USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Comprehensive Deniability Analysis of Signal Handshake Protocols: X3DH, PQXDH to Fully Post-Quantum with Deniable Ring Signatures

Shuichi Katsumata, *PQShield & AIST;* Guilhem Niot, *PQShield & Univ Rennes, CNRS, IRISA;* Ida Tucker and Thom Wiggers, *PQShield*

https://www.usenix.org/conference/usenixsecurity25/presentation/katsumata

This paper is included in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

Comprehensive Deniability Analysis of Signal Handshake Protocols: X3DH, PQXDH to Fully Post-Quantum with Deniable Ring Signatures

Shuichi Katsumata [1,2] *Guilhem Niot* [1,3] *Ida Tucker* [1]
Thom Wiggers [1]
[1] *PQShield*
[2] *AIST*
[3] *Univ Rennes, CNRS, IRISA*

Abstract

The Signal protocol relies on a 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 deniability for these protocols, including post-quantum variants, use models highly tailored to the individual protocols and generally make ad-hoc adaptations to "standard" AKE definitions, obscuring the concrete deniability guarantees and complicating comparisons across protocols. Building on Hashimoto et al.'s abstraction for Signal handshake protocols (USENIX'25), we address this gap by presenting a unified framework for analyzing their deniability. We analyze Signal's classically secure X3DH and harvest-now-decrypt-later-secure PQXDH, and show that PQXDH is deniable against harvest-now-*judge*-later attacks, where a quantum judge retrospectively assesses the participation of classical users. We further analyze post-quantum alternatives like RingXKEM, whose deniability relies on ring signatures (RS). By introducing a novel metric inspired by differential privacy, we provide 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.

1 Introduction

The Signal protocol [38, 45] does not just power the Signal app, it also underpins messaging apps such as WhatsApp [54], Google RCS [25], and Facebook Messenger [39], 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 [45]. This handshake was originally implemented as X3DH [38], based on Triple Diffie-Hellman [34]. In late 2023, as a step towards fully post-quantum (PQ) security, X3DH was replaced with PQXDH [33], offering protection against "harvest-now-decrypt-later" (HNDL) attacks. There also exist several proposals for fully PQ Signal handshake protocols [8, 12, 27, 29].

Until recently, 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. Recently, Hashimoto et al. [29] proposed *Bundled Authenticated Key Exchange* (BAKE) protocols, allowing to analyze existing Signal handshake protocols in a unified manner. This was 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. This led to a more efficient PQ handshake protocol, RingXKEM, relying on ring signatures (RS) and Merkle trees, which could not have been captured in previous models.

The main goal of [29] was to define a security model for BAKE, treating key indistinguishability and authentication properties. However, the issue of *deniability* — one of the key features of the original Signal protocol [33, 38] — was left open. 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 [33, 38]. Selecting the most suitable protocols thus requires not only considering key indistinguishability and authentication guarantees, but also placing equal emphasis on their deniability guarantees. However, as was the case with key indistinguishability and authentication, existing protocols have been analyzed using tailored deniability models [8, 12, 22, 27, 53] — often treating protocols informally as AKE protocols and disregarding the implication of using prekey bundles — thereby giving incomplete analyses and complicating meaningful comparisons.

1.1 Contributions

In this work, we propose a unified framework for analyzing the deniability of BAKE protocols, thereby completing the formal treatment of Signal handshake protocols initiated by [29]. We analyze X3DH and PQXDH, proving for the first time that PQXDH is deniable even against *quantum* distinguishers. Additionally, we examine the deniability of fully PQ BAKE protocols, such as RingXKEM, and provide instantiations of ring signatures based on NIST-standardized signatures, to encourage adoption. We will now detail these contributions.

Unified deniability model. We formally capture the deniability of general Signal handshake (i.e., BAKE) protocols, building upon prior work on the deniability of AKE [13, 14, 16, 51, 52], and adaptations made for specific handshake protocols [8, 12, 22, 27, 53]. Following the general approach common to existing simulation-based definitions, we define an *accuser*, who collects evidence which is provided to a *distinguisher*[1], who, in turn, decides if the evidence could have been simulated by the accuser. Our model differentiates information that may leak from the accused user's device, and evidence disclosed by accusers, to the distinguisher. We account for varying adversarial capabilities, distinguishing between honest-but-curious accusers (*standard* deniability) and malicious accusers (*strong* deniability), as did [27]. We also introduce notions of *local* deniability, where an accused participant can deny having engaged in a BAKE protocol with the accuser, and a strictly stronger *global* deniability, which further ensures both participants can deny their involvement, even if the accuser is an outsider to the conversation. Our notions of deniability benefit from a hierarchical structure, allowing for ease of comparisons.

Unique features of our model. By adopting the BAKE formalism of [29], 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 [33, 38]. Our model distinguishes deniability guarantees based on whether one-time prekeys are depleted, revealing interesting separations. For example, in X3DH and PQXDH, using last-resort prekeys does not affect local deniability, but does harm global deniability, just as it impacted key indistinguishability in [29].

Moreover, we highlight that when a batch of prekey bundles is generated, a *user state* is generated — this was 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 removed after the receiver completes the 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. See Sec. 1.2 for more details.

Harvest now judge later. Extending beyond classical deniability, we consider deniability against quantum accusers and distinguishers. We also consider scenarios — relevant *today* — where the accuser is classical, but the distinguisher is quantum. Indeed, transcripts stored now could later be exploited when quantum capabilities become available, a risk we term "harvest-now-*judge*-later" (HNJL).[2] Surprisingly, while harvest-now-*decrypt*-later attacks have garnered significant attention, particularly for the PQXDH protocol, no prior work provides this level of analysis for deniability. We close this gap and prove PQXDH to be deniable against HNJL attacks.

A pragmatic metric for deniability. We introduce a novel measure for deniability inspired by concepts in differential privacy and differential indistinguishability [2, 19, 40]. Prior works required the real evidence π_{real} and simulated evidence π_{sim} to be indistinguishable by the distinguisher D. That is, the statistical distance of the two distributions is close: $|\Pr[\mathsf{D}(\pi_{\mathsf{real}}) = 0] \approx \Pr[\mathsf{D}(\pi_{\mathsf{sim}}) = 0]|$. 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. We thus only require a relaxed condition: $\Pr[\mathsf{D}(\pi_{\mathsf{real}}) = 0] \approx \mu \cdot \Pr[\mathsf{D}(\pi_{\mathsf{sim}}) = 0]$ for some multiplicative slack μ (possibly non-negligibly) close to 1. Technically, this means the two distributions are close in terms of the *hockey-stick divergence* [48]. As discussed later, this new pragmatic metric for deniability is the key enabler for building efficient PQ (deniable) ring signatures from NIST-standardized signatures.

Analysis of X3DH and PQXDH. In the classical setting, we prove that both X3DH and PQXDH attain the highest level of (standard and strong) deniability one may hope for, so long as one-time prekey bundles are not depleted. Otherwise, deniability only holds if the distinguisher does not see the accused users' state (relevant to the conversation). This is because states associated with last-resort prekey-bundles are not immediately deleted after use; if the accused user's phone is seized before the next generation of a prekey bundle batch, this state leaks to the distinguisher and can be used to prove participation. This distinction highlights the advantage of using the BAKE framework [29], accurately modeling prekey bundles.

[1] Prior work has also used the term "Judge"; we prefer "distinguisher", which better reflects its algorithmic (as opposed to human) nature.

[2] To avoid confusion with the acronym of harvest-now-*decrypt*-later, we chose the term *judge*[1] as opposed to *distinguish*.

We further prove that PQXDH is *standard* HNJL deniable by considering quantum distinguishers in the quantum random oracle model (QROM), consistent with its HNDL key-indistinguishability security. However, due to technical issues in handling the QROM and generic group model (GGM) [49], we could not prove *strong* HNJL deniability without resorting to strong knowledge type assumptions[3]. We leave it as an important future work and defer the details to Sec. 4.2.

Analysis of fully PQ protocols. We extend our study to Hashimoto et al.'s fully PQ RingXKEM [29], which uses RSs. While they demonstrated its key indistinguishability properties, we finalize the security evaluation by proving deniability. Finally, we examine SignXKEM, a RingXKEM variant suggested in [26], 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. Table 1 summarizes the deniability of these protocols.

Ring signature instantiations. Proving the deniability of RingXKEM is straightforward if the underlying RS scheme is anonymous, but compact lattice-based RSs satisfying anonymity turn out difficult to construct; the most compact schemes [24, 35] only provide roughly 30 bits of security for anonymity when instantiated with concrete parameters (see Sec. 6 for details). 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 [47], and the additional signature candidate MAYO [3]. These RS schemes are as compact as the state-of-the-art. We further provide implementations outperforming previous works by a factor 32–66× for signing, and 146–1025× for verification. We note that, in line with the security proofs of the underlying signatures, our proofs are in the classical random oracle model (ROM), but not in the QROM.

1.2 Related Work

We refer to App. A for the general concept of deniability.

Deniability of X3DH. The introduction of Off-the-Record Messaging established deniability as a key feature for secure messaging [6]. The Signal protocol adopted this goal in its X3DH handshake [38], analyzed by Vatandas et al. [53] under the simulation-based model of [16] for AKEs. Their work proves offline deniability for X3DH under knowledge-of-exponent assumptions, and links the deniability of a communication session to the deniability of the key agreement protocol starting the session. This allows to extend results on the deniability of handshake protocols to the entire conversation.

Deniability of PQXDH. Signal's recently deployed PQXDH protocol [33] combines the classical X3DH handshake with a PQ KEM. While not fully PQ, it provides key indistinguishability against "harvest-now-decrypt-later" adversaries, as analyzed in [5, 21, 29]. The deniability guarantees of PQXDH were recently analyzed in [22], viewing it as a specific handshake protocol proposed by [21]. The analysis assumes a classical distinguisher, leaving deniability guarantees against "harvest-now-judge-later" adversaries open; and only considers distinguishers either having no access to *any* secret key, or full access to *all* secret keys (of both accusing and accused; and for identity keys and prekeys). They hence do not capture scenarios where accusers disclose more information to the distinguisher than accused users. We also note that their model does not distinguish the attained deniability guarantees when one-time prekey bundles are, or are not, depleted.

Deniability of fully PQ protocols. Hashimoto et al. [27] construct a PQ Signal handshake protocol from RSs, and show deniability against honest-but-curious quantum adversaries, using a simplification of the deniability model of [16]. To prove deniability against malicious (classical) accusers, they require knowledge assumptions. In concurrent work, Brendel et al. [8] proposed a similar protocol based on designated verifier signatures, along with a new game-based deniability notion, which focuses solely on sender deniability but captures scenarios where judges may coerce users to reveal secret keys. Recently, Collins et al. [12] introduced K-Waay, a protocol based on split-KEMs [9, 42]. They extend the framework of Brendel et al. to model receivers who hand over their entire state to the distinguisher. Collins et al. do not model one-time vs. last-resort prekey bundles, but rather suggest that, if prekey bundles run out, then the sender should reuse an old one. However, their deniability model does not capture prekey bundle reuse. Both [8, 12] only consider honest-but-curious accusers, and attain deniability against quantum distinguishers.

A more detailed comparison. We here outline differences between our deniability model for BAKE, and tailored deniability models from prior works. The primary differences are explained in Sec. 1.1. Compared to [27, 53], we allow the distinguisher to obtain leakage from honest users; accounting for scenarios where judges may coerce secret keys from accused users. The models introduced in [8, 22] allow judges to compromise secrets associated to identity keys and last resort prekeys of *all* users, and [12] also capture the leakage of state associated to one-time prekeys. While our model captures these scenarios, it also differentiates which specific keys leak from accused as opposed to accusing users. Additionally, [8, 12] do not incorporate the simulation of prekey generation, and limit their study to honest-but-curious adversaries. Global deniability is not captured by the models of [8, 12, 27, 53]. Fiedler et al. [22] require deniability of prekey bundle uploads

[3] Knowledge assumptions assume the existence of knowledge extractors, for which there exist no concrete constructions. The resulting simulators are hence not efficiently implementable in the real world.

