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

August 27, 2025

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, Katsumata, and Wiggers'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 the settings for which PQXDH is (un)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.

This is the full version of a paper appearing in the proceedings of the 34th USENIX Security Symposium (USENIX Security '25). This version additionally includes formal statements and proofs for the Deniable BAKE protocols considered in this work, as well as additional details and proofs for our ring signature constructions.

Contents

1. Introduction

The Signal protocol [MP16; PM16] does not just power the Signal app, it also underpins messaging apps such as WhatsApp [Wha23], Google RCS [Goo22], and Facebook Messenger [Met23], collectively serving billions of users. To initiate a conversation, Signal users perform a handshake protocol to establish a shared key, which is then used for encrypted communication via the Double Ratchet protocol [PM16]. This handshake was originally implemented as X3DH [MP16], based on Triple Diffie-Hellman [KP05]. In late 2023, as a step towards fully post-quantum (PQ) security, X3DH was replaced with PQXDH [KS23], offering protection against "harvest-now-decrypt-later" (HNDL) attacks. There also exist several proposals for fully PQ Signal handshake protocols [Bre+22; Col+24; Has+22; HKW25].

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, Katsumata, and Wiggers [HKW25] 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 [HKW25] 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 [KS23; MP16] — 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 [KS23; MP16]. 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 [Bre+22; Col+24; FJ24; Has+22; Vat+20] — 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 [HKW25]. 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 [CF11; Dag+13; DGK06; UG15; UG18], and adaptations made for specific handshake protocols [Bre+22; Col+24; FJ24; Has+22; Vat+20]. 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 [Has+22]. 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 [HKW25], we can capture the lifecycle of prekey bundles. Each batch of prekey bundles contains a number of *one-time* prekey bundles, and a single *last-resort* prekey bundle. One-time prekey bundles are deleted after use. The last-resort prekey bundle ensures recipient availability even if they are offline for extended amounts of time; it is only used if all one-time prekey bundles are used up, and is only deleted when a new batch of prekeys is uploaded [KS23; MP16]. Our model distinguishes 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 [HKW25].

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

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

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 in a setting where the accusers are limited to be honest-but-curious. More interestingly though, against malicious receiver accusers, we show PQXDH to be *un*deniable, showing that protocols may become undeniable when the distinguisher is more powerful than the accuser. Intuitively, using the powerful distinguisher, a weak accuser can "prove" to the distinguisher that it could not have computed some secret known to the sender. This is akin to a recent result by Fiedler and Langrehr [FL25], where they show X3DH and PQXDH to be undeniable when the running time of the (classical, polynomial time) accuser is shorter compared to the (classical, polynomial time) distinguisher, and when all parties have access to some extra auxiliary input.

A pragmatic metric for deniability. We introduce a novel measure for deniability inspired by concepts in differential privacy and differential indistinguishability [Bac+15; Dwo+06; Mir+09]. Prior works required the real evidence π_{real} and simulated evidence π_{sim} to be indistinguishable by 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* [SV16]. 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 [HKW25], accurately modeling prekey bundles.

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. Finally, we show that, in the classical ROM[3], PQXDH is strong HNJL deniable in case the accuser is limited to a malicious sender. However, as stated above, PQXDH is *not* strong HNJL deniable when the accuser is a malicious receiver.

Analysis of fully PQ protocols. We extend our study to Hashimoto et al.'s fully PQ RingXKEM [HKW25], 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 [Has+21], which replaces RSs by (plain) signatures, and show that it offers some level of deniability under limited leakage and disclosure, highlighting the precise nature of our model in capturing weaker notions of deniability. 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 [GJK24b; LAZ19b] 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 [Pre+22], and the additional signature candidate MAYO [Beu+24]. Our RS using Falcon is based on the generic lattice-based RS design by [GJK24b]. Our 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.

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

[3] We leave it as future work to extend this result to the QROM.

