receiver strong deniability.

9 Ring Signatures from Falcon and MAYO

In this section, we construct two novel 2-user ring signatures achieving our deniability notion from Section 8.1. The goal is to design optimized 2-ring signatures based on NIST standards, for an increased potential of adoption. We identified two strong candidate signature schemes: the standardized signature Falcon [Pre+22], and the additional signature candidate MAYO [Beu+24]. Both of these schemes have short signatures, and thus great potential for compact ring signatures.

9.1 Falcon-Based Ring Signature

This section provides a high-level construction for our Falcon-based ring signature. Formal definitions are given in Appendix I.1 of [Kat+25].

Falcon is a hash-and-sign signature scheme standardized by NIST. It is built over NTRU lattices, that is lattices generated using a uniform-looking polynomial $h = g \cdot f^{-1}$ where f, g are short polynomials in the ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$. Key generation calls an NTRU trapdoor sampler $\text{TpdGen}()$ to obtain a public polynomial h , and a trapdoor basis \mathbf{B} for the corresponding NTRU lattice. Then, to sign a message M , one samples a salt salt , and computes a target $c = H(M, \text{salt})$ from a hash function. The trapdoor then allows for sampling a short preimage $(u, v) = \text{PreSmp}(\mathbf{B}, \sigma, -c)$ in the NTRU lattice of c , i.e. such that $h \cdot u + v = c$. The signature is (salt, u) . Verification first recovers $v = c - h \cdot u$ from $c = H(M, \text{salt})$, and verifies the shortness of (u, v) , i.e. that $\|(u, v)\| \leq \beta$.

Ring Signature Construction. We build a ring signature FalconRS from Falcon and overcome limitations of previous works [GJK24b; LAZ19] as it seems impossible to build an RS from the Falcon parameters under standard anonymity. We instead aim for our relaxed deniability notion.

Our RS scheme samples a public polynomial $h_i \in \mathcal{R}_q$ and trapdoor \mathbf{B}_i for each ring user $i \in [N]$. Signing is modified to find a preimage of c for an aggregation of the public keys: we sample $v, (u_i)_{i \in [N]}$ such that $c = v + \sum_i h_i \cdot u_i$ knowing only the trapdoor for a single h_i . This aggregation is inspired by ring trapdoor functions [BK10], although lattices allow further optimization: we can reuse coordinate v for all signers and omit it in the final signature since it can be recovered from c and the u_i . This was previously proposed in Gandalf [GJK24b].

Concretely, when party i signs, it first samples the contribution for other parties as discrete Gaussians of parameter σ , i.e., $u_j \stackrel{\$}{\leftarrow} \chi_u$ for $j \neq i$ (χ_u is tailcut to $[-\eta', \eta']$ for implementation purposes). It then samples $(u_i, v) \stackrel{\$}{\leftarrow} \text{PreSmp}(\mathbf{B}_i, \sigma, -c')$, where $c' = H(M, \text{salt}) - \sum_{j \neq i} u_j \cdot h_j$, to complete the signature $\text{sig} = (u_i)_{i \in [N]}$. Leveraging the fact (u_i, v) appears as Gaussian distributed when c' is uniform, we can show that the signature distribution is roughly independent of the signer, ensuring deniability. To cover the larger number of elements in the ring signature, we introduce a new verification bound β_{sig} . The construction is formalized in Alg. 8.

Security Analysis and Parameters. FalconRS uses the same base parameters as Falcon-512. The ring verification bound β_{sig} is set to $1.1 \cdot \sqrt{3n}\sigma$ to bound the norm of two user contributions.