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

Signal handshake protocol deniability properties

	Protocol:	X3DH Classic $\mathcal{A}$/D	PQXDH Classic $\mathcal{A}$/D	PQXDH Classic $\mathcal{A}$ Quantum D		RingXKEM Classic or Quantum $\mathcal{A}$/D		SignXKEM Classic or Quantum $\mathcal{A}$/D	
	Deniability Level	**Leakage** leak	disc	**Leakage** leak	disc	**Leakage** leak	disc	**QROM**	**Leakage** leak	disc	**QROM**	**Leakage** leak	disc	**QROM**
	local	●	●	●	●	●	●	✓	●	●	✓	◕	◴	✓
	global	◕	●	◕	●	◕	●	✓	●	●	✓	◕	◴	✓
strong-	local	●†	●	●†	●	?	?	?	●^{SO}	●	?	◕^{SO}	●	?
strong-	global	◕†	●	◕†	●	?	?	?	●^{SO}	●	?	◕^{SO}	●	?
	Security [29]	Classical		Harvest-Now Decrypt-Later				Fully post-quantum						

Legend

Icon	Last-resort prekey: No	Yes
	leak or disc	leak or disc
●	high	high
◕	high	med
◴	med	low
?	Open problem	
◌^{SO}	Accusers $\mathcal{A}$ restricted to being senders, no deniability otherwise.	
†	Proof using GGM.	

Example: RingXKEM is local deniable with leakage leak = high and disclosure disc = high even if a last-resort prekey bundle was used. SignXKEM is local deniable with leak = high and disc = med, but restricted to leak = med and disc = low if a last resort prekey bundle is used.

Remark: For strong deniability, we always set disc = high, since we have no control over the information a malicious accuser may reveal.

(i.e. of the general use of the Signal protocol), whereas we consider deniability within specific conversations, this results in different conclusions regarding the deniability guarantees of PQXDH.

Scope of our work. We focus on the deniability of *messages*. This means that like prior work (e.g., [8, 12]), 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 handshake key by both sender and receiver, even if there is evidence of registration. Other security notions than deniability can also enhance privacy in secure messaging. For instance, one can add sender/receiver privacy [36, 37], metadata protection [**CCS:TovWeiGil24**, 28], anonymity and unlinkability [46, 50]. We leave the interplay of deniability and other privacy notions as an interesting future work. We also do not consider network-level adversaries. We believe countermeasures such as padding [41] or anonymous communication [17, 30] to be complimentary to deniability, but leave their analysis open.

Lastly, deniability is not the only way to enhance privacy in secure messaging. For instance, one can add sender/receiver privacy [36, 37], metadata protection [**CCS:TovWeiGil24**, 28], anonymity and unlinkability [46, 50].

Online Version. Due to space limitations, we defer some appendices to the full version. This version is freely available online at eprint.iacr.org/2025/1090.

2 Modeling Signal Handshake Protocols and Problem Setting

We model Signal handshake protocols as *bundled authenticated key exchange* (BAKE) protocols, introduced in [29]. BAKE is a variant of the traditional AKE model allowing to formally model prekey bundles and user states. It is general enough to capture Signal's X3DH and PQXDH [33, 38], among other constructions such as RingXKEM [29] and variants [8, 26, 27]. In [29], they defined *key indistinguishability* of BAKE, a fundamental security property for any key exchange protocol.

In this work, we define what it means for a BAKE to be *deniable*, a key security feature of Signal's X3DH and PQXDH protocols. Before introducing the formal definition of deniability, we recall the definition of a BAKE protocol and explain the high-level problem setting for deniability.

Standard notations and definitions. Let $\mathbb{Z}$ denote the natural numbers, and $\mathbb{Z}_N$ the natural numbers modulo N. If P is a point on an elliptic curve, we denote multiplication by scalar k as $[k]P$. We denote by λ the security parameter, and by $[N]$ the set of integers $\{1, \ldots, N\}$. Definitions of basic cryptographic primitives such as symmetric key encryption, signature schemes, and KEMs are deferred to App. A in the full version.

2.1 Modeling Signal Handshake Protocols with Bundled AKEs

We recall the definition of a BAKE protocol [29].

Definition 1. A *two-round bundled authenticated key exchange* protocol BAKE consists of the following four probabilistic polynomial time (PPT) algorithms, where $L \in \mathsf{poly}(\lambda)$.

$\mathsf{BAKE.IdKeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{ik}, \mathsf{isk})$: The identity key generation algorithm, on input security parameter 1^λ, outputs identity public key ik and associated secret key isk.

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

$\mathsf{BAKE.Send}(\mathsf{isk}_s, \mathsf{ik}_r, \mathsf{prek}_{r,t}) \xrightarrow{\$} (K, \rho)$: On input a sender s's identity secret key isk_s, the intended receiver r's identity key ik_r, and a prekey bundle $\mathsf{prek}_{r,t}$, outputs a session key K and a handshake message ρ.

$\mathsf{BAKE.Receive}(\mathsf{isk}_r, \mathsf{st}_r, \mathsf{ik}_s, t, \rho) \rightarrow (K', \mathsf{st}_r)$: The (deterministic) receiver algorithm, on input a receiver r's identity secret key isk_r and state st_r, a sender s's identity key ik_s, with the identifier of the used prekey bundle $t \in [L] \cup \{\perp\}$, and a handshake message ρ, outputs a key K' and an updated state st_r. Key agreement may fail, in which case $K' = \perp$, and the state is rolled back.

A BAKE protocol is a two-party protocol, with a server relaying communications. Unlike standard AKE protocols, BAKE supports prekey bundles, a feature central to Signal handshake protocols [33, 38]. Prekey bundles, uploaded to a server, are pre-generated key material consumed during new communication setups, enabling senders to establish secure sessions with offline recipients, facilitating asynchronous communication. Prekey bundles are generally one-time use, which naively limits the number of handshakes. To ensure availability, even during extended recipient offline periods, a *last-resort prekey bundle* is used when the list of one-time bundles is depleted. This bundle, designated by the label $\perp$, is not deleted after use until the next PreKeyBundleGen is performed once back online. In protocol execution, the server first distributes all one-time prekey bundles; and only once these are exhausted is the last-resort prekey used [38].

2.2 Deniability: Entities and Roles

The entities involved in deniability are categorized as follows, following established terminologies, e.g., [11, 16, 50].

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.

(Insider) accusers: The set of corrupted users, denoted as C, that communicate[4] with an accused user. Their goal is to prove that the accused user ran a BAKE protocol with them. We consider two levels of insider accusers: *honest-but-curious* accusers and *malicious* accusers. The former honestly follows the protocol description but may collect as much information as possible to accuse their peer; for instance, this captures a device injected by a malware, secretly storing all the states on the device while still using the official secure messaging application. 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 (often called "judge") outputting a *verdict* on whether a user participated in a BAKE protocol.

2.3 Distinguisher Capabilities

The distinguisher 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 [16, 53]. To make its verdict, the distinguisher may further be provided information through so-called *leakage* and *disclosure* functions. The former dictates how much information of the accused user leaks to the distinguisher, whereas the latter dictates how much information the accusing user discloses to the distinguisher. A BAKE protocol is more deniable if it allows leaking and disclosing more information to the distinguisher.

Leakage function for accused users. The amount of information leakage of an accused user is formalized using a function $\mathcal{L}_{\mathsf{leak}}$. We consider three levels of leakage: $\mathsf{leak} = \mathsf{low}$ is the weakest setting where no leakage occurs; $\mathsf{leak} = \mathsf{med}$ leaks the identity secret key; and, $\mathsf{leak} = \mathsf{high}$ leaks all secret information of the accused user.

Definition 2. The *leakage function* $\mathcal{L}_{\mathsf{leak}}$ for the set $\mathcal{H}$ of accused users of a BAKE protocol is defined as follows:

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

Disclosure function for honest-but-curious accusers. The amount of information disclosed by the honest-but-curious (insider) accusers is formalized using a function $\mathcal{D}_{\mathsf{disc}}$. Similarly to the above, we consider three levels of disclosures: $\mathsf{disc} = \mathsf{low}$ is the weakest setting where the accuser only discloses its identity secret key; $\mathsf{disc} = \mathsf{med}$ additionally discloses its *current* state; lastly, $\mathsf{disc} = \mathsf{high}$ 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.[5]

Definition 3. The *disclosure function* $\mathcal{D}_{\mathsf{disc}}$ for the set C of honest-but-curious (insider) accusers of a BAKE protocol is defined as follows:

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

Disclosure function for malicious accusers. Malicious accusers can deviate arbitrarily from the protocol, such as by maliciously generating prekeys, possibly without knowing the

[4] 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.

[5] 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.

associated secrets. We thus assume the malicious accuser to always disclose their entire state $\mathsf{st}_{\mathcal{A}}$. Note that despite the general unpredictability of malicious accusers, we require that $\mathsf{st}_{\mathcal{A}}$ includes their identity secret keys, based on the assumption that they register a valid identity public key (cf. Sec. 2.5).

2.4 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.

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

Formalizing what it means to "deny participating" is one of our main technical contributions, which we explain in Sec. 3. 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 — such as in highly sensitive communications, like 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.

2.5 Modeling Choices and Simplifications

This subsection outlines the modeling choices and assumptions underpinning our work.

Focus on deniability of protocol participation. We do not aim to address the deniability of being a user of the secure messaging application. Specifically, user registration of identity keys and prekey bundle uploads may not be deniable. Our goal is to show that users using the app can plausibly deny participating in any communication. Such practical levels of deniability were also considered in, e.g., [8, 12].

Honest registration of identity keys. Similarly to other works on the deniability of Signal handshake protocols [8, 32, 53], we assume that users *honestly* generate their identity keys. This simplifies the definition and proofs of deniability while maintaining practical relevance. Works allowing maliciously generated identity keys [22, 27] rely on strong knowledge type assumptions, forcing the proof to go through by pushing the burden onto the assumption. We can also rely on zero-knowledge proofs, guaranteeing that a registered identity public key has an associated identity secret key.

Disclosure vs. leakage. 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. This highlights that accusers are expected to actively provide information to the distinguisher to incriminate the accused user, and limits the leakage and disclosure combinations we analyze. We summarize the combinations of leakage and disclosure we consider in App. C in the full version.

3 Defining Deniable Bundled AKE Protocols

We define the *deniability* of a BAKE protocol. The main novelty in our definitions is a new pragmatic metric for deniability, inspired by differential privacy and differential indistinguishability [2, 19, 40], 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.

3.1 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. Sec. 2.3. A standard way to formalize this is to construct a *simulator* that can output simulated evidence indistinguishable from real evidence to a distinguisher [20]. This has been used to define deniable AKEs, e.g., [16]. We highlight that this simulator must not only exist but also be constructible in the real world [43]. An accused user must convince the judge (i.e., 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 [2, 19, 40]. 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: $|\Pr[\mathsf{D}(\pi_{\mathsf{real}}) = 0] - \Pr[\mathsf{D}(\pi_{\mathsf{sim}}) = 0]| = \delta(\lambda)$ 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: $\Pr[\mathsf{D}(\pi_{\mathsf{real}}) = 0] \leq \mu(\lambda) \cdot \Pr[\mathsf{D}(\pi_{\mathsf{sim}}) = 0] + \delta(\lambda)$. Technically, this means the two distributions are close in terms of the *hockey-stick divergence* [48]. While the original definition is recovered by setting $\mu(\lambda) = 1 + \mathsf{negl}(\lambda)$, 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.[6]

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 [47] 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 Sec. 5 and Sec. 6.1 for details.