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	●† ●	●† ●	$●^{SO}$ ●	?	$●^{SO}$ ●	?	$◕^{SO}$ ●	?
strong-	global	◕† ●	◕† ●	$◕^{SO}$ ●	?	$◓^{SO}$ ●	?	$◕^{SO}$ ●	?
Security [HKW25]		Classical	Harvest-Now Decrypt-Later			Fully post-quantum			

Legend

Icon	**Last-resort prekey: No** leak/disc	**Yes** leak/disc
●	high	high
◕	high	med
◓	med	med
◴	med	low
○	low	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 using a last resort prekey bundle.

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

1.2. Related Work

Deniability of X3DH. The introduction of Off-the-Record Messaging established deniability as a key feature for secure messaging [BGB04]. The Signal protocol adopted this goal in its X3DH handshake [MP16], analyzed by Vatandas et al. [Vat+20] under the simulation-based model of [DGK06] 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 PQXDH protocol [KS23], deployed in 2023, 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 [Bha+24; FG25; HKW25]. The deniability guarantees of PQXDH were recently analyzed in [FJ24], viewing it as a specific handshake protocol proposed by [FG25]. 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. In a very recent work, Fiedler and Langrehr [FL25] showed the *un*deniability of X3DH and PQXDH when receivers are malicious, are provided as auxiliary input some hint on a Diffie-Hellman instance, and can maliciously generate their identity keys. More concretely, their result relies on the non-standard *hinted* computational Diffie-Hellman (CDH) problem which is difficult for a (classical) accuser running in time T but easy for a (classical) distinguisher running in time $T' \gg T$. When the distinguisher is quantum, as in the case of HNJL deniability, we show that we can simply rely on the standard CDH problem.

Deniability of fully PQ protocols. Hashimoto et al. [Has+22] 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 [DGK06]. To prove deniability against malicious (classical) accusers, they require knowledge assumptions. In concurrent work, Brendel et al. [Bre+22] 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. [Col+24] introduced K-Waay, a protocol based on split-KEMs [Bre+20; Nio25]. 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 [Bre+22; Col+24] 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 [Has+22; Vat+20], 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 [Bre+22; FJ24] allow judges to compromise secrets associated to identity keys and last resort prekeys of *all* users, and [Col+24] 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, [Bre+22; Col+24] 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 [Bre+22; Col+24; Has+22; Vat+20]. Fiedler and Janson [FJ24] require deniability of prekey bundle uploads (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., [Bre+22; Col+24]), we do not consider the deniability of registering or uploading prekey bundles. Our focus is the deniability of the handshake message and the computation of the resulting 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 [Lun18; Mad+22], metadata protection [HKP22; TWG24], anonymity and unlinkability [PH10; Ung+15]. 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 [Nik+19] or anonymous communication [DMS04; JSH24] 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 [Lun18; Mad+22], metadata protection [HKP22; TWG24], anonymity and unlinkability [PH10; Ung+15].

2. Modeling Signal Handshake Protocols and Problem Setting

We model Signal handshake protocols as *bundled authenticated key exchange* (BAKE) protocols, introduced in [HKW25]. 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 [KS23; MP16], among other constructions such as RingXKEM [HKW25] and variants [Bre+22; Has+21; Has+22]. In [HKW25], 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.

2.1. Modeling Signal Handshake Protocols with Bundled AKEs

We recall the definition of a BAKE protocol [HKW25].

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.

$\mathsf{BAKE.Receive}(\mathsf{isk}_r, \mathsf{st}_r, \mathsf{ik}_s, t, \rho) \to (K', \mathsf{st}_r)$: 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 [KS23; MP16]. 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 [MP16].

2.2. Deniability: Entities and Roles

The entities involved in deniability are categorized as follows, following established terminologies, e.g., [CCH23; DGK06; Ung+15].

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 [DGK06; Vat+20]. 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:

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

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:

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

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

[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.

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., [Bre+22; Col+24].