Unforgeability. FalconRS can be proven unforgeable in an analogous manner to Falcon [GJK24a], reducing to the same NTRU and RSIS assumptions, though the SIS bound is increased by a factor $\sqrt{k+1}/\sqrt{2}$, and its preimage space is of dimension $k+1$ for rings of k users. This leads to a core-SVP security of 111 bits; a reasonable degradation over the 120 bits of Falcon.

```

1: function FalconRS.KeyGen( $1^\lambda$ )
2:    $h, \mathbf{B} \xleftarrow{\$} \text{TpdGen}() \triangleright$  Sample lattice generator  $h$ , and trapdoor
3:   return ( $\text{rvk} := h, \text{rsk} := \mathbf{B}$ )
4: function FalconRS.Sign( $\text{rsk}_i, M, \text{RL} := \{\text{rvk}_j\}_j$ )
5:    $\mathbf{B} := \text{rsk}_i; \{h_j\}_j := \{\text{rvk}_j\}_j$ 
6:    $\text{salt} \xleftarrow{\$} \{0, 1\}^k; c := H(\text{salt}, \text{RL}, M) \in \mathcal{R}_q$ 
7:   for  $j \neq i$  do  $u_j \leftarrow \chi_u \triangleright \chi_u$  is  $\mathcal{D}_{\mathbb{Z}^n, \sigma, 0}$  tailcut to  $[-\eta', \eta']$ 
8:    $c' := c - \sum_{j \neq i} h_j \cdot u_j$ 
9:    $(u_i, v) := \text{PreSmp}(\mathbf{B}, \sigma, -c') \triangleright$  Pre-image sampling
10:  if  $\|((u_j)_j, v + c')\| > \beta_{\text{sig}}$  then restart  $\triangleright$  Too large
11:  return  $\text{sig} := (\text{salt}, \{u_j\}_j)$ 
12: function FalconRS.Verify( $\text{RL} := \{\text{rvk}_j\}_j, M, \text{sig}$ )
13:   $(\text{salt}, \{u_i\}_i) := \text{sig}; \{h_i\}_i := \{\text{rvk}_i\}_i$ 
14:   $c := H(\text{salt}, \text{RL}, M) \in \mathcal{R}_q$ 
15:   $v := c - \sum_i h_i \cdot u_i \triangleright$  Compute  $v$  such that  $c = v + \sum_i h_i \cdot u_i$ 
16:  return  $\|((u_i)_i, v)\| \leq \beta_{\text{sig}}$   $\triangleright$  Verify pre-image shortness

```

Algorithm 8: Falcon-based ring signature scheme

Deniability. Proving anonymity appears unfeasible for an RS based on the Falcon parameters, as the distribution of signatures is at a non-negligible distance from the ideal one. Instead, we show that FalconRS is deniable. We defer the formal statement to Appendix I.1.2 of [Kat+25], and provide the intuition here.

The real signature distribution differs from the ideal one for two reasons: (i) the convolution of discrete Gaussians only approximates a discrete Gaussian with larger parameters, and (ii) the use of approximations in internal computations (tail cuts, floating points, and polynomial approximations). The use of tail cuts translates to a statistical distance characterized by the deniability term δ . Falcon cuts tails with probability roughly 2^{-70} , and we similarly select the tailcut parameter η' of χ_u to ensure a negligible tailcut probability of 2^{-70} (i.e. $\eta' := \lceil \sqrt{70} \cdot \log(2) \cdot \sqrt{2} \cdot \sigma \rceil = 1633$). This guarantees a small term δ . Interestingly, the other approximations introduce a relative error in the signature distribution, so that the probability of sampling a given signature is multiplied by a factor close to 1. These are absorbed by the multiplicative slack μ , which exactly captures such factors between probabilities.

We formalize and evaluate the errors introduced by each approximation in Appendix I.1.3 of [Kat+25], obtaining that FalconRS is (μ, δ) -deniable for $\mu = 1 + 2^{-27}$ and $\delta = 2^{-57}$. We note that our analysis readily applies to Gandalf [GJK24b], which, at a high level, simply chooses a different trapdoor generator and preimage sampler. While Gandalf initially claimed an anonymity of 2^{-70} , an updated version acknowledged a proof flaw and an anonymity of only 2^{-30} . Alternatively, Gandalf can be proven *deniable*, with roughly the same μ, δ as FalconRS.

9.2 MAYO-based Ring Signature

We provide a second RS construction based on MAYO, a candidate in NIST's Call for Additional Signature Schemes. Viewing MAYO as a hash-then-sign signature scheme, we can apply generic transformations from [AOS02] to obtain an efficient RS. We analyze parameter sets proposed for standardization and provide alternative ones achieving higher deniability.

MAYO is designed over quadratic maps. At a high level, it chooses a map $\mathcal{P} : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$ with trapdoor tp , as public and secret keys respectively, from which is derived a larger map $\mathcal{P}^* : \mathbb{F}_q^{kn} \rightarrow \mathbb{F}_q^m$. To sign message M , one computes a target $\mathbf{t} = H(M, \text{salt})$ (where salt is a random value), and samples a preimage \mathbf{u} such that $\mathcal{P}^*(\mathbf{u}) = \mathbf{t}$, leveraging the trapdoor tp . The final signature is $\text{sig} := (\text{salt}, \mathbf{u})$.

The transform by Abe, Ohkubo and Suzuki [AOS02] allows us to generically turn MAYO into an RS. We defer the full description to Appendix I.2 of [Kat+25]. Analogously to FalconRS, the final MayoRS ring signature includes a pre-image $\mathbf{u} \in \mathbb{F}_q^{kn}$ for each ring user, as well as a seed generating a target. In the two-user setting, the signature size is thus roughly doubled over MAYO.

Security Analysis and Parameters. We here overview the unforgeability and deniability of MayoRS, deferring formal statements and proofs to Appendix I.2.1 of [Kat+25].