3.2 Deniability Against Honest-but-Curious Accusers

We first define local and global deniability against honest-but-curious accusers,[7] formally defined through a game in Alg. 1. The distinguisher D is given the set of identity keys and prekey bundles, and can adaptively query transcripts exchanged between users (cf. Ln. 16). At the end of the game, D is given the leakage and disclosed information of the accused and accusing users (cf. Lns. 19 to 21). 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., $\texttt{mode} = \texttt{real}$) is overwhelmingly likely to be within some multiplicative slack μ of the probability D outputs b in the simulated world (i.e., $\texttt{mode} = \texttt{sim}$). Importantly, our definition captures the deniability of last-resort prekey bundles as well (cf. Ln. 28).

The simulated world is defined using a simulator $\mathsf{Sim} = (\mathsf{SimPreK}, \mathsf{SimTrans}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_{C})$. Below, for readability, we call the accused and accusing users as *honest* and *corrupted* users, denoted with $\mathcal{H}$ and C, respectively. As shown in Ln. 13, $\mathsf{SimPreK}$ simulates the prekey bundles of all the users. Importantly, 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 Sec. 2.5). 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.

[6] Note that we do not require the other direction: $\Pr[\mathsf{D}(\pi_{\mathsf{sim}}) = 0] \leq \mu'(\lambda) \cdot \Pr[\mathsf{D}(\pi_{\mathsf{real}}) = 0] + \delta'(\lambda)$. 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.

[7] As our definition implicitly captures outsider accusers, accusers will mean insider accusers in this section unless otherwise stated (cf. Sec. 2.2).

Next, $\mathsf{SimTrans}$ simulates the honest user's transcripts and session keys using only the accuser's secret information. Insider sender and receiver accusers are captured by Lns. 33 to 37, respectively. Outsider accusers, used for global deniability, are captured by Lns. 38 to 40, where no user secrets are provided to the simulator.

Lastly, $\mathsf{SimSt}_{\mathcal{H}}$ and SimSt_{C} simulate the honest and corrupt user's state, which may be leaked and disclosed to the distinguisher, respectively.[8] As the state is only used by the $\mathsf{BAKE.Receive}$ algorithm, $\mathsf{SimSt}_{\mathcal{H}}$ is run when simulating an honest receiver r's updated state (cf. Ln. 41). Importantly, we do not allow $\mathsf{SimSt}_{\mathcal{H}}$ to take the sender information s (and hence simulation state $\mathsf{st}_{\mathsf{Sim}}$) as input. 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. As an extreme example, consider a BAKE protocol where the updated state includes a signature on s under r's identity key; such a protocol where the updated state depends on the sender should not be considered deniable. On the other hand, we allow $\mathsf{SimSt}_{\mathcal{H}}$ to take the identity secret key isk_r as input since, by definition of the $\mathsf{BAKE.PreKeyBundleGen}$ algorithm, the state can depend on isk_r. Finally, we consider a one-shot simulation for the corrupted users, where SimSt_{C} outputs the initial and updated states (cf. Ln. 18). As SimSt_{C} can take the simulation state $\mathsf{st}_{\mathsf{Sim}}$ as input, which can record all the oracle queries to O_{ATK}, this is without loss of generality.

Formally, we define local and global deniability as follows.

Definition 4 (Local deniability)**.** Let $\mathcal{N} := [N]$ for $N \in \mathbb{N}$ be the set of all users, $\mathcal{H} \subseteq \mathcal{N}$ such that $\mathcal{H} \neq \emptyset$ be the set of accused users, and $C := \mathcal{N} \backslash \mathcal{H}$ be the set of accusers. Let $Q = \mathsf{poly}(\lambda)$ 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 $\mathcal{L}_{\mathsf{leak}}$ and disclosure function $\mathcal{D}_{\mathsf{disc}}$ with $\mathsf{leak}, \mathsf{disc} \in \{\mathsf{low}, \mathsf{med}, \mathsf{high}\}$, if there exists an efficient simulator $\mathsf{Sim} = (\mathsf{SimPreK}, \mathsf{SimTrans}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_{C})$ such that for any efficient distinguisher D we have

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

where $\mathsf{Game}^{\mathsf{local}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}},\mathcal{D}_{\mathsf{disc}}}$ is given in Alg. 1.

Definition 5 (Global deniability)**.** We define *global deniability* identically to local deniability in Def. 4 except that the distinguisher D plays the game $\mathsf{Game}^{\mathsf{global}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}},\mathcal{D}_{\mathsf{disc}}}$ in Alg. 1.

Above, 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., $Q = 2^{64}$).

[8] Put differently, we can ignore these simulators if $\mathsf{leak} \in \{\mathsf{low}, \mathsf{med}\}$ or $\mathsf{disc} = \mathsf{low}$ (cf. Sec. 2.3).

Algorithm 1 Games for local and global deniability with respect to leakage function $\mathcal{L}_{\mathsf{leak}}$, and disclosure function $\mathcal{D}_{\mathsf{disc}}$. Boxed text is only relevant to global deniability.

function $\mathsf{Game}^{\mathsf{ATK}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}},\mathcal{D}_{\mathsf{disc}}}(1^\lambda, \mathtt{mode})$
  $\mathcal{C} := \mathcal{N} \setminus \mathcal{H}$
  for user $u \in \mathcal{N}$ **do**
    $(\mathsf{ik}_u, \mathsf{isk}_u) \xleftarrow{\$} \mathsf{BAKE.IdKeyGen}(1^\lambda)$
    $\mathsf{counter}_u := 0$ ▷ *Track how many prekey bundles are used*
    if $[\![u \in \mathcal{H}]\!]$ **then**
      $(\vec{\mathsf{prek}}_u, \mathsf{st}_u) \xleftarrow{\$} \mathsf{BAKE.PreKeyBundleGen}(\mathsf{isk}_u)$
  if $[\![\mathtt{mode} = \mathtt{real}]\!]$ **then**
    for user $u \in \mathcal{C}$ **do**
      $(\vec{\mathsf{prek}}_u, \mathsf{st}_u) \xleftarrow{\$} \mathsf{BAKE.PreKeyBundleGen}(\mathsf{isk}_u)$
      $\mathsf{st}^{\mathsf{init}}_u := \mathsf{st}_u$
  else ▷ $\mathtt{mode} = \mathtt{sim}$
    ▷ *Simulate prekey bundles of accused users* $\mathcal{H}$ ◁
    $((\vec{\mathsf{prek}}^*_u)_{u\in\mathcal{H}}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{C}}, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimPreK}\left((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in\mathcal{C}}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}\right)$
    $(\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}} \leftarrow (\vec{\mathsf{prek}}^*_u)_{u\in\mathcal{H}}$
  $\mathsf{st}_{\mathsf{D}} \xleftarrow{\$} \mathsf{D}^{\mathcal{O}_{\mathsf{ATK}}}((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in\mathcal{N}})$ ▷ D *obtains transcripts*
  if $[\![\mathtt{mode} = \mathtt{sim}]\!]$ **then**
    $(\mathsf{st}_u, \mathsf{st}^{\mathsf{init}}_u)_{u\in\mathcal{C}} \xleftarrow{\$} \mathsf{SimSt}_{\mathcal{C}}(\mathsf{st}_{\mathsf{Sim}})$ ▷ *Corrupted states*
  $\mathsf{leak}_{\mathsf{D}} := \mathcal{L}_{\mathsf{leak}}((\mathsf{isk}_u, \mathsf{st}_u)_{u\in\mathcal{H}})$
  $\mathsf{disc}_{\mathsf{D}} := \mathcal{D}_{\mathsf{disc}}((\mathsf{isk}_u, \mathsf{st}_u, \mathsf{st}^{\mathsf{init}}_u)_{u\in\mathcal{C}})$
  $\mathsf{aux}_{\mathsf{D}} := (\mathsf{leak}_{\mathsf{D}}, \mathsf{disc}_{\mathsf{D}})$
  $b \xleftarrow{\$} \mathsf{D}(\mathsf{st}_{\mathsf{D}}, \mathsf{aux}_{\mathsf{D}})$ ▷ D *outputs a bit* $b \in \{0,1\}$
  return b
function $\mathcal{O}_{\mathsf{ATK}}(s, r)$
  require $[\![(s,r) \in \mathcal{N}\times\mathcal{N}]\!] \wedge [\![s \neq r]\!] \wedge [\![(s,r) \notin \mathcal{C}\times\mathcal{C}]\!]$
  if $[\![\mathsf{ATK} = \mathsf{Local}]\!]$ **then require** $[\![(s,r) \notin \mathcal{H}\times\mathcal{H}]\!]$
  $\mathsf{counter}_r \leftarrow \mathsf{counter}_r + 1$; $t := \mathsf{counter}_r$
  if $[\![t > L]\!]$ **then** $t \leftarrow \perp$ ▷ *Use last-resort prekey bundle*
  if $[\![\mathtt{mode} = \mathtt{real}]\!]$ **then** ▷ *Run real sender and receiver*
    $(K, \rho) \xleftarrow{\$} \mathsf{BAKE.Send}(\mathsf{isk}_s, \mathsf{ik}_r, \mathsf{prek}_{r,t})$
    $(K', \mathsf{st}_r) \xleftarrow{\$} \mathsf{BAKE.Receive}(\mathsf{isk}_r, \mathsf{st}_r, \mathsf{ik}_s, t, \rho)$
  else ▷ $\mathtt{mode} = \mathtt{sim}$
    if $[\![(s,r) \in \mathcal{H}\times\mathcal{C}]\!]$ **then** ▷ *Simulate with receiver secrets*
      $(K, \rho, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimTrans}(\mathsf{isk}_r, \mathsf{st}_{\mathsf{Sim}}, (s,r,t))$
    else ▷ *Honest receiver*
      if $[\![(s,r) \in \mathcal{C}\times\mathcal{H}]\!]$ **then** ▷ *Simulate with sender secrets*
        $(K', \rho, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimTrans}(\mathsf{isk}_s, \mathsf{st}_{\mathsf{Sim}}, (s,r,t))$
      else ▷ $(s,r) \in \mathcal{H}\times\mathcal{H} \Rightarrow \mathsf{ATK} = \mathsf{Global}$
        ▷ *Simulate without any secrets* ◁
        $(K, K', \rho, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimTrans}(\perp, \mathsf{st}_{\mathsf{Sim}}, (s,r,t))$
      $\mathsf{st}_r \xleftarrow{\$} \mathsf{SimSt}_{\mathcal{H}}(\mathsf{isk}_r, \mathsf{st}_r)$ ▷ *Update honest receiver state*
  if $[\![(s,r) \in \mathcal{H}\times\mathcal{C}]\!]$ **then return** (K, ρ)
  else if $[\![(s,r) \in \mathcal{C}\times\mathcal{H}]\!]$ **then return** (K', ρ)
  else return (K, K', ρ) ▷ $(s,r) \in \mathcal{H}\times\mathcal{H} \Rightarrow \mathsf{ATK} = \mathsf{Global}$

Remark 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* $\mathcal{O}_{\mathsf{ATK}}$ *only on inputs* (s,r)*, where* $r \in \mathcal{H}$*. 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.*

3.3 Deniability Against Malicious Accusers

We define *strong* local and global deniability, that is deniability against malicious accusers, similarly as above except that we explicitly consider an accuser algorithm $\mathcal{A}$ in the real world (cf. Alg. 2, Ln. 9, and Alg. 3, Ln. 11). Note that in Alg. 2, Ln. 9 we no longer assume prekey bundles are honestly generated. Moreover, the simulator $\mathsf{Sim}_{\mathcal{A}}$ can depend on the accuser $\mathcal{A}$ — this is sufficient as an accused user can plea that an accuser was internally running $\mathsf{Sim}_{\mathcal{A}}$.