Honest registration of identity keys. Similarly to other works on the deniability of Signal handshake protocols [Bre+22; Jia+22; Vat+20], we assume that users *honestly* generate their identity keys. This simplifies the definition and proofs of deniability while maintaining practical relevance. Works allowing maliciously generated identity keys [FJ24; Has+22] 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.

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 [Bac+15; Dwo+06; Mir+09], 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 [DNS98]. This has been used to define deniable AKEs, e.g., [DGK06]. We highlight that this simulator must not only exist but also be constructible in the real world [Pas03]. 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 [Bac+15; Dwo+06; Mir+09]. Prior works required the real evidence π_{real} and simulated evidence π_{sim} to be *indistinguishable* by a distinguisher D. That is, they (informally) required the statistical distance to be close: $|\Pr[\mathsf{D}(\pi_{\mathsf{real}}) = 0] - \Pr[\mathsf{D}(\pi_{\mathsf{sim}}) = 0]| = \delta(\lambda)$ 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* [SV16]. 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 [Pre+22] is insufficient for the standard notion of deniability, it is sufficient for the relaxed definition with a multiplicative slack $\mu(\lambda) = 1 + 2^{-27}$. See 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. 29).

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, 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.

Next, 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. 34 to 38, respectively. Outsider accusers, used for global deniability, are captured by Lns. 39 to 41, 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 BAKE.Receive algorithm, $\mathsf{SimSt}_{\mathcal{H}}$ is run when simulating an honest receiver r's updated state (cf. Ln. 42). 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 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.

[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).

[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 $\mathsf{Game}^{\mathsf{ATK}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}},\mathcal{D}_{\mathsf{disc}}}(1^\lambda, \mathtt{mode})$
  $C \coloneqq \mathcal{N} \setminus \mathcal{H}$
  for user $u \in \mathcal{N}$ **do**
    $(\mathsf{ik}_u, \mathsf{isk}_u) \xleftarrow{\$} \mathsf{BAKE.IdKeyGen}(1^\lambda)$
    $\mathsf{counter}_u \coloneqq 0$ ▷ *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 $u \in C$ **do**
      $(\vec{\mathsf{prek}}_u, \mathsf{st}_u) \xleftarrow{\$} \mathsf{BAKE.PreKeyBundleGen}(\mathsf{isk}_u)$
      $\mathsf{st}^{\mathsf{init}}_u \coloneqq \mathsf{st}_u$
  else ▷ $\mathtt{mode} = \mathtt{sim}$
    ▷ *Simulate prekey bundles of accused users* $\mathcal{H}$ ◁
    $((\vec{\mathsf{prek}}^*_u)_{u\in\mathcal{H}}, (\vec{\mathsf{prek}}_u)_{u\in C}, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{SimPreK}\left((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}\right)$
    $(\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}} \leftarrow (\vec{\mathsf{prek}}^*_u)_{u\in\mathcal{H}}$
  $\mathsf{st}_{\mathsf{D}} \xleftarrow{\$} \mathsf{D}^{O_{\mathsf{ATK}}}((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in\mathcal{N}})$ ▷ D *obtains transcripts*
  if $[\![\mathtt{mode} = \mathtt{sim}]\!]$ **then**
    $(\mathsf{st}_u, \mathsf{st}^{\mathsf{init}}_u)_{u\in C} \xleftarrow{\$} \mathsf{SimSt}_C(\mathsf{st}_{\mathsf{Sim}})$ ▷ *Corrupted states*
  $\mathsf{leak}_{\mathsf{D}} \coloneqq \mathcal{L}_{\mathsf{leak}}((\mathsf{isk}_u, \mathsf{st}_u)_{u\in\mathcal{H}})$
  $\mathsf{disc}_{\mathsf{D}} \coloneqq \mathcal{D}_{\mathsf{disc}}((\mathsf{isk}_u, \mathsf{st}_u, \mathsf{st}^{\mathsf{init}}_u)_{u\in C})$
  $\mathsf{aux}_{\mathsf{D}} \coloneqq (\mathsf{leak}_{\mathsf{D}}, \mathsf{disc}_{\mathsf{D}})$
  $b \xleftarrow{\$} \mathsf{D}(\mathsf{st}_{\mathsf{D}}, \mathsf{aux}_{\mathsf{D}})$ ▷ D *outputs a bit* $b \in \{0,1\}$
  return b
function $O_{\mathsf{ATK}}(s, r)$
  require $[\![(s,r) \in \mathcal{N}\times\mathcal{N}]\!] \wedge [\![s \neq r]\!] \wedge [\![(s,r) \notin C \times C]\!]$
  if $[\![\mathsf{ATK} = \mathsf{Local}]\!]$ **then require** $[\![(s,r) \notin \mathcal{H}\times\mathcal{H}]\!]$
  $\mathsf{counter}_r \leftarrow \mathsf{counter}_r + 1$
  $t \coloneqq \mathsf{counter}_r$
  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 $[\![(s,r) \in \mathcal{H}\times C]\!]$ **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 $[\![(s,r) \in C\times\mathcal{H}]\!]$ **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 $[\![(s,r) \in \mathcal{H}\times C]\!]$ **then return** (K, ρ)
  else if $[\![(s,r) \in C\times\mathcal{H}]\!]$ **then return** (K', ρ)
  else return (K, K', ρ) ▷ $(s,r) \in \mathcal{H}\times\mathcal{H} \Rightarrow \mathsf{ATK} = \mathsf{Global}$