Unforgeability. Using the generic transform of [AOS02], we reduce the unforgeability of MayoRS to that of MAYO. We note that there is a loss in the reduction, linear in the number of random oracle queries made by the adversary. This does not, we consider, lead to an improved attack.

Deniability. We prove deniability of MayoRS by observing that first sampling a target \mathbf{t} then a pre-image \mathbf{u} of \mathbf{t} is roughly equivalent to first sampling $\mathbf{u} \leftarrow \mathbb{F}_q^{kn}$ and taking $\mathbf{t} = \mathcal{P}^*(\mathbf{u})$. A small portion B of the vectors \mathbf{u} cannot be sampled in the first scenario. Adjusting for the number of system participants N , this leads to deniability with $(\mu = 1, \delta = 2N \cdot B)$, where $2N$ is a tightness slack.

One could directly use the parameters from the MAYO specification, but MAYO₁ and MAYO₂ achieve rather low deniability, so we provide three alternative parameter sets for NIST level I, with different trade-offs in size and deniability. We denote them MAYO*, MAYO**, and MAYO*** and detail their parameters in Appendix I.2.2 of [Kat+25].

We compare the sizes and performance of FalconRS and MayoRS with the original schemes in Tables 2 and 3. Ring signature sizes and computation times are all approximately doubled over the base scheme. We also include deniability guarantees achieved by each scheme. We provide implementations and compare our RSs to prior works in Section 6.3 of [Kat+25]. Our constructions are as compact as the state-of-the-art, and built on more scrutinized schemes. Additionally, our implementations largely outperform the state-of-the-art, by factors 32–66× for signing, 146–1025× for verification.

Table 2: Sizes and deniability (μ, δ) of FalconRS and MayoRS depending on the base scheme, aiming for NIST level I.

Base Scheme	(μ, δ)	PK	Sig	2-RSig
Falcon-512	$1 + 2^{-27}, 2^{-57}$	897 B	666 B	1288 B
MAYO ₁	$1, 2N \cdot 2^{-36}$	1168 B	321 B	650 B
MAYO ₂	$1, 2N \cdot 2^{-36}$	5488 B	180 B	368 B
MAYO*	$1, 2N \cdot 2^{-52}$	1569 B	335 B	677 B
MAYO**	$1, 2N \cdot 2^{-83}$	1591 B	374 B	756 B
MAYO***	$1, 2N \cdot 2^{-124}$	1771 B	492 B	992 B

Table 3: Performance of normal and 2-ring versions of Falcon and MAYO. Experiments executed on a Ryzen Pro 7 5850U @ 3GHz. Numbers are in Megacycles (Mc).

Scheme	Keygen	Sign		Verify	
		normal	2-ring	normal	2-ring
Falcon-512	6.2 Mc	0.26 Mc	0.74 Mc	0.02 Mc	0.04 Mc
MAYO ₁	0.24 Mc	0.88 Mc	1.1 Mc	0.17 Mc	0.28 Mc
MAYO ₂	0.65 Mc	1.1 Mc	1.5 Mc	0.09 Mc	0.16 Mc

10 Comparison

In this section, we will first compare the security and deniability properties of the discussed protocols, followed by a comparison of the efficiency of the different schemes.

Security. The BAKE abstraction and security model allows us to make a direct comparison of the security properties of Signal handshake protocols; we show an overview in Table 5. By setting the powers of the adversary and modeling unavoidable attacks, we were able to show that PQXDH is indeed secure against HNDL attacks, as long as the adversary is not able to obtain the secrets for the post-quantum KEM prekeys. Additionally, receivers in both X3DH and PQXDH cannot avoid user state compromise impersonation attacks, while senders are only weakly forward secure. Our proposal, RingXKEM, is post-quantum, and proving its security does not require ruling out additional unavoidable attacks: it is secure against user-state compromise impersonation attacks and fully forwards secure.

Deniability. Through our deniability model, we showed that X3DH and PQXDH have (strong) local deniability, but (strong) global deniability requires that the (classical) distinguisher does not obtain the secret corresponding to the prekey bundle used. Against quantum distinguishers, show strong deniability of PQXDH for accused receivers. Looking at the quantum-safe proposal RingXKEM, we show that though we get local deniability with the highest leakage possible against honest-but-curious accusers, global deniability relies on the deletion of user states. Furthermore, showing sender deniability against malicious receivers is not possible without a proof that the ring verification key was honestly generated.

Table 5: Security comparison of BAKE protocols.