We also make a subtle yet crucial design choice. Specifically, we only consider deniability of *successful* BAKE executions. If a malicious sender sends a handshake message ρ which is not accepted (i.e., the honest receiver outputs $\perp$), then we do not require deniability (cf. Lns. 19 to 24). This is because an accused user only needs to deny having sent a message to the sender using the established session key; if no keys were established, there is no need to deny. This allows us to get around an artificial theoretical obstacle in the proof: the simulator $\mathsf{SimTrans}_{\mathcal{A}}$ no longer needs to know if ρ output by the malicious sender is accepted, which it cannot know without the receiver's secret keys. Concretely, we allow $\mathsf{SimTrans}_{\mathcal{A}}$ to output a simulated session key K'^*, *conditioned on* the receiver accepting — if the receiver rejects, then K'^* is simply discarded by the game. Lastly, since $\mathsf{SimSt}_{\mathcal{H}}$ needs to simulate the honest receiver's updated state, the game roles back the updated state after executing BAKE.Receive (cf. Lns. 20 and 21).

We define strong local and global deniability as follows.

Definition 6 (Strong local deniability)**.** Let $N, \mathcal{H}, C, Q$ be defined as in Def. 4. A BAKE protocol is (μ,δ)-*strongly local deniable* against *malicious* accusers with respect to leakage function $\mathcal{L}_{\mathsf{leak}}$ with $\mathsf{leak} \in \{\mathsf{low}, \mathsf{med}, \mathsf{high}\}$, if for any efficient accuser $\mathcal{A}$, there exists an efficient simulator $\mathsf{Sim}_{\mathcal{A}} = (\mathsf{SimPreK}_{\mathcal{A}}, \mathsf{SimTrans}_{\mathcal{A}}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_{\mathcal{A}})$ such that for any efficient distinguisher D we have

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

where $\mathsf{Game}^{\text{strong-local}}_{\mathcal{A},\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}}}$ is given in Alg. 2.

Definition 7 (Strong global deniability)**.** We define *strong global* deniability identically to strong local deniability in Def. 6, except that the distinguisher D plays the game $\mathsf{Game}^{\text{strong-global}}_{\mathcal{A},\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}}}$ in Alg. 2.

4 Deniability of X3DH and PQXDH

X3DH is the original handshake protocol used by Signal [38], based on Triple Diffie–Hellman [34]. In 2023, Signal rolled

Algorithm 2 Games for strong local and global deniability with respect to a leakage function $\mathcal{L}_{\mathsf{leak}}$. Shaded boxes highlight differences compared to standard deniability.

1: **function** $\mathsf{Game}^{\text{strong-ATK}}_{\mathcal{A},\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}}}(1^\lambda, \mathtt{mode})$
2: $C \coloneqq [N] \setminus \mathcal{H}$
3: **for** user $u \in [N]$ **do**
4: $(\mathsf{ik}_u, \mathsf{isk}_u) \xleftarrow{\$} \mathsf{BAKE.IdKeyGen}(1^\lambda)$
5: $\mathsf{counter}_u \coloneqq 0$ ▷ *Keep track of num prekey bundles used*
6: **if** $[\![u \in \mathcal{H}]\!]$ **then**
7: $(\vec{\mathsf{prek}}_u, \mathsf{st}_u) \xleftarrow{\$} \mathsf{BAKE.PreKeyBundleGen}(\mathsf{isk}_u)$
8: **if** $[\![\mathtt{mode} = \mathtt{real}]\!]$ **then** ▷ *Adversarial prekey bundles*
9: $((\vec{\mathsf{prek}}_u)_{u\in C}, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}((\mathsf{ik}_u)_{u\in[N]}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}})$
10: **if** $[\![\mathtt{mode} = \mathtt{sim}]\!]$ **then**
11: $((\vec{\mathsf{prek}}^*_u)_{u\in\mathcal{H}}, (\vec{\mathsf{prek}}_u)_{u\in C}, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimPreK}_{\mathcal{A}}((\mathsf{ik}_u)_{u\in[N]}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}})$
12: $(\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}} \leftarrow (\vec{\mathsf{prek}}^*_u)_{u\in\mathcal{H}}$ ▷ *Simulate for accused $\mathcal{H}$*
13: ▷ *D obtains transcripts exchanged between users* ◁
14: $\mathsf{st}_{\mathsf{D}} \xleftarrow{\$} \mathsf{D}^{\mathcal{O}_{\mathsf{ATK}}}((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in[N]})$ ▷ *cf. Alg. 3*
15: **if** $[\![\mathtt{mode} = \mathtt{sim}]\!]$ **then**
16: $\mathsf{st}_{\mathcal{A}} \xleftarrow{\$} \mathsf{SimSt}_{\mathcal{A}}(\mathsf{st}_{\mathsf{Sim}})$ ▷ *Simulate adversary state*
17: $\mathsf{leak}_{\mathsf{D}} \coloneqq \mathcal{L}_{\mathsf{leak}}((\mathsf{isk}_u, \mathsf{st}_u)_{u\in\mathcal{H}})$
18: $\mathsf{aux}_{\mathsf{D}} \coloneqq (\mathsf{leak}_{\mathsf{D}}, \mathsf{st}_{\mathcal{A}})$
19: ▷ *D outputs $b \in \{0,1\}$ after obtaining leakage $\mathsf{aux}_{\mathsf{D}}$* ◁
20: $b \xleftarrow{\$} \mathsf{D}(\mathsf{st}_{\mathsf{D}}, \mathsf{aux}_{\mathsf{D}})$
21: **return** b

Algorithm 3 The oracle for strong local and global deniability games used in Alg. 2. Shaded boxes highlight differences compared to standard deniability. Boxed text is only relevant to global deniability. In Ln. 21, "_" indicates that $\mathsf{st}_{\mathsf{tmp}}$ is deleted and no longer used by the game.

1: **function** $\mathcal{O}_{\mathsf{ATK}}(s, r)$
2: **require** $[\![(s,r) \in [N]\times[N]]\!] \wedge [\![s \neq r]\!] \wedge [\![(s,r) \notin C \times C]\!]$
3: **if** $[\![\mathsf{ATK} = \mathsf{Local}]\!]$ **then require** $[\![(s,r) \notin \mathcal{H}\times\mathcal{H}]\!]$
4: $\mathsf{counter}_r \leftarrow \mathsf{counter}_r + 1$
5: $t \coloneqq \mathsf{counter}_r$
6: **if** $[\![t > L]\!]$ **then** $t \leftarrow \perp$ ▷ *Use last-resort prekey bundle*
7: **if** $[\![\mathtt{mode} = \mathtt{real}]\!]$ **then**
8: **if** $[\![s \in \mathcal{H}]\!]$ **then** ▷ *Run sender*
9: $(K, \rho) \xleftarrow{\$} \mathsf{BAKE.Send}(\mathsf{isk}_s, \mathsf{ik}_r, \mathsf{prek}_{r,t})$
10: **else** ▷ *$s \in C$, malicious sender*
11: $(\rho, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}(\mathsf{isk}_s, \mathsf{st}_{\mathcal{A}}, (s, r, t))$
12: **if** $[\![r \in \mathcal{H}]\!]$ **then** ▷ *Run receiver*
13: $(K', \mathsf{st}_r) \xleftarrow{\$} \mathsf{BAKE.Receive}(\mathsf{isk}_r, \mathsf{st}_r, (\mathsf{ik}_s, t, \rho))$
14: **else** ▷ `mode = sim`
15: **if** $[\![(s,r) \in \mathcal{H}\times C]\!]$ **then** ▷ *Simulate with receiver secrets*
16: $(K, \rho, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimTrans}_{\mathcal{A}}(\mathsf{isk}_r, \mathsf{st}_{\mathsf{Sim}}, (s, r, t))$
17: **else** ▷ *Honest receiver*
18: **if** $[\![(s,r) \in C\times\mathcal{H}]\!]$ **then** ▷ *Simulate with sender secrets*
19: $(K'^*, \rho, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimTrans}_{\mathcal{A}}(\mathsf{isk}_s, \mathsf{st}_{\mathsf{Sim}}, (s, r, t))$
20: $\mathsf{st}_{\mathsf{tmp}} \coloneqq \mathsf{st}_r$ ▷ *Copy state*
21: $(K', \_) \xleftarrow{\$} \mathsf{BAKE.Receive}(\mathsf{isk}_r, \mathsf{st}_{\mathsf{tmp}}, (\mathsf{ik}_s, t, \rho))$
22: ▷ *Replace with simulated key* if *receiver accepts* ◁
23: **if** $[\![K' \neq \perp]\!]$ **then**
24: $K' \leftarrow K'^*$
25: **else** ▷ $(s,r) \in \mathcal{H}\times\mathcal{H} \Rightarrow \mathsf{ATK} = \mathsf{Global}$
26: ▷ *Simulate without any secrets* ◁
27: $(K, K', \rho, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimTrans}_{\mathcal{A}}(\mathsf{st}_{\mathsf{Sim}}, (s, r, t))$
28: $\mathsf{st}_r \leftarrow \mathsf{SimSt}_{\mathcal{H}}(\mathsf{isk}_r, \mathsf{st}_r)$ ▷ *Update honest receiver state*
29: **if** $[\![(s,r) \in \mathcal{H}\times C]\!]$ **then return** (K, ρ)
30: **else if** $[\![(s,r) \in C\times\mathcal{H}]\!]$ **then return** (K', ρ)
31: **else return** (K, K', ρ) ▷ $(s,r) \in \mathcal{H}\times\mathcal{H} \Rightarrow \mathsf{ATK} = \mathsf{Global}$

out a variant PQXDH [38], which adds a post-quantum KEM key exchange to X3DH, securing against harvest-now, decrypt-later (HNDL) adversaries. In this section, we examine the deniability of X3DH and PQXDH.

4.1 The X3DH and PQXDH Protocols

In Alg. 4, we give the identity key generation algorithm and the algorithm that is used for prekey bundle generation in X3DH and PQXDH as described by Hashimoto et al. [29]. PQXDH is a strict extension to X3DH; functionality that is exclusive to PQXDH is marked with a gray dotted box. The classic key agreement for both consists of Diffie–Hellman (DH) with combinations of the sender and receiver's long-term identity keys, the receiver's prekey bundle's signed DH prekey, and, if not using a last-resort prekey bundle, one-time DH prekey, and an ephemeral DH key generated by the sender (cf. Lns. 7 to 12). PQXDH simply adds a KEM key exchange to X3DH, as shown in Ln. 13.

The description of the Send algorithm for X3DH and PQXDH is given in Alg. 5, where PQXDH-specific functionality is marked with a gray dotted box.

Note that we follow the description given by [29, Section 4], which adds a confirmation tag in lieu of modeling the sending of an encrypted message as in Signal's documentation. In practice, Signal uses the Double Ratchet algorithm to send this message; internally, it contains a MAC that achieves the same functionality. Hashimoto et al. write that Signal is considering similar changes to add better separation between the PQXDH and Double Ratchet protocols. For a detailed comparison between Alg. 5, the Signal documentation, and Signal's implementation, refer to [29, Appendix C].

Algorithm 6 shows the algorithm used by receivers in X3DH and PQXDH. The key computation is analogous to that in the Send algorithm given by Alg. 5; however, to prevent replay attacks, the Receive algorithm additionally records a list of messages received. For details about this protection measure, refer to Hashimoto et al. [29, Sec. 4].