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

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 O_{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. Alg. 3, 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 rolls back the updated state after executing $\mathsf{BAKE.Receive}$ (cf. Alg. 3, 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

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

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

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** $\mathsf{Game}^{\text{strong-ATK}}_{\mathcal{A},\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}}}(1^\lambda, \texttt{mode})$
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** $[\![\texttt{mode} = \texttt{real}]\!]$ **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** $[\![\texttt{mode} = \texttt{sim}]\!]$ **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: $\mathsf{st}_{\mathsf{D}} \xleftarrow{\$} \mathsf{D}^{O_{\mathsf{ATK}}}((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in[N]})$ ▷ *cf. Alg. 3*
15: **if** $[\![\texttt{mode} = \texttt{sim}]\!]$ **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

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

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** $O_{\mathsf{ATK}}(s, r)$
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: $t := \mathsf{counter}_r$
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** ▷ $\mathtt{mode} = \mathtt{sim}$
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: $\mathsf{st}_{\mathsf{tmp}} := \mathsf{st}_r$ ▷ *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) ▷ $(s,r) \in \mathcal{H} \times \mathcal{H} \Rightarrow \mathsf{ATK} = \mathsf{Global}$

4. Deniability of X3DH and PQXDH

X3DH is the original handshake protocol used by Signal [MP16], based on Triple Diffie–Hellman [KP05]. In 2023, Signal rolled out a variant PQXDH [MP16], 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, Katsumata, and Wiggers [HKW25]. 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 14). PQXDH simply adds a KEM key exchange to X3DH, as shown in Ln. 15.

The description of the Send and Receive algorithms for X3DH and PQXDH are given in Algs. 5 and 6, respectively, where PQXDH-specific functionality is marked with a gray dotted box. Note that we follow the description given by [HKW25, 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

Algorithm 4 X3DH and PQXDH identity key and prekey bundle generation algorithms [HKW25]. 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** $(\vec{\mathsf{prek}}_u, \mathsf{st}_u := (D_{\mathsf{prek}}, D_{\rho_\perp}))$

PQXDH and Double Ratchet protocols. For a detailed comparison between Alg. 5, the Signal documentation, and Signal's implementation, refer to [HKW25, 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, Katsumata, and Wiggers [HKW25, 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 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. E.

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. 14). 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

Algorithm 5 The PQXDH and X3DH.Send algorithm [HKW25].