Protocol	Adversary	Forward Secrecy	User-State Compromise Impersonation	Protocol-specific adversary restrictions
X3DH	Classical	Sender: weak Receiver: full	Receiver vulnerable	No quantum/HNDL adversaries.
PQXDH	HNDL	Sender: weak Receiver: full	Receiver vulnerable	KEM secret can not be revealed to HNDL adversary.
RingXKEM	Quantum	Full	Secure	No RingXKEM specific restrictions.

Efficiency. Table 6 compares instantiations of X3DH, PQXDH and RingXKEM, as well as an overview of how far from (currently) standardized cryptography the instantiations are. All schemes use a verification key as identity public key; RingXKEM additionally uses a KEM public key. We show the sizes using Kyber-1024 following PQXDH, as well as with Kyber-512 which matches the security of the (ring) signature scheme. The X3DH prekey bundle consists of two ECDH public keys and a signature, PQXDH adds a KEM public key and a signature. For RingXKEM, the prekey bundle is a KEM public key, a Merkle tree authentication path, a signature on the root of this tree, and an RS verification key. The X3DH handshake message is an ECDH public key and an authentication tag; PQXDH adds a KEM ciphertext. For RingXKEM, this message is an encrypted (ring) signature and two ciphertexts.

Due to our Merkle tree optimization, the increase in server storage requirements for the prekey bundles is only about 1 kB in total. The RingXKEM instantiation based on NIST standard Falcon [Pre+22] is also close in bandwidth requirements to the instantiation based on custom scheme Gandalf, for similar levels of deniability.

Regarding the runtime of RingXKEM, in Section 9, we already discussed performance metrics for the proposed ring signatures. As they are well under typical network latencies, we will conclude that all primitives considered are computationally efficient; we provide a further discussion of RingXKEM’s performance compared to X3DH and PQXDH in Section 6 of [HKW25].

Table 6: Practicality of deniable Signal handshake protocols. Batch size $L = 100$, sizes in bytes.

Protocol	Fully PQ	KEX	Authentication Primitive	Identity public key	Prekey bundle		Handshake message	Standardized Crypto
					Individual	L -key storage		
X3DH	✗	X25519	XEd25519 [Per16]	32	128	3296	64	Derived from X25519
PQXDH	✗	DH+Kyber-1024	XEd25519 [Per16]	32	1696	166 496	1632	Derived from X25519
RingXKEM	✓	Kyber-512	Gandalf [GJK24b]	1696	2582	81 526	2804	Custom scheme
RingXKEM	✓	Kyber-1024	Gandalf [GJK24b]	2464	3350	158 326	4404	Custom scheme
RingXKEM	✓	Kyber-512	FalconRS	1697	2619	81 563	2856	Based on Falcon
RingXKEM	✓	Kyber-1024	FalconRS	2465	3387	158 363	4456	Based on Falcon
RingXKEM	✓	Kyber-512	MAYO*	2369	2960	81 904	2245	Based on MAYO
RingXKEM	✓	Kyber-1024	MAYO*	3137	3728	158 704	3845	Based on MAYO

Acknowledgements

This paper is based on results obtained from a project, JPNP24003, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

- [AOS02] Masayuki Abe, Miyako Ohkubo and Koutarou Suzuki. “1-out-of- n Signatures from a Variety of Keys”. In: *ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. LNCS. Springer, Berlin, Heidelberg, Dec. 2002, pp. 415–432. DOI: [10.1007/3-540-36178-2_26](https://doi.org/10.1007/3-540-36178-2_26) (cit. on pp. 15, 16).