4.2 Summary: Deniability of X3DH & PQXDH

We summarize the levels of deniability satisfied by X3DH and PQXDH. See Tab. 1 for a complete overview, and recall that

Algorithm 4 X3DH and PQXDH identity key and prekey bundle generation algorithms [29]. PQXDH-exclusive code is marked like this.

function PQX3DH.IdKeyGen(1^λ)
  $\overline{\mathsf{isk}} \xleftarrow{\$} \mathbb{Z}_p$; $\overline{\mathsf{ik}} := [\overline{\mathsf{isk}}]G$
  $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}(1^\lambda)$
  return $(\mathsf{ik} := (\overline{\mathsf{ik}}, \mathsf{vk}), \mathsf{isk} := (\overline{\mathsf{isk}}, \mathsf{sk}))$

function PQX3DH.PreKeyBundleGen(isk_u)
  $(\overline{\mathsf{isk}}_u, \mathsf{sk}_u) \leftarrow \mathsf{isk}_u$
  $D_{\mathsf{prek}}, D_{\rho_\perp} := \emptyset$ ▷ *Initialize empty lists*
  ▷ *Generate what Signal calls the* signed prekey ◁
  $\mathsf{spksec}_u \xleftarrow{\$} \mathbb{Z}_p$; $\mathsf{spk}_u := [\mathsf{spksec}]G$
  $\sigma_{\mathsf{spk}_u} \xleftarrow{\$} \mathsf{Sig.Sign}(\mathsf{sk}_u, \mathsf{spk}_u)$
  ▷ *Create the* L one-time *prekey bundles* ◁
  for $t \in [L]$ **do**
    $\mathsf{osk}_{u,t} \xleftarrow{\$} \mathbb{Z}_p$; $\mathsf{opk}_{u,t} := [\mathsf{osk}_{u,t}]G$
    $(\mathsf{ek}_{u,t}, \mathsf{dk}_{u,t}) \xleftarrow{\$} \mathsf{KEM.KeyGen}(1^\lambda)$
    $\sigma_{\mathsf{ek}_{u,t}} \xleftarrow{\$} \mathsf{Sig.Sign}(\mathsf{sk}_u, \mathsf{ek}_{u,t})$
    $\mathsf{prek}_{u,t} := (\mathsf{spk}_u, \sigma_{\mathsf{spk}_u}, \mathsf{opk}_{u,t}, \mathsf{ek}_{u,t}, \sigma_{\mathsf{ek}_{u,t}})$
    $D_{\mathsf{prek}}[t] \leftarrow (\mathsf{prek}_{u,t}, (\mathsf{spksec}_u, \mathsf{osk}_{u,t}, \mathsf{dk}_{u,t}))$
  ▷ *Set up the* last-resort *prekey bundle* ◁
  $(\mathsf{ek}_{u,\perp}, \mathsf{dk}_{u,\perp}) \xleftarrow{\$} \mathsf{KEM.KeyGen}(1^\lambda)$
  $\sigma_{\mathsf{ek}_{u,\perp}} \xleftarrow{\$} \mathsf{Sig.Sign}(\mathsf{sk}_u, \mathsf{ek}_{u,\perp})$
  $\mathsf{prek}_{u,\perp} := (\mathsf{spk}_u, \sigma_{\mathsf{spk}_u}, \perp, \mathsf{ek}_{u,\perp}, \sigma_{\mathsf{ek}_{u,\perp}})$
  $D_{\mathsf{prek}}[L+1] \leftarrow (\mathsf{prek}_{u,\perp}, (\mathsf{spksec}_u, \perp, \mathsf{dk}_{u,\perp}))$
  return $(\vec{\mathsf{prek}}_u, \mathsf{st}_u := (D_{\mathsf{prek}}, D_{\rho_\perp}))$

Algorithm 5 The PQXDH and X3DH.Send algorithm [29].

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

the level of deniability can be sorted based on the leakage function $\mathcal{L}_{\mathsf{leak}}$ and disclosure function $\mathcal{D}_{\mathsf{disc}}$, where $(\mathsf{leak}, \mathsf{disc})$ dictates the amount of secret information given to the distinguisher (cf. Sec. 2.3). For the formal statements, see App. F in the full version.

Local deniability. Both protocols achieve the highest level of local deniability (i.e., $(\mathsf{leak}, \mathsf{disc}) = (\mathsf{high}, \mathsf{high})$). 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).

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., $(\mathsf{leak}, \mathsf{disc}) = (\mathsf{high}, \mathsf{high})$). However, if the accused user used their last-resort prekey bundle, it can only support global deniability with $(\mathsf{leak}, \mathsf{disc}) = (\mathsf{med}, \mathsf{high})$, 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. 5, Ln. 12). 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. 6, Lns. 12 and 13). Hence, accused users can only leak their identity secret key (i.e., $\mathsf{leak} = \mathsf{med}$).

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.[9] We show deniability by the session key K being KDF-derived from KEM key $\mathsf{ss}_{\mathsf{KEM}}$ (cf. Alg. 5, Ln. 14). 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) [49] to show strong (local and global) deniability. Knowledge assumptions assume the existence of knowledge extractors, for which there exist no concrete constructions. The resulting simulators are hence not efficiently implementable, which violates the requirements of [43], requir-

[9] Against classical D, global deniability of PQXDH follows from X3DH.

Algorithm 6 The PQXDH and X3DH.Receive algorithm [29].

function PQX3DH.Receive($\mathsf{isk}_r, \mathsf{st}_r, \mathsf{ik}_s, t, \rho$)
  $(\overline{\mathsf{isk}}_r, \mathsf{sk}_r) \leftarrow \mathsf{isk}_r$; $(\overline{\mathsf{ik}}_s, \mathsf{vk}_s) \leftarrow \mathsf{ik}_s$
  $(D_{\mathsf{prek}}, D_{\rho_\perp}) \leftarrow \mathsf{st}_r$
  if $[\![t \neq \perp]\!]$ **then** ▷ *One-time prekey bundle*
    require $[\![D_{\mathsf{prek}}[t] \neq \perp]\!]$ ▷ *Check if unused.*
    $(\mathsf{prek}_{r,t}, (\mathsf{spksec}_r, \mathsf{osk}_{r,t}, \mathsf{dk}_{r,t})) \leftarrow D_{\mathsf{prek}}[t]$
  else ▷ *Last-resort prekey bundle (i.e., $t = \perp$)*
    require $[\![\rho \notin D_{\rho_\perp}]\!]$ ▷ *Check ρ is not replayed.*
    $D_{\rho_\perp} \leftarrow D_{\rho_\perp} \cup \{\rho\}$
    $(\mathsf{prek}_{r,t}, (\mathsf{spksec}_r, \perp, \mathsf{dk}_{r,t})) \leftarrow D_{\mathsf{prek}}[t]$
  $(\mathsf{epk}, \mathsf{ct}, \tau_{\mathsf{conf}}) \leftarrow \rho$
  $\mathsf{ss}_1 \coloneqq [\mathsf{spksec}_r]\overline{\mathsf{ik}}_s$; $\mathsf{ss}_2 \coloneqq [\overline{\mathsf{isk}}_r]\mathsf{epk}$
  $\mathsf{ss}_3 \coloneqq [\mathsf{spksec}_r]\mathsf{epk}$; $\mathsf{ss} \coloneqq \mathsf{ss}_1 \| \mathsf{ss}_2 \| \mathsf{ss}_3$
  if $[\![t \neq \perp]\!]$ **then** ▷ *One-time prekey bundle*
    $\mathsf{ss}_4 \coloneqq [\mathsf{osk}_{r,t}]\mathsf{epk}$; $\mathsf{ss} \coloneqq \mathsf{ss}_1 \| \mathsf{ss}_2 \| \mathsf{ss}_3 \| \mathsf{ss}_4$
  $\mathsf{ss}_{\mathsf{KEM}} \leftarrow \mathsf{KEM.Decaps}(\mathsf{dk}_{r,t}, \mathsf{ct})$
  $\mathsf{content} \coloneqq \mathsf{ik}_s \| \mathsf{ik}_r \| \mathsf{prek}_{r,t} \| \mathsf{epk} \| \mathsf{ct}$
  $K \| \tau'_{\mathsf{conf}} \coloneqq \mathsf{KDF}(\mathsf{ss} \| \mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$
  require $[\![\tau_{\mathsf{conf}} = \tau'_{\mathsf{conf}}]\!]$
  ▷ *Delete prekey bundle if not last-resort* ◁
  if $[\![t \neq \perp]\!]$ **then** $D_{\mathsf{prek}}[t] \leftarrow \perp$
  $\mathsf{st}_r \leftarrow (D_{\mathsf{prek}}, D_{\rho_\perp})$
  return (K, st_r)

ing accusers the ability to show the existence of such simulator to the judge in practice. 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 [10], 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 [10], or extending GGM to work over non-prime groups. We leave these considerations for future work.

Similarly, we show strong local and global deniability of PQXDH against a classical accuser and a *classical* distinguisher. However, interestingly, we are not able to prove strong deniability against a *quantum* distinguisher. This is caused by our reliance on the GGM, a model that only assumes generic adversaries; the proof cannot handle a classical accuser in the GGM, while considering a quantum distinguisher performing non-generic operations (i.e., breaking DL using Shor's algorithm). We thus leave it as an interesting open problem to show HNJL deniability of PQXDH.[10]

Remark 2. *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* $\mathsf{opk}_{r,t}$ *and vice versa. This does not harm our deniability results for PQXDH. As this follows from the same argument as above, only with more case distinctions, we refer the interested reader to App. F.5 in the full version.*

5 Deniability of RingXKEM

RingXKEM is a PQ Signal handshake protocol by Hashimoto et al. [29], using ring signatures to simultaneously ensure authentication and deniability. This is an optimized protocol based on [8, 26, 27] where receiver bandwidth and storage requirements on the server are reduced by using Merkle trees for the authentication of prekey bundles.

5.1 Deniable Ring Signatures

We first recall the syntax of ring signatures. Standard notions of correctness and unforgeability are provided in App. A.4 in the full version.

Definition 8 (Ring Signatures)**.** A ring signature (RS) scheme consists of three PPT algorithms:

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

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

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

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. Sec. 3.1). 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 9 (Deniability)**.** Let $\mu : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}^+$ be a positive-valued function and $Q = \mathsf{poly}(\lambda)$ an upper bound on the number of signing queries. A ring signature scheme is (μ, δ)-*deniable* if for any $N = \mathsf{poly}(\lambda)$, and efficient adversary $\mathcal{A}$,

[10] We can bypass this issue by simply relying on a strong knowledge type assumption allowing for a knowledge extractor. However, as mentioned above, this does not give any concrete means to construct a simulator in the real world, rendering the result useless in practice.

Algorithm 7 Game for the deniability of RS.

function $\mathsf{Game}^{\mathsf{Deny}}_{\mathsf{RS},b,\mathcal{A}}(1^\lambda, Q)$
  for user $u \in [N]$ **do** $(\mathsf{rvk}_u, \mathsf{rsk}_u) \xleftarrow{\$} \mathsf{RS.KeyGen}(1^\lambda)$
  $q := 0$ ▷ *Number of queries made*
  return $b \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{RS.Sign}}(\cdot,\cdot,\cdot)}((\mathsf{rvk}_u, \mathsf{rsk}_u)_{u\in[N]})$
function $\mathcal{O}_{\mathsf{RS.Sign}}(\mathsf{M}, u_0, u_1)$
  require $[\![(u_0, u_1) \in [N] \times [N]]\!] \wedge [\![q < Q]\!]$
  $q \leftarrow q + 1$
  return $\sigma \xleftarrow{\$} \mathsf{RS.Sign}(\mathsf{rsk}_{u_b}, \mathsf{M}, \{\mathsf{rvk}_{u_0}, \mathsf{rvk}_{u_1}\})$

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

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

As alluded to in Sec. 3.1, 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 [47] is deniable with a multiplicative slack $\mu = 1 + 2^{-27}$; if we used 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 up 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.