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

Algorithm 6 The PQXDH and X3DH.Receive algorithm [HKW25].

function PQX3DH.Receive($\mathsf{isk}_r, \mathsf{st}_r, \mathsf{ik}_s, t, \rho$)
  $(\overline{\mathsf{isk}}_r, \mathsf{sk}_r) \leftarrow \mathsf{isk}_r$; $(\overline{\mathsf{ik}}_s, \mathsf{vk}_s) \leftarrow \mathsf{ik}_s$
  $(D_{\mathsf{prek}}, D_{\rho_\bot}) \leftarrow \mathsf{st}_r$
  if $[\![t \neq \bot]\!]$ **then** ▷ *One-time prekey bundle*
    require $[\![D_{\mathsf{prek}}[t] \neq \bot]\!]$ ▷ *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., $t = \bot$)*
    require $[\![\rho \notin D_{\rho_\bot}]\!]$ ▷ *Check ρ is not replayed.*
    $D_{\rho_\bot} \leftarrow D_{\rho_\bot} \cup \{\rho\}$
    $(\mathsf{prek}_{r,t}, (\mathsf{spksec}_r, \bot, \mathsf{dk}_{r,t})) \leftarrow D_{\mathsf{prek}}[t]$
  $(\mathsf{epk}, \mathsf{ct}, \tau_{\mathsf{conf}}) \leftarrow \rho$
  $\mathsf{ss}_1 := [\mathsf{spksec}_r]\overline{\mathsf{ik}}_s$; $\mathsf{ss}_2 := [\overline{\mathsf{isk}}_r]\mathsf{epk}$
  $\mathsf{ss}_3 := [\mathsf{spksec}_r]\mathsf{epk}$; $\mathsf{ss} := \mathsf{ss}_1 \| \mathsf{ss}_2 \| \mathsf{ss}_3$
  if $[\![t \neq \bot]\!]$ **then** ▷ *One-time prekey bundle*
    $\mathsf{ss}_4 := [\mathsf{osk}_{r,t}]\mathsf{epk}$; $\mathsf{ss} := \mathsf{ss}_1 \| \mathsf{ss}_2 \| \mathsf{ss}_3 \| \mathsf{ss}_4$
  $\mathsf{ss}_{\mathsf{KEM}} \leftarrow \mathsf{KEM.Decaps}(\mathsf{dk}_{r,t}, \mathsf{ct})$
  $\mathsf{content} := \mathsf{ik}_s \| \mathsf{ik}_r \| \mathsf{prek}_{r,t} \| \mathsf{epk} \| \mathsf{ct}$
  $K \| \tau'_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss} \| \mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$
  require $[\![\tau_{\mathsf{conf}} = \tau'_{\mathsf{conf}}]\!]$
  ▷ *Delete prekey bundle if not last-resort* ◁
  if $[\![t \neq \bot]\!]$ **then** $D_{\mathsf{prek}}[t] \leftarrow \bot$
  $\mathsf{st}_r \leftarrow (D_{\mathsf{prek}}, D_{\rho_\bot})$
  return (K, st_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., leak = 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. 16). As long as the distinguisher cannot decrypt the ciphertext ct included in ρ, K remains indistinguishable.

Strong local and global classical deniability. Unlike previous works relying on knowledge type assumptions, we rely on the generic group model (GGM) [Sho97] to show strong (local and global) deniability. 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 [Pas03], requiring 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 [CPZ20], giving us more confidence in generic accusers.

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

Similarly, we show strong local and global deniability of PQXDH against a classical accuser and a *classical* distinguisher.

Strong HNJL local and global deniability for receivers. For strong (local and global) HNJL deniability of PQXDH, we are only able to prove deniability for the *receiver*; see the paragraph below for a discussion on why the deniability of the *sender* breaks. In fact, we can only prove strong deniability for the receiver in the *classical* ROM, while still enabling quantum capability to the 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 a key and a confirmation tag $K\|\tau_{\mathsf{conf}}$, since the receiver must recompute the same tag τ_{conf} as the sender to accept. If such a key K exists, 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.