- [Bac+15] Michael Backes, Aniket Kate, Sebastian Meiser and Tim Ruffing. “Secrecy Without Perfect Randomness: Cryptography with (Bounded) Weak Sources”. In: *ACNS 2015*. Ed. by Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko and Michalis Polychronakis. Vol. 9092. LNCS. Springer, Cham, June 2015, pp. 675–695. doi: [10.1007/978-3-319-28166-7_33](https://doi.org/10.1007/978-3-319-28166-7_33) (cit. on pp. 2, 10).
- [Beu+24] Ward Beullens, Fabio Campos, Sofia Celi, Basil Hess and Matthias J. Kannwischer. *MAYO*. Tech. rep. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures>. National Institute of Standards and Technology, 2024 (cit. on pp. 2, 14).
- [Bha+24] Karthikeyan Bhargavan, Charlie Jacomme, Franziskus Kiefer and Rolfe Schmidt. “Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging”. In: *USENIX Security 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, Aug. 2024. URL: <https://www.usenix.org/conference/usenixsecurity24/presentation/bhargavan> (cit. on p. 4).
- [BK10] Zvika Brakerski and Yael Tauman Kalai. *A Framework for Efficient Signatures, Ring Signatures and Identity Based Encryption in the Standard Model*. Cryptology ePrint Archive, Report 2010/086. 2010. URL: <https://eprint.iacr.org/2010/086> (cit. on p. 14).
- [Bre+22] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson and Douglas Stebila. “Post-quantum Asynchronous Deniable Key Exchange and the Signal Handshake”. In: *PKC 2022, Part II*. Ed. by Goichiro Hanaoka, Junji Shikata and Yohei Watanabe. Vol. 13178. LNCS. Springer, Cham, Mar. 2022, pp. 3–34. doi: [10.1007/978-3-030-97131-1_1](https://doi.org/10.1007/978-3-030-97131-1_1) (cit. on pp. 1–3, 7, 9).
- [CF11] Cas Cremers and Michele Feltz. *One-round Strongly Secure Key Exchange with Perfect Forward Secrecy and Deniability*. Cryptology ePrint Archive, Report 2011/300. 2011. URL: <https://eprint.iacr.org/2011/300> (cit. on p. 7).
- [Coh+17] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt and Douglas Stebila. “A Formal Security Analysis of the Signal Messaging Protocol”. In: *2017 IEEE European Symposium on Security and Privacy*. IEEE Computer Society Press, Apr. 2017, pp. 451–466. doi: [10.1109/EuroSP.2017.27](https://doi.org/10.1109/EuroSP.2017.27) (cit. on pp. 3, 7).
- [Coh+20] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt and Douglas Stebila. “A Formal Security Analysis of the Signal Messaging Protocol”. In: *Journal of Cryptology* 33.4 (Oct. 2020), pp. 1914–1983. doi: [10.1007/s00145-020-09360-1](https://doi.org/10.1007/s00145-020-09360-1) (cit. on pp. 3, 7).
- [Col+24] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin and Serge Vaudenay. “K-Waay: Fast and Deniable Post-Quantum X3DH without Ring Signatures”. In: *USENIX Security 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, Aug. 2024. URL: <https://www.usenix.org/conference/usenixsecurity24/presentation/collins> (cit. on pp. 1, 3, 7, 9).
- [CPZ20] Melissa Chase, Trevor Perrin and Greg Zaverucha. “The Signal Private Group System and Anonymous Credentials Supporting Efficient Verifiable Encryption”. In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1445–1459. doi: [10.1145/3372297.3417887](https://doi.org/10.1145/3372297.3417887) (cit. on p. 12).
- [Dag+13] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach and Cristina Onete. “A Cryptographic Analysis of OPACITY - (Extended Abstract)”. In: *ESORICS 2013*. Ed. by Jason Crampton, Sushil Jajodia and Keith Mayes. Vol. 8134. LNCS. Springer, Berlin, Heidelberg, Sept. 2013, pp. 345–362. doi: [10.1007/978-3-642-40203-6_20](https://doi.org/10.1007/978-3-642-40203-6_20) (cit. on p. 7).
- [DGK06] Mario Di Raimondo, Rosario Gennaro and Hugo Krawczyk. “Deniable authentication and key exchange”. In: *ACM CCS 2006*. Ed. by Ari Juels, Rebecca N. Wright and Sabrina De Capitani di Vimercati. ACM Press, Oct. 2006, pp. 400–409. doi: [10.1145/1180405.1180454](https://doi.org/10.1145/1180405.1180454) (cit. on pp. 7, 8, 10).
- [DNS98] Cynthia Dwork, Moni Naor and Amit Sahai. “Concurrent Zero-Knowledge”. In: *30th ACM STOC*. ACM Press, May 1998, pp. 409–418. doi: [10.1145/276698.276853](https://doi.org/10.1145/276698.276853) (cit. on p. 10).
- [Dwo+06] Cynthia Dwork, Frank McSherry, Kobbi Nissim and Adam Smith. “Calibrating Noise to Sensitivity in Private Data Analysis”. In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Berlin, Heidelberg, Mar. 2006, pp. 265–284. doi: [10.1007/11681878_14](https://doi.org/10.1007/11681878_14) (cit. on pp. 2, 10).
- [FG24] Rune Fiedler and Felix Günther. *Security Analysis of Signal’s PQXDH Handshake*. Cryptology ePrint Archive, Report 2024/702. 2024. URL: <https://eprint.iacr.org/2024/702> (cit. on pp. 3, 4, 7).
- [FJ24] Rune Fiedler and Christian Janson. “A Deniability Analysis of Signal’s Initial Handshake PQXDH”. In: *Proceedings on Privacy Enhancing Technologies 2024* (Oct. 2024), pp. 907–928. doi: [10.56553/popets-2024-0148](https://doi.org/10.56553/popets-2024-0148) (cit. on p. 7).