5.2 The RingXKEM protocol

The RingXKEM protocol, ignoring the Merkle tree optimization, can be summarized as a key exchange using an ephemeral KEM key for forward secrecy, a long-term KEM identity key to authenticate the receiver, and an RS to authenticate the sender. Each user generates a pair of RS keys $(\mathsf{rvk}, \mathsf{rsk})$ as part of their identity key. An RS verification key $\widehat{\mathsf{rvk}}$ is also added to the prekey bundles, however the associated secret $\widehat{\mathsf{rsk}}$ is never used, and immediately disposed of after generation. In contrast to other elements in the prekey bundle, $\widehat{\mathsf{rvk}}$ is not authenticated for deniability. The sender authenticates by signing the handshake message for the ring of rvk_s and $\widehat{\mathsf{rvk}}_r$.

Signing each prekey bundle for authentication would be costly, due to the size of PQ signatures. Hashimoto et al. [29] thus proposed an optimization, where a Merkle tree of KEM public keys $\widehat{\mathsf{dk}}_{u,t}$ replaces individual signatures with a path to the root and a single signature on the root. This amortizes storage costs across all prekey bundles uploaded by a single call to PreKeyBundleGen. Note that although we include the Merkle tree path in our description of prekey bundles, the server can recompute the path and tree root from uploaded bundles, eliminating the need to store them.

In Alg. 8 we show the details for the identity key generation and prekey bundle generation algorithms for RingXKEM. Any functionality exclusive to either algorithm is marked by grey dotted boxes for RingXKEM and by plain dash-dotted boxes for SignXKEM.

Algorithm 8 RingSignXKEM's identity key and prekey bundle generation algorithms.

function RingSignXKEM.IdKeyGen(1^λ)
  $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{KEM.KeyGen}(1^\lambda)$
  $(\mathsf{rvk}, \mathsf{rsk}) \xleftarrow{\$} \mathsf{RSSign.KeyGen}(1^\lambda)$
  return $(\mathsf{ik} := (\mathsf{ek}, \mathsf{rvk}), \mathsf{isk} := (\mathsf{dk}, \mathsf{rsk}))$
function RingSignXKEM.PreKeyBundleGen(isk_u)
  $(\mathsf{dk}_u, \mathsf{rsk}_u) \leftarrow \mathsf{isk}_u$
  $D_{\mathsf{kem}}, D_{\rho_\perp} := \emptyset$ ▷ *Initialize empty lists*
  for $t \in [L] \cup \{\perp\}$ **do**
    $(\widehat{\mathsf{ek}}_{u,t}, \widehat{\mathsf{dk}}_{u,t}) \xleftarrow{\$} \mathsf{KEM.KeyGen}(1^\lambda)$
  ▷ *Create and sign Merkle tree* ◁
  $(\mathsf{root}_u, \mathsf{tree}_u) \leftarrow \mathsf{MerkleTree}((\widehat{\mathsf{ek}}_{u,t})_{t\in[L]\cup\{\perp\}})$
  $\sigma_{u,\mathsf{root}} \xleftarrow{\$} \mathsf{RSSign.Sign}(\mathsf{rsk}_u, \mathsf{root}_u, \{\mathsf{rvk}_u\})$
  $(\widehat{\mathsf{rvk}}_u, \_) \xleftarrow{\$} \mathsf{RS.KeyGen}(1^\lambda)$ ▷ *Discard* rsk
  for $t \in [L]$ **do** ▷ *One-time prekey bundles*
    $\mathsf{path}_{u,t} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}_u, t)$
    $\mathsf{prek}_{u,t} := (\widehat{\mathsf{ek}}_{u,t}, \mathsf{path}_{u,t}, \mathsf{root}_u, \sigma_{u,\mathsf{root}}, \widehat{\mathsf{rvk}}_u)$
    $D_{\mathsf{kem}}[t] \leftarrow (\mathsf{prek}_{u,t}, \widehat{\mathsf{dk}}_{u,t})$
  ▷ *Last-resort prekey bundle* $t = \perp$ ◁
  $\mathsf{path}_{u,\perp} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}_u, L+1)$
  $\mathsf{prek}_{u,\perp} := (\widehat{\mathsf{ek}}_{u,\perp}, \mathsf{path}_{u,\perp}, \mathsf{root}_u, \sigma_{u,\mathsf{root}}, \widehat{\mathsf{rvk}}_u)$
  $D_{\mathsf{kem}}[L+1] \leftarrow (\mathsf{prek}_{u,\perp}, \widehat{\mathsf{dk}}_{u,\perp})$
  return $\begin{pmatrix} \vec{\mathsf{prek}}_u := (\mathsf{prek}_{u,t})_{t\in[L]\cup\{\perp\}}, \\ \mathsf{st}_u := (D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_u, D_{\rho_\perp}) \end{pmatrix}$

The algorithm defining the RingXKEM Send algorithm, which previously appeared in [29], is given in Alg. 9.

In Alg. 10, we provide the details for the Receive algorithm. Again, the algorithm records a list of messages received to prevent replay attacks.

5.3 Summary: Deniability of RingXKEM

We summarize the level of deniability that RingXKEM satisfies. See Tab. 1 for a complete overview. The formal statements are provided in App. H in the full version.

Local and global deniability. RingXKEM achieves the highest level of local and global deniability. For 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 $\widehat{\mathsf{rvk}}_r$) was used to sign, so that either the sender or the receiver could have produced it.

Algorithm 9 The RingSignXKEM.Send algorithm

function RingSignXKEM.Send($\mathsf{isk}_s, \mathsf{ik}_r, \mathsf{prek}_r$)
  $(\mathsf{dk}_s, \mathsf{rsk}_s) \leftarrow \mathsf{isk}_s$; $(\mathsf{ek}_r, \mathsf{rvk}_r) \leftarrow \mathsf{ik}_r$
  $(\widehat{\mathsf{ek}}_r, \mathsf{path}_r, \mathsf{root}_r, \sigma_{r,\mathsf{root}}, \widehat{\mathsf{rvk}}_r) \leftarrow \mathsf{prek}_r$

  require $[\![\mathsf{ReconstructRoot}(\widehat{\mathsf{ek}}_r, \mathsf{path}_r) = \mathsf{root}_r]\!]$
  require $[\![\mathsf{RS.Sign.Verify}(\mathsf{vk}_r, \{\mathsf{rvk}_r\}, \widehat{\mathsf{ek}}_r, \sigma_{r,\mathsf{root}}) = 1]\!]$
  $(\mathsf{ss}_r, \mathsf{ct}_r) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek}_r)$
  $(\widehat{\mathsf{ss}}_r, \widehat{\mathsf{ct}}_r) \xleftarrow{\$} \mathsf{KEM.Encaps}(\widehat{\mathsf{ek}}_{r,t})$
  $\mathsf{context} \coloneqq \mathsf{ik}_s \| \mathsf{ik}_r \| \mathsf{prek}_{r,t} \| \mathsf{ct}_r \| \widehat{\mathsf{ct}}_r$
  $K \| K_{\mathsf{ske}} \coloneqq \mathsf{KDF}(\mathsf{ss}_r \| \widehat{\mathsf{ss}}_r, \mathsf{context})$
  $\sigma_s \xleftarrow{\$} \mathsf{RS.Sign.Sign}(\mathsf{rsk}_s, \mathsf{context}, \{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\})$
  $\mathsf{ct}_{\mathsf{ske}} \xleftarrow{\$} \mathsf{SKE.Enc}(K_{\mathsf{ske}}, \sigma_s)$ ▷ *Mask signature*
  $\rho \coloneqq (\mathsf{ct}_r, \widehat{\mathsf{ct}}_r, \mathsf{ct}_{\mathsf{ske}})$
  return (K, ρ)

Algorithm 10 The RingSignXKEM.Receive algorithm

function RingSignXKEM.Receive($\mathsf{isk}_r, \mathsf{st}_r, \mathsf{ik}_s, t, \rho$)
  $(\mathsf{dk}_r, \mathsf{rsk}_r) \leftarrow \mathsf{isk}_r$; $(\mathsf{ek}_s, \mathsf{rvk}_s) \leftarrow \mathsf{ik}_s$
  $(D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_r, D_{\rho_\perp}) \leftarrow \mathsf{st}_r$
  $(\mathsf{ct}_r, \widehat{\mathsf{ct}}_r, \mathsf{ct}_{\mathsf{ske}}) \leftarrow \rho$
  ▷ *Check t^{th} prekey bundle was not deleted.* ◁
  require $[\![D_{\mathsf{kem}}[t] \neq \perp]\!]$
  if $[\![t = \perp]\!]$ **then**
    require $[\![(\mathsf{ct}_r, \widehat{\mathsf{ct}}_r) \notin D_{\rho_\perp}]\!]$ ▷ *Prevent replays*
    $D_{\rho_\perp} \leftarrow D_{\rho_\perp} \cup \{(\mathsf{ct}_r, \widehat{\mathsf{ct}}_r)\}$
  $(\mathsf{prek}_{r,t}, \widehat{\mathsf{dk}}_{r,t}) \leftarrow D_{\mathsf{kem}}[t]$
  $\mathsf{ss}_r \coloneqq \mathsf{KEM.Decaps}(\mathsf{dk}_r, \mathsf{ct}_r)$
  $\widehat{\mathsf{ss}}_r \coloneqq \mathsf{KEM.Decaps}(\widehat{\mathsf{dk}}_{r,t}, \widehat{\mathsf{ct}}_r)$
  $\mathsf{content} \coloneqq \mathsf{ik}_s \| \mathsf{ik}_r \| \mathsf{prek}_{r,t} \| \mathsf{ct}_r \| \widehat{\mathsf{ct}}_r$
  $K \| K_{\mathsf{ske}} \coloneqq \mathsf{KDF}(\mathsf{ss}_r \| \widehat{\mathsf{ss}}_r, \mathsf{content})$
  $\sigma_s \coloneqq \mathsf{SKE.Dec}(K_{\mathsf{ske}}, \mathsf{ct}_{\mathsf{ske}})$ ▷ *Unmask signature*
  require $[\![\mathsf{RS.Sign.Verify}(\mathsf{vk}_s, \{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\}, \mathsf{content}, \sigma_s) = 1]\!]$
  if $[\![t \neq \perp]\!]$ **then**
    $D_{\mathsf{kem}}[t] \leftarrow \perp$ ▷ *Delete prekey bundle*
  $\mathsf{st}_r \leftarrow (D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_r, D_{\rho_\perp})$
  return (K, st_r)

We further have global deniability, thanks to the RS key $\widehat{\mathsf{rvk}}$ in the prekey bundle not being authenticated. Specifically, the simulator, given prekey bundles of honest users, can substitute the verification key $\widehat{\mathsf{rvk}}$ 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. 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*; see App. H.3.1 in the full version for a discussion on why the deniability of the *sender* breaks (see also Rem. 1). 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 in Sec. 3.3, that is, the simulator need only simulate when the (honest) receiver accepts. Indeed, for the receiver to accept, the KDF, modeled as a random oracle, must have output keys $K \| K_{\mathsf{ske}}$ for which the ciphertext $\mathsf{ct}_{\mathsf{ske}}$ in the (possibly maliciously generated) handshake message ρ decrypts correctly, yielding a valid signature of content for the ring $\{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\}$. 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.