Attack on strong HNJL deniability for senders. Interestingly, the situation changes when considering strong deniability for *senders*. We observe that there is an attack on strong sender deniability if the distinguisher is *quantum* while the accuser and simulator are classical, regardless of the classical or quantum ROM. Namely, a malicious receiver can craft a public key spk_r in its prekey bundle, such that it can later prove that it does not know the associated secret key spksec_r. For instance, it can hash to a random group element so that it does not know the associating secret exponent. This means that if a handshake message contains a key K and tag τ_{conf} derived from inputs $\mathsf{ss}_1, \mathsf{ss}_2, \mathsf{ss}_3$, and ss_4 (if the last resort prekey is not used), that are correctly computed, then these must have been computed with the sender's secrets. The key difference here compared to deniability in the classical setting is in the distinguisher's ability to tell that $\mathsf{ss}_1, \mathsf{ss}_2, \mathsf{ss}_3$, and ss_4 are correctly computed. Indeed, a distinguisher with quantum capabilities can compute the secrets associated to sender and receiver public keys, and simply recompute these values to check, and thereby decide whether the transcript was produced by the sender.
As explained above, a similar attack does not work if the malicious accuser is a sender, since without knowledge of the secret keys, the sender cannot compute a tag τ_{conf} that passes the receiver's check on Alg. 6 Ln. 19.

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. E.6.*

5. Deniability of RingXKEM

RingXKEM is a PQ Signal handshake protocol by Hashimoto, Katsumata, and Wiggers [HKW25], using ring signatures to simultaneously ensure authentication and deniability. This is an optimized protocol based on [Bre+22; Has+21; Has+22] where receiver bandwidth and storage requirements on the server are reduced by using Merkle trees for the authentication of prekey bundles.

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

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 $b \xleftarrow{\$} \mathcal{A}^{O_{\mathsf{RS.Sign}}(\cdot,\cdot,\cdot)}((\mathsf{rvk}_u, \mathsf{rsk}_u)_{u\in[N]})$
function $O_{\mathsf{RS.Sign}}(\mathsf{M}, u_0, u_1)$
  require $[\![(u_0, u_1) \in [N] \times [N]]\!] \wedge [\![q < Q]\!]$
  $q \leftarrow q + 1$
  return $\sigma \xleftarrow{\$} \mathsf{RS.Sign}(\mathsf{rsk}_{u_b}, \mathsf{M}, \{\mathsf{rvk}_{u_0}, \mathsf{rvk}_{u_1}\})$

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.

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.
$\mathsf{RS.Verify}(\mathsf{RL}, \mathsf{M}, \mathsf{sig}) \to 1/0$: 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 $\mu : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+$ 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 $\mathcal{A}$,

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

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 [Pre+22] 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, Katsumata, and Wiggers [HKW25] 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. The algorithms defining the RingXKEM Send and Receive algorithms are given in Algs. 9 and 10.

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

function RingSignXKEM.IdKeyGen(1^λ)
  $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{KEM.KeyGen}(1^\lambda)$
  $(\mathsf{rvk}, \mathsf{rsk}) \xleftarrow{\$} \mathsf{RS.Sign.KeyGen}(1^\lambda)$
  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* ◁
  $(\mathsf{root}_u, \mathsf{tree}_u) \leftarrow \mathsf{MerkleTree}((\widehat{\mathsf{ek}}_{u,t})_{t \in [L] \cup \{\perp\}})$
  $\sigma_{u,\mathsf{root}} \xleftarrow{\$} \mathsf{RS.Sign.Sign}(\mathsf{rsk}_u, \mathsf{root}_u, \{\mathsf{rvk}_u\})$
  $(\widehat{\mathsf{rvk}}_u, \_) \xleftarrow{\$} \mathsf{RS.KeyGen}(1^\lambda)$ ▷ *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 $\begin{pmatrix} \vec{\mathsf{prek}}_u := (\mathsf{prek}_{u,t})_{t \in [L] \cup \{\perp\}}, \\ \mathsf{st}_u := (D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_u, D_{\rho_\perp}) \end{pmatrix}$