- [GJK24a] Phillip Gajland, Jonas Janneck and Eike Kiltz. *A Closer Look at Falcon*. Cryptology ePrint Archive, Paper 2024/1769. 2024. URL: <https://eprint.iacr.org/2024/1769> (cit. on p. 14).
- [GJK24b] Phillip Gajland, Jonas Janneck and Eike Kiltz. “Ring Signatures for Deniable AKEM: Gandalf’s Fellowship”. In: *CRYPTO 2024, Part I*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14920. LNCS. Springer, Cham, Aug. 2024, pp. 305–338. DOI: [10.1007/978-3-031-68376-3_10](https://doi.org/10.1007/978-3-031-68376-3_10) (cit. on pp. 2, 14, 15, 17).
- [Goo22] Google. *Messages End-to-End Encryption Overview*. Technical paper. Feb. 2022. URL: https://www.gstatic.com/messages/papers/messages_e2ee.pdf (cit. on p. 1).
- [Has+21] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski and Thomas Prest. “An Efficient and Generic Construction for Signal’s Handshake (X3DH): Post-Quantum, State Leakage Secure, and Deniable”. In: *PKC 2021, Part II*. Ed. by Juan Garay. Vol. 12711. LNCS. Springer, Cham, May 2021, pp. 410–440. DOI: [10.1007/978-3-030-75248-4_15](https://doi.org/10.1007/978-3-030-75248-4_15) (cit. on pp. 2, 3, 6, 7).
- [Has+22] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski and Thomas Prest. “An Efficient and Generic Construction for Signal’s Handshake (X3DH): Post-quantum, State Leakage Secure, and Deniable”. In: *Journal of Cryptology* 35.3 (July 2022), p. 17. DOI: [10.1007/s00145-022-09427-1](https://doi.org/10.1007/s00145-022-09427-1) (cit. on pp. 1–3, 6, 7, 9).
- [HKW25] Keitaro Hashimoto, Shuichi Katsumata and Thom Wiggers. “Bundled Authenticated Key Exchange: A Concrete Treatment of (Post-Quantum) Signal’s Handshake Protocol”. In: *USENIX Security 2025*. to appear. USENIX, 13th Jan. 2025. URL: <https://eprint.iacr.org/2025/040> (cit. on pp. 2–6, 11, 17).
- [Jia+22] Shaoquan Jiang, Yeow Meng Chee, San Ling, Huaxiong Wang and Chaoping Xing. “A new framework for deniable secure key exchange”. In: *Inf. Comput.* 285.PB (May 2022). ISSN: 0890-5401. DOI: [10.1016/j.ic.2022.104866](https://doi.org/10.1016/j.ic.2022.104866). URL: <https://doi.org/10.1016/j.ic.2022.104866> (cit. on p. 9).
- [Kat+25] Shuichi Katsumata, Guilhem Niot, Ida Tucker and Thom Wiggers. “Concrete Treatment of Signal Handshake’s Deniability: Efficient Post-Quantum Deniable Ring Signature”. In: *USENIX Security 2025*. to appear. USENIX, 2025. URL: <https://eprint.iacr.org/2025/xxx> (cit. on pp. 2, 3, 11–16).
- [KBB17] Nadim Kobeissi, Karthikeyan Bhargavan and Bruno Blanchet. “Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach”. In: *2017 IEEE European Symposium on Security and Privacy*. IEEE Computer Society Press, Apr. 2017, pp. 435–450. DOI: [10.1109/EuroSP.2017.38](https://doi.org/10.1109/EuroSP.2017.38) (cit. on p. 4).
- [KP05] Caroline Kudla and Kenneth G. Paterson. “Modular Security Proofs for Key Agreement Protocols”. In: *ASIACRYPT 2005*. Ed. by Bimal K. Roy. Vol. 3788. LNCS. Springer, Berlin, Heidelberg, Dec. 2005, pp. 549–565. DOI: [10.1007/11593447_30](https://doi.org/10.1007/11593447_30) (cit. on pp. 1, 4).
- [KS23] Ehren Kret and Rolfe Schmidt. *The PQXDH Key Agreement Protocol*. Protocol documentation. 18th Oct. 2023. URL: <https://signal.org/docs/specifications/pqxdh/> (cit. on pp. 1, 2, 4, 5, 7).
- [LAZ19] Xingye Lu, Man Ho Au and Zhenfei Zhang. “Raptor: A Practical Lattice-Based (Linkable) Ring Signature”. In: *ACNS 2019*. Ed. by Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa and Moti Yung. Vol. 11464. LNCS. Springer, Cham, June 2019, pp. 110–130. DOI: [10.1007/978-3-030-21568-2_6](https://doi.org/10.1007/978-3-030-21568-2_6) (cit. on p. 14).
- [Met23] Meta, Inc. *Messenger End-to-End Encryption Overview*. Technical white paper. 6th Dec. 2023. URL: https://engineering.fb.com/wp-content/uploads/2023/12/MessengerEnd-to-EndEncryptionOverview_12-6-2023.pdf (cit. on p. 1).
- [Mir+09] Ilya Mironov, Omkant Pandey, Omer Reingold and Salil P. Vadhan. “Computational Differential Privacy”. In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Berlin, Heidelberg, Aug. 2009, pp. 126–142. DOI: [10.1007/978-3-642-03356-8_8](https://doi.org/10.1007/978-3-642-03356-8_8) (cit. on pp. 2, 10).
- [MP16] Moxie Marlinspike and Trevor Perrin. *The X3DH Key Agreement Protocol*. Protocol documentation. 4th Nov. 2016. URL: <https://signal.org/docs/specifications/x3dh/> (cit. on pp. 1, 2, 4, 5, 7).
- [Pas03] Rafael Pass. “On Deniability in the Common Reference String and Random Oracle Model”. In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Berlin, Heidelberg, Aug. 2003, pp. 316–337. DOI: [10.1007/978-3-540-45146-4_19](https://doi.org/10.1007/978-3-540-45146-4_19) (cit. on p. 10).
- [Per16] Trevor Perrin. *The XEdDSA and VXEdDSA Signature Schemes*. documentation. 20th Oct. 2016. URL: <https://signal.org/docs/specifications/xeddsa/> (cit. on p. 17).
- [PM16] Trevor Perrin and Moxie Marlinspike. *The Double Ratchet Algorithm*. Protocol documentation. 20th Nov. 2016. URL: <https://signal.org/docs/specifications/doubleratchet/> (cit. on p. 1).