5.4 Alternative BAKE from Plain Signatures

At the cost of a loss in deniability, one can use plain digital signatures instead of ring signatures. Such a scheme was suggested in [26], and benefits from being more efficient and simpler to implement. A description of the resulting scheme, which we call SignXKEM, is given in Algs. 8 to 10. The key difference from RingXKEM is that a (plain) signature is used to authenticate the sender. SignXKEM offers a limited notion of deniability, relying on the IND-CPA security of the KEM and SKE schemes to hide the sender's no-longer-deniable signature. The level of deniability is provided in Tab. 1, where the formal statements are provided in App. I in the full version. It is worth noting that SignXKEM is the only protocol presented in this work where one must restrict the information disclosed by accusers for deniability to hold, even in the (standard) local setting. Indeed, if one cannot assume secure key erasure, this protocol may not be satisfactory. As such, this protocol may not provide deniability if one is communicating with untrusted peers.

6 Ring Signatures from Falcon and MAYO

In this section, we construct two novel 2-user ring signatures achieving our deniability notion from Sec. 5.1. 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 [47], and the additional signature candidate MAYO [3]. Both of these schemes have short signatures, and thus great potential for compact ring signatures.

6.1 Falcon-Based Ring Signature

This section provides a high-level construction for our Falcon-based ring signature. Formal definitions are given in App. J.1 in the full version.

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 $\mathsf{TpdGen}()$ 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 $c = H(\mathsf{M}, \mathsf{salt})$ from a hash function. The trapdoor then allows for sampling a short preimage $(u, v) = \mathsf{PreSmp}(\mathbf{B}, \sigma, -c)$ 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 $c = H(\mathsf{M}, \mathsf{salt})$, and verifies the shortness of (u, v), i.e. that $\|(u, v)\| \leq \beta$.

Algorithm 11 Falcon-based ring signature scheme

function $\mathsf{FalconRS.KeyGen}(1^\lambda)$
  $h, \mathbf{B} \xleftarrow{\$} \mathsf{TpdGen}()$ ▷ *Sample lattice generator h, and trapdoor*
  return $(\mathsf{rvk} := h, \mathsf{rsk} := \mathbf{B})$
function $\mathsf{FalconRS.Sign}(\mathsf{rsk}_i, \mathsf{M}, \mathsf{RL} := \{\mathsf{rvk}_j\}_j)$
  $\mathbf{B} := \mathsf{rsk}_i$; $\{h_j\}_j := \{\mathsf{rvk}_j\}_j$
  $\mathsf{salt} \xleftarrow{\$} \{0,1\}^\kappa$; $c := H(\mathsf{salt}, \mathsf{RL}, \mathsf{M}) \in \mathcal{R}_q$
  for $j \neq i$ **do** $u_j \xleftarrow{\$} \chi_u$ ▷ *χ_u is $\mathcal{D}_{\mathbb{Z}^n,\sigma,0}$ tailcut to $[-\eta', \eta']$*
  $c' := t - \sum_{j \neq i} h_j \cdot u_j$
  $(u_i, v) := \mathsf{PreSmp}(\mathbf{B}, \sigma, -c')$ ▷ *Pre-image sampling*
  if $[\![\|((u_j)_j, v + c')\| > \beta_{\mathsf{sig}}]\!]$ **then restart** ▷ *Too large*
  return $\mathsf{sig} := (\mathsf{salt}, \{u_j\}_j)$
function $\mathsf{FalconRS.Verify}(\mathsf{RL} := \{\mathsf{rvk}_j\}_j, \mathsf{M}, \mathsf{sig})$
  $(\mathsf{salt}, \{u_i\}_i) := \mathsf{sig}$; $\{h_i\}_i := \{\mathsf{rvk}_i\}_i$
  $c := H(\mathsf{salt}, \mathsf{RL}, \mathsf{M}) \in \mathcal{R}_q$
  $v := c - \sum_i h_i \cdot u_i$ ▷ *Compute v such that $c = v + \sum_i h_i \cdot u_i$*
  return $[\![\|((u_i)_i, v)\| \leq \beta_{\mathsf{sig}}]\!]$ ▷ *Verify pre-image shortness*

6.1.1 Ring Signature Construction

We build a ring signature $\mathsf{FalconRS}$ from Falcon and overcome limitations of previous works [24, 35] 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 [7], 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 [24].

Concretely, when party i signs, it first samples the contribution for other parties as discrete Gaussians of parameter σ, i.e., $u_j \xleftarrow{\$} \chi_u$ for $j \neq i$ (χ_u is tailcut to $[-\eta', \eta']$ for implementation purposes). It then samples $(u_i, v) \xleftarrow{\$} \mathsf{PreSmp}(\mathbf{B}_i, \sigma, -c')$, where $c' = H(\mathsf{M}, \mathsf{salt}) - \sum_{j \neq i} u_j \cdot h_j$, to complete the signature $\mathsf{sig} = (u_i)_{i \in [N]}$. 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. 11.

6.1.2 Security Analysis and Parameters

$\mathsf{FalconRS}$ 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. $\mathsf{FalconRS}$ can be proven unforgeable in an analogous manner to Falcon [23], 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.

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 $\mathsf{FalconRS}$ is deniable. We defer the formal statement to App. J.1.2 in the full version, 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 App. J.1.3 in the full version, obtaining that $\mathsf{FalconRS}$ is (μ, δ)-deniable for $\mu = 1 + 2^{-27}$ and $\delta = 2^{-57}$. We note that our analysis readily applies to Gandalf [24], 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 $\mathsf{FalconRS}$.

6.2 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 [1] 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 $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ with trapdoor tp, as public and secret keys respectively, from which is derived a larger map $\mathcal{P}^* : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$. To sign message M, one computes a target $\mathbf{t} = H(\mathsf{M}, \mathsf{salt})$ (where salt is a random value), and samples a preimage u such that $\mathcal{P}^*(\mathbf{u}) = \mathbf{t}$, leveraging the trapdoor tp. The final signature is $\mathsf{sig} \coloneqq (\mathsf{salt}, \mathbf{u})$.

The transform by Abe et al. [1] allows us to generically turn MAYO into an RS. We defer the full description to App. J.2 in the full version. 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.

6.2.1 Security Analysis and Parameters

We here overview the unforgeability and deniability of MayoRS, deferring formal statements and proofs to App. J.2.1 in the full version.

Unforgeability. Using the generic transform of [1], 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 MAYO_1 and MAYO_2 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 App. J.2.2 in the full version.

We compare the sizes and performance of FalconRS and MayoRS with the original schemes in Tabs. 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[11] and compare our RSs to prior works in App. J.3 in the full version. 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.

[11] Available as an artifact through DOI 10.5281/zenodo.15571694

7 Efficiency Comparison

Though RingXKEM has better deniability than SignXKEM, we need to consider the cost in terms of bandwidth and runtime. In Sec. 6, 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; though Hashimoto et al. [29, Sec. 6] also discussed bandwidth, they did not consider SignXKEM and only instantiated RingXKEM with Gandalf [24].

Table 4 compares instantiations of RingXKEM and SignXKEM, 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 and SignXKEM additionally use 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 and SignXKEM, the prekey bundle is a KEM public key, a Merkle tree authentication path, a signature on the root of this tree, and, for RingXKEM, an RS verification key. The X3DH handshake message is an ECDH public key and an authentication tag; PQXDH adds a KEM ciphertext. For RingXKEM and SignXKEM, this message is an encrypted (ring) signature and two ciphertexts.

We can clearly see in the table that the reduction in deniability by switching from RingXKEM to SignXKEM only leads to a modest decrease in transmission sizes, typically less than 1 kB. Due to the Merkle tree optimization proposed in [29], 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 [47] is also close in bandwidth requirements to the instantiation based on custom scheme Gandalf, for similar levels of deniability.

Acknowledgments

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

Ethic Considerations

The security and privacy properties of Signal's handshake protocol is relied on by many. This makes it relevant and important to understand and make comparisons between the deniability properties of (proposals for) Signal handshake protocols.

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

Risks and risk mitigation. As prior work has thoroughly investigated the security of X3DH and PQXDH, we deemed any risk of finding previously unknown vulnerabilities exceedingly unlikely. If any significant issues had been found, we would have coordinated with Signal developers on how to best protect Signal's users, both of the Signal app itself, and other users of Signal including Facebook Messenger, WhatsApp, and others, with responsible disclosure practices.

Benefits. Signal is transitioning towards full post-quantum security. We aim to contribute to this by providing new results and a model for comparing relevant deniability properties.

Open Science

The deniability model for Bundled AKE protocols is documented in this paper. For the proposed Falcon- and MAYO-based Ring signature schemes, we have experimental implementations made available in our artifact. It also includes prior ring signature implementations [4, 35] adapted with our benchmark code. There are no other datasets or implementations relevant to this paper.

References

[1] 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.

[2] Michael Backes, Aniket Kate, Sebastian Meiser, and Tim Ruffing. "Secrecy Without Perfect Randomness: Cryptography with (Bounded) Weak Sources." In: *ACNS 15International Conference on Applied Cryptography and Network Security*. 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.

[3] Ward Beullens, Fabio Campos, Sofía 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.

[4] Ward Beullens, Shuichi Katsumata, and Federico Pintore. "Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices." In: *ASI-*

Table 4: 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 [44]	32	128	3296	64	Derived from X25519
PQXDH	✗	DH+Kyber-1024	XEd25519 [44]	32	1696	166 496	1632	Derived from X25519
RingXKEM	✓	Kyber-512	Gandalf [24]	1696	2582	81 526	2804	Custom scheme
RingXKEM	✓	Kyber-1024	Gandalf [24]	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
SignXKEM	✓	Kyber-512	Falcon [47]	1697	1722	80 666	2202	NIST standard
SignXKEM	✓	Kyber-1024	Falcon [47]	2465	2490	157 466	3802	NIST standard
SignXKEM	✓	Kyber-512	MAYO_1 [3]	1968	1377	80 321	1857	NIST on-ramp R2
SignXKEM	✓	Kyber-1024	MAYO_1 [3]	2736	2145	157 121	3457	NIST on-ramp R2
SignXKEM	✓	Kyber-512	MAYO_2 [3]	6288	1236	80 180	1716	NIST on-ramp R2
SignXKEM	✓	Kyber-1024	MAYO_2 [3]	7056	2004	156 980	3316	NIST on-ramp R2

ACRYPT 2020, Part II. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer, Cham, Dec. 2020, pp. 464–492. DOI: 10.1007/978-3-030-64834-3_16.

[5] 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.

[6] Nikita Borisov, Ian Goldberg, and Eric Brewer. "Off-the-record communication, or, why not to use PGP." In: *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. CCS04: 11th ACM Conference on Computer and Communications Security 2004 (Washington DC USA). New York, NY, USA: ACM, Oct. 28, 2004, pp. 77–84. ISBN: 9781581139686. DOI: 10.1145/1029179.1029200. URL: https://otr.cypherpunks.ca/otr-wpes.pdf.

[7] 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.

[8] 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.

[9] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. "Towards Post-Quantum Security for Signal's X3DH Handshake." In: *SAC 2020*. Ed. by Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn. Vol. 12804. LNCS. Springer, Cham, Oct. 2020, pp. 404–430. DOI: 10.1007/978-3-030-81652-0_16.

[10] 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.

[11] Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. *Real World Deniability in Messaging*. Cryptology ePrint Archive, Report 2023/403. 2023. URL: https://eprint.iacr.org/2023/403.

[12] 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.

[13] 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.

[14] 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.

[15] Mario Di Raimondo and Rosario Gennaro. "New Approaches for Deniable Authentication." In: *Journal of Cryptology* 22.4 (Oct. 2009), pp. 572–615. DOI: 10.1007/s00145-009-9044-3.