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 $[\![\, \mathsf{RS.Sign.Verify}(\mathsf{vk}_r, \{\mathsf{rvk}_r\}, \widehat{\mathsf{ek}}_r, \sigma_{r,\mathsf{root}}) = 1 \,]\!]$
  $(\mathsf{ss}_r, \mathsf{ct}_r) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek}_r)$
  $(\widehat{\mathsf{ss}}_r, \widehat{\mathsf{ct}}_r) \xleftarrow{\$} \mathsf{KEM.Encaps}(\widehat{\mathsf{ek}}_{r,t})$
  $\mathsf{context} := \mathsf{ik}_s \| \mathsf{ik}_r \| \mathsf{prek}_{r,t} \| \mathsf{ct}_r \| \widehat{\mathsf{ct}}_r$
  $K \| K_{\mathsf{ske}} := \mathsf{KDF}(\mathsf{ss}_r \| \widehat{\mathsf{ss}}_r, \mathsf{context})$
  $\sigma_s \xleftarrow{\$} \mathsf{RS.Sign.Sign}(\mathsf{rsk}_s, \mathsf{context}, \{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\})$
  $\mathsf{ct}_{\mathsf{ske}} \xleftarrow{\$} \mathsf{SKE.Enc}(K_{\mathsf{ske}}, \sigma_s)$ ▷ *Mask signature*
  $\rho := (\mathsf{ct}_r, \widehat{\mathsf{ct}}_r, \mathsf{ct}_{\mathsf{ske}})$
  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*
    $D_{\rho_\perp} \leftarrow D_{\rho_\perp} \cup \{ (\mathsf{ct}_r, \widehat{\mathsf{ct}}_r) \}$
  $(\mathsf{prek}_{r,t}, \widehat{\mathsf{dk}}_{r,t}) \leftarrow D_{\mathsf{kem}}[t]$
  $\mathsf{ss}_r := \mathsf{KEM.Decaps}(\mathsf{dk}_r, \mathsf{ct}_r)$
  $\widehat{\mathsf{ss}}_r := \mathsf{KEM.Decaps}(\widehat{\mathsf{dk}}_{r,t}, \widehat{\mathsf{ct}}_r)$
  $\mathsf{content} := \mathsf{ik}_s || \mathsf{ik}_r || \mathsf{prek}_{r,t} || \mathsf{ct}_r || \widehat{\mathsf{ct}}_r$
  $K || K_{\mathsf{ske}} := \mathsf{KDF}(\mathsf{ss}_r || \widehat{\mathsf{ss}}_r, \mathsf{content})$
  $\sigma_s := \mathsf{SKE.Dec}(K_{\mathsf{ske}}, \mathsf{ct}_{\mathsf{ske}})$ ▷ *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)

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. G.

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

We further have global deniability, thanks to the RS key $\widehat{\mathsf{rvk}}$ in the prekey bundle not being authenticated. Specifically, the simulator, given prekey bundles of honest users, can substitute the verification key $\widehat{\mathsf{rvk}}_r$ with one for which it knows the associated secret key. It can then easily compute the required ring signature and simulate the communication of two honest users. However, note that the receiver's state $\mathsf{st}_r := (D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_r, D_{\rho_\perp})$ contains the (non-simulated) verification key $\widehat{\mathsf{rvk}}_r$, as such, if the this state leaks, the distinguisher can easily tell apart real and simulated executions. Hence, global deniability holds so long as the honest receiver's state does not leak, i.e., for $\mathsf{leak} = \mathsf{med}$.

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. G.3.1 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 [Has+21], 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. H. 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 introduce 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 [Pre+22], and the additional signature candidate MAYO [Beu+24]. Both of these schemes have short signatures, and thus great potential for compact ring signatures.

6.1. Falcon-Based Ring Signature

This section provides a high-level description for our Falcon-based ring signature, based on the generic lattice-based RS design by [GJK24b] instantiated with Falcon-specific components. Formal definitions are given in App. I.1.

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 $\|(u, v)\| \leq \beta$.

6.1.1. Ring Signature Construction

We propose a ring signature FalconRS built from Falcon, and overcome theoretical limitations of previous works [GJK24b; LAZ19b] as it seems impossible to prove standard anonymity when using Falcon components. We instead aim for our relaxed deniability notion.

Our RS scheme samples a public polynomial $h_i \in \mathcal{R}_q$ and trapdoor $\mathbf{B}_i$ for each ring user $i \in [N]$. Signing is modified to find a preimage of c for an aggregation of the public keys: we sample $v, (u_i)_{i\in[N]}$ such that $c = v + \sum_i h_i \cdot u_i$ knowing only the trapdoor for a single h_i. This aggregation is inspired by ring trapdoor functions [BK10], although lattices allow further optimization: we can reuse coordinate v for all signers and omit it in the final signature since it can be recovered from c and the u_i. This optimization was recently analyzed in [GJK24b], proposing a generic lattice-based RS construction called Gandalf.[10]

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

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

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

Deniability. Proving anonymity appears unfeasible for an RS based on the Falcon parameters, as the distribution of signatures is at a non-negligible distance from the ideal one. Instead, we show that FalconRS is deniable. We defer the formal statement to App. I.1.2, 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

[10] Note that [GJK24b] instantiates their generic construction Gandalf using a trapdoor sampler from Antrag [Esp+23] and a preimage sampler from Mitaka [Esp+22]. Their concrete RS instantiation is also called Gandalf. Below, to avoid confusion, we always mean the concrete instantiation when referring to Gandalf.

Algorithm 11 Falcon-based ring signature scheme

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

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. $\eta' \coloneqq \lceil \sqrt{70 \cdot \log(2)} \cdot \sqrt{2} \cdot \sigma \rceil = 1633$). 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. I.1.3, obtaining that FalconRS is (μ, δ)-deniable for $\mu = 1 + 2^{-27}$ and $\delta = 2^{-57}$. We note that our analysis readily applies to Gandalf [GJK24b], which chooses a different trapdoor generator and preimage sampler to instantiate the generic construction of [GJK24b]. While Gandalf initially claimed an anonymity of 2^{-70}, an updated version acknowledged a proof flaw and an anonymity of only 2^{-30}. Alternatively, Gandalf can be proven *deniable*, with roughly the same μ, δ as FalconRS.

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 [AOS02] to obtain an efficient RS. We analyze parameter sets proposed for standardization and provide alternative ones achieving higher deniability.

MAYO is designed over quadratic maps. At a high level, it chooses a map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ 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, Ohkubo, and Suzuki [AOS02] allows us to generically turn MAYO into an RS. We defer the full description to App. I.2. 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. I.2.1.

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

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

One could directly use the parameters from the MAYO specification, but 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. I.2.2.

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 Sec. 6.3. Our constructions are as compact as the state-of-the-art, and built on more scrutinized schemes. Additionally, our implementations largely outperform the state-of-the-art, by factors 32–66× for signing, 146–1025× for verification.

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

Base Scheme	(μ, δ)	**PK**	**Sig**	**2-RSig**
Falcon-512	$1 + 2^{-27}, 2^{-57}$	897 B	666 B	1288 B
MAYO_1	$1, 2N \cdot 2^{-36}$	1168 B	321 B	650 B
MAYO_2	$1, 2N \cdot 2^{-36}$	5488 B	180 B	368 B
MAYO^*	$1, 2N \cdot 2^{-52}$	1569 B	335 B	677 B
MAYO^{**}	$1, 2N \cdot 2^{-83}$	1591 B	374 B	756 B
MAYO^{***}	$1, 2N \cdot 2^{-124}$	1771 B	492 B	992 B

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

Scheme	**Keygen**	**Sign**		**Verify**	
		normal	