- [Pre+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte and Zhenfei Zhang. *FALCON*. Tech. rep. available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>. National Institute of Standards and Technology, 2022 (cit. on pp. 2, 10, 13, 14, 17).
- [Res18] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Aug. 2018. doi: [10.17487/RFC8446](https://doi.org/10.17487/RFC8446) (cit. on p. 5).
- [Sch24] Rolfe Schmidt. “Private communications”. 2024 (cit. on p. 5).
- [Sho97] Victor Shoup. “Lower Bounds for Discrete Logarithms and Related Problems”. In: *EUROCRYPT’97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Berlin, Heidelberg, May 1997, pp. 256–266. doi: [10.1007/3-540-69053-0_18](https://doi.org/10.1007/3-540-69053-0_18) (cit. on p. 12).
- [SV16] Igal Sason and Sergio Verdú. “ f -Divergence Inequalities”. In: *IEEE Transactions on Information Theory* 62.11 (2016), pp. 5973–6006. doi: [10.1109/TIT.2016.2603151](https://doi.org/10.1109/TIT.2016.2603151) (cit. on p. 10).
- [UG15] Nik Unger and Ian Goldberg. “Deniable Key Exchanges for Secure Messaging”. In: *ACM CCS 2015*. Ed. by Indrajit Ray, Ninghui Li and Christopher Kruegel. ACM Press, Oct. 2015, pp. 1211–1223. doi: [10.1145/2810103.2813616](https://doi.org/10.1145/2810103.2813616) (cit. on p. 7).
- [UG18] Nik Unger and Ian Goldberg. “Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging”. In: *PoPETs 2018.1* (Jan. 2018), pp. 21–66. doi: [10.1515/popets-2018-0003](https://doi.org/10.1515/popets-2018-0003) (cit. on p. 7).
- [Vat+20] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn and Hugo Krawczyk. “On the Cryptographic Deniability of the Signal Protocol”. In: *ACNS 2020, Part II*. Ed. by Mauro Conti, Jianying Zhou, Emiliano Casalichio and Angelo Spognardi. Vol. 12147. LNCS. Springer, Cham, Oct. 2020, pp. 188–209. doi: [10.1007/978-3-030-57878-7_10](https://doi.org/10.1007/978-3-030-57878-7_10) (cit. on pp. 7–9).
- [Wha23] WhatsApp. *WhatsApp Encryption Overview*. Technical white paper. 27th Sept. 2023. URL: <https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf> (cit. on p. 1).

A Definition of local and global deniability

Formally, we define local and global deniability as follows.

Definition A.1 (Local deniability). Let $N := [N]$ for $N \in \mathbb{N}$ be the set of all users, $\mathcal{H} \subseteq N$ such that $\mathcal{H} \neq \emptyset$ be the set of accused users, and $C := N \setminus \mathcal{H}$ be the set of accusers. Let $Q = \text{poly}(\lambda)$ be an upper bound on the number of oracle queries made by the distinguisher D .

A BAKE protocol is (μ, δ) -locally deniable against honest-but-curious accusers with respect to leakage function $\mathcal{L}_{\text{leak}}$ and disclosure function $\mathcal{D}_{\text{disc}}$ with $\text{leak}, \text{disc} \in \{\text{low}, \text{med}, \text{high}\}$, if there exists an efficient simulator $\text{Sim} = (\text{SimPreK}, \text{SimTrans}, \text{SimSt}_{\mathcal{H}}, \text{SimSt}_C)$ such that for any efficient distinguisher D we have

$$\Pr \left[\text{Game}_{D, \mathcal{H}, \mathcal{L}_{\text{leak}}, \mathcal{D}_{\text{disc}}}^{\text{local}}(1^\lambda, \text{real}) = 0 \right] \leq \mu(\lambda, Q) \cdot \Pr \left[\text{Game}_{D, \mathcal{H}, \mathcal{L}_{\text{leak}}, \mathcal{D}_{\text{disc}}}^{\text{local}}(1^\lambda, \text{sim}) = 0 \right] + \delta(\lambda),$$

where $\text{Game}_{D, \mathcal{H}, \mathcal{L}_{\text{leak}}, \mathcal{D}_{\text{disc}}}^{\text{local}}$ is given in Alg. 9.

Definition A.2 (Global deniability). We define global deniability identically to local deniability in Definition A.1 except that the distinguisher D plays the game $\text{Game}_{D, \mathcal{H}, \mathcal{L}_{\text{leak}}, \mathcal{D}_{\text{disc}}}^{\text{global}}$ in Alg. 9.

```

1: function GameATKD, H, Lleak, Ddisc(1λ, mode)
2:   C := N \ H
3:   for user u ∈ N do
4:     (iku, isku) ←s BAKE.IdKeyGen(1λ)
5:     counteru := 0 ▷ Track how many prekey bundles are used
6:     if [u ∈ H] then
7:       (preku, stu) ←s BAKE.PreKeyBundleGen(isku)
8:     if [mode = real] then
9:       for user u ∈ C do
10:        (preku, stu) ←s BAKE.PreKeyBundleGen(isku)
11:        stuinit := stu
12:     else ▷ mode = sim
13:       ▷ Simulate prekey bundles of accused users H
14:       ((preku*)u ∈ H, (preku, stu)u ∈ C, stSim) ←s SimPreK((iku)u ∈ N, (isku)u ∈ C, (preku)u ∈ H)
15:       (preku)u ∈ H ← (preku*)u ∈ H
16:       stD ←s DOATK((iku, preku)u ∈ N) ▷ D obtains transcripts
17:       if [mode = sim] then
18:         (stu, stuinit)u ∈ C ←s SimStC(stSim) ▷ Corrupted states
19:         leakD := Lleak((isku, stu)u ∈ H)
20:         discD := Ddisc((isku, stu, stuinit)u ∈ C)
21:         auxD := (leakD, discD)
22:         b ←s D(stD, auxD) ▷ D outputs a bit b ∈ {0, 1}
23:       return b
24: function OATK(s, r)
25:   require [(s, r) ∈ N × N] ∧ [s ≠ r] ∧ [(s, r) ∉ C × C]
26:   if [ATK = Local] then require [(s, r) ∉ H × H]
27:   counterr ← counterr + 1; t := counterr
28:   if [t > L] then t ← ⊥ ▷ Use last-resort prekey bundle
29:   if [mode = real] then ▷ Run real sender and receiver
30:     (K, ρ) ←s BAKE.Send(isks, ikr, prekr, t)
31:     (K', str) ←s BAKE.Receive(iskr, str, iks, t, ρ)
32:   else ▷ mode = sim
33:     if [(s, r) ∈ H × C] then ▷ Simulate with receiver secrets
34:       (K, ρ, stSim) ←s SimTrans(iskr, stSim, (s, r, t))
35:     else ▷ Honest receiver
36:       if [(s, r) ∈ C × H] then ▷ Simulate with sender secrets
37:         (K', ρ, stSim) ←s SimTrans(isks, stSim, (s, r, t))
38:       else ▷ (s, r) ∈ H × H ⇒ ATK = Global
39:         ▷ Simulate without any secrets
40:         (K, K', ρ, stSim) ←s SimTrans(⊥, stSim, (s, r, t))
41:       str ←s SimStH(iskr, str) ▷ Update honest receiver state
42:   if [(s, r) ∈ H × C] then return (K, ρ)
43:   else if [(s, r) ∈ C × H] then return (K', ρ)
44:   else return (K, K', ρ) ▷ (s, r) ∈ H × H ⇒ ATK = Global

```

Algorithm 9: Games for local and global deniability with respect to leakage function L_{leak} , and disclosure function D_{disc} . Boxed text is only relevant to global deniability.