[16] 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.

[17] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. "Tor: The Second-Generation Onion Router." In: *USENIX Security 2004*. Ed. by Matt Blaze. USENIX Association, Aug. 2004, pp. 303–320. DOI: 10.21236/ada465464.

[18] Yevgeniy Dodis, Jonathan Katz, Adam Smith, and Shabsi Walfish. "Composability and On-Line Deniability of Authentication." In: *TCC 2009*. Ed. by Omer Reingold. Vol. 5444. LNCS. Springer, Berlin, Heidelberg, Mar. 2009, pp. 146–162. DOI: 10.1007/978-3-642-00457-5_10.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Google. *Messages End-to-End Encryption Overview*. Technical paper. Feb. 2022. URL: https://www.gstatic.com/messages/papers/messages_e2ee.pdf.

[26] 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.

[27] 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.

[28] Keitaro Hashimoto, Shuichi Katsumata, and Thomas Prest. "How to Hide MetaData in MLS-Like Secure Group Messaging: Simple, Modular, and Post-Quantum." In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 1399–1412. DOI: 10.1145/3548606.3560679.

[29] 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, Jan. 13, 2025. URL: https://eprint.iacr.org/2025/040.

[30] Kee Jefferys, Maxim Shishmarev, and Simon Harman. *Session: End-To-End Encrypted Conversations With Minimal Metadata Leakage*. Tech. rep. Session, July 2024.

[31] Shaoquan Jiang. *Timed Encryption and Its Application*. Cryptology ePrint Archive, Report 2010/546. 2010. URL: https://eprint.iacr.org/2010/546.

[32] 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. URL: https://doi.org/10.1016/j.ic.2022.104866.

[33] Ehren Kret and Rolfe Schmidt. *The PQXDH Key Agreement Protocol*. Protocol documentation. Oct. 18, 2023. URL: https://signal.org/docs/specifications/pqxdh/.

[34] 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.

[35] Xingye Lu, Man Ho Au, and Zhenfei Zhang. "Raptor: A Practical Lattice-Based (Linkable) Ring Signature." In: *ACNS 19International Conference on Applied Cryptography and Network Security*. 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.

[36] Joshua Lund. *Technology preview: Sealed sender for Signal*. Oct. 2018. URL: https://signal.org/blog/sealed-sender/.

[37] Varun Madathil, Alessandra Scafuro, István András Seres, Omer Shlomovits, and Denis Varlakov. "Private Signaling." In: *USENIX Security 2022*. Ed. by Kevin R. B. Butler and Kurt Thomas. USENIX Association, Aug. 2022, pp. 3309–3326.

[38] Moxie Marlinspike and Trevor Perrin. *The X3DH Key Agreement Protocol*. Protocol documentation. Nov. 4, 2016. URL: https://signal.org/docs/specifications/x3dh/.

[39] Meta, Inc. *Messenger End-to-End Encryption Overview*. Technical white paper. Dec. 6, 2023. URL: https://engineering.fb.com/wp-content/uploads/2023/12/MessengerEnd-to-EndEncryptionOverview_12-6-2023.pdf.

[40] 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.

[41] Kirill Nikitin, Ludovic Barman, Wouter Lueks, Matthew Underwood, Jean-Pierre Hubaux, and Bryan Ford. "Reducing Metadata Leakage from Encrypted Files and Communication with PURBs." In: *PoPETs* 2019.4 (Oct. 2019), pp. 6–33. DOI: 10.2478/popets-2019-0056.

[42] Guilhem Niot. *Practical Deniable Post-Quantum X3DH: A Lightweight Split-KEM for K-Waay*. Cryptology ePrint Archive, Paper 2025/853. 2025. URL: https://eprint.iacr.org/2025/853.

[43] 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.

[44] Trevor Perrin. *The XEdDSA and VXEdDSA Signature Schemes*. documentation. Oct. 20, 2016. URL: https://signal.org/docs/specifications/xeddsa/.

[45] Trevor Perrin and Moxie Marlinspike. *The Double Ratchet Algorithm*. Protocol documentation. Nov. 20, 2016. URL: https://signal.org/docs/specifications/doubleratchet/.

[46] A Pfitzmann and Marit Hansen. "A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management." In: 34 (Jan. 2010). URL: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.

[47] 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.

[48] 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.

[49] 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.

[50] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. "SoK: Secure Messaging." In: *2015 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2015, pp. 232–249. DOI: 10.1109/SP.2015.22.

[51] 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.

[52] 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.

[53] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. "On the Cryptographic Deniability of the Signal Protocol." In: *ACNS 20International Conference on Applied Cryptography and Network Security, Part II*. Ed. by Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi. Vol. 12147. LNCS. Springer, Cham, Oct. 2020, pp. 188–209. DOI: 10.1007/978-3-030-57878-7_10.

[54] WhatsApp. *WhatsApp Encryption Overview*. Technical white paper. Sept. 27, 2023. URL: https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf.

[55] Andrew Chi-Chih Yao and Yunlei Zhao. "Deniable Internet Key Exchange." In: *ACNS 10International Conference on Applied Cryptography and Network Security*. Ed. by Jianying Zhou and Moti Yung. Vol. 6123. LNCS. Springer, Berlin, Heidelberg, June 2010, pp. 329–348. DOI: 10.1007/978-3-642-13708-2_20.

A Other Related Works

Deniable authentication, introduced in [20], provides both authentication and deniability: a sender can prove authorship of a message to its recipient, but the recipient (or an eavesdropping third party) cannot convince a third party of the sender's participation. This notion was formalized through a simulation paradigm, requiring transcripts to be efficiently simulatable. In particular, Pass [43] observes that a simulator for deniability needs to be implementable: the simulator can not rewind the adversary, reprogram a random oracle, or use similar techniques for simulation in thought experiments. Building upon [20], Di Raimondo et al. [16] extended deniability to authenticated key exchange (AKE) protocols. They introduced concurrent deniability, ensuring indistinguishability between real and simulated protocol executions even when the adversary may freely interact with honest parties as either initiator or as responder, interleaving between executions at will. They also introduced partial deniability, where parties

can deny having communicated with specific peers but not having participated in some protocol execution. Their models are defined with respect to a distinguisher that is presented with evidence after the fact, i.e. an offline distinguisher, and require the simulatability of both transcripts and session keys. A property related to partial deniability is the so-called peer-deniability [13]. Other works adapted this model to their setting [31, 55], or proposed alternative models, such as outsider deniability [14]. Stronger notions of deniability have been defined, too. Forward deniability [15] requires the protocol to be statistical zero-knowledge. Another strong (and somewhat less intuitive) flavor of deniability, involving an on-line distinguisher which can interact with protocol participants during the execution, was introduced in [18]. This notion is difficult to obtain [51] and arguably less important than offline-deniability, since a protocol that is not offline-deniable leaves proofs of communication that anyone can verify later. In this work we focus on offline deniability.

B Strong Implies Standard Deniability

In this section, we demonstrate that, as one would expect, deniability against malicious adversaries implies deniability against honest but curious adversaries.

Lemma 1 (Strong Local Deniability $\Rightarrow$ (Standard) Local Deniability)**.** *If a two-round bundled authenticated key exchange protocol* BAKE *is strongly local deniable for accused users* $\mathcal{H} \subseteq \mathcal{N}$ *against* malicious *accusers* $C \coloneqq \mathcal{N} \setminus \mathcal{H}$ *with respect to a leakage function* $\mathcal{L}_{\mathsf{leak}}$ *with* $\mathsf{leak} \in \{\mathsf{low}, \mathsf{med}, \mathsf{high}\}$*, then it is locally deniable for accused users* $\mathcal{H}$ *against* honest-but-curious *accusers* C *for the same leakage function* $\mathcal{L}_{\mathsf{leak}}$*, and for any disclosure function* $\mathcal{D}_{\mathsf{disc}}$*.*

Proof. We will prove this by contradiction. The intuition of the proof is as follows: we show that we can construct an accuser for the malicious security game that is equivalent to the local honest security game. If this accuser has negligible advantage, no accuser can exist that has non-negligible advantage in the local honest security game.

We assume BAKE is *not* (standard) local deniable for accused users $\mathcal{H}$ for some leakage function $\mathcal{L}_{\mathsf{leak}}$ with $\mathsf{leak} \in \{\mathsf{low}, \mathsf{med}, \mathsf{high}\}$ and for any disclosure function $\mathcal{D}_{\mathsf{disc}}$. That is, for any simulator $\mathsf{Sim} = (\mathsf{SimPreK}, \mathsf{SimTrans})$, there exists a distinguisher D for which $\mathsf{Adv}^{\mathsf{local}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}},\mathcal{D}_{\mathsf{disc}}}(1^\lambda)$ is non-negligible.

Let us now describe an accuser $\mathcal{A}$ for the strong deniability game. If $\mathtt{mode} = \mathtt{real}$, then, on input $((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}, \mathsf{st}_{\mathcal{A}})$, $\mathcal{A}$ runs the honest prekey generation algorithm $(\vec{\mathsf{prek}}_u, \mathsf{st}_u) \xleftarrow{\$} \mathsf{BAKE.PreKeyBundleGen}(\mathsf{isk}_u)$ for all $u \in C$ and outputs $((\vec{\mathsf{prek}}_u)_{u\in C}, \mathsf{st}_{\mathcal{A}} \coloneqq (\mathsf{isk}_u, \mathsf{st}^{\mathsf{init}}_u \coloneqq \mathsf{st}_u)_{u\in C})$. When $\mathcal{A}$ is invoked within oracle $O_{\mathsf{strong\text{-}local}}(s, r)$ for $(s, r) \in \mathcal{N} \times \mathcal{H}$, it executes $(K, \rho) \xleftarrow{\$} \mathsf{BAKE.Send}(\mathsf{isk}_s, \mathsf{ik}_r, \mathsf{prek}_{r,t})$, where note that this is well-defined as we have $\mathsf{isk}_s \in \mathsf{st}_{\mathcal{A}}$. The accuser then outputs $(\rho, \mathsf{st}_{\mathcal{A}})$.

For the sake of contradiction, let us assume an arbitrary simulator $\mathsf{Sim}_{\mathcal{A}} = (\mathsf{SimPreK}_{\mathcal{A}}, \mathsf{SimTrans}_{\mathcal{A}})$ for the malicious deniability game, and that, for any distinguisher D', $\mathsf{Adv}^{\mathsf{strong\text{-}local}}_{\mathsf{D}',\mathcal{L}_{\mathsf{leak}}}(1^\lambda)$ is negligible. Due to the way we defined the accuser $\mathcal{A}$ (namely to honestly execute the protocol), $\mathsf{Sim}_{\mathcal{A}}$ can be viewed as a simulator Sim against the local honest deniability game. Moreover, the information leaked from accused users by $\mathcal{L}_{\mathsf{leak}}$ is the same in both games, while the information disclosed by $\mathcal{A}$ to D' in $\mathsf{st}_{\mathcal{A}}$ contains (at least) the information output by $\mathcal{D}_{\mathsf{disc}}$ for the corresponding $\mathsf{disc} \in \{\mathsf{low}, \mathsf{med}, \mathsf{high}\}$.

Even in the most restricted leakage setting, the distinguisher D' in the local malicious deniability game obtains $\mathsf{st}_{\mathcal{A}} = (\mathsf{isk}_u, \mathsf{st}^{\mathsf{init}}_u)_{u\in C}$. Therefore, if any D' has negligible advantage given the output of $\mathcal{L}_{\mathsf{leak}}$, so does any D against the local honest deniability game given the output of $\mathcal{L}_{\mathsf{leak}}$. However, this contradicts our assumption made at start of the proof, completing the proof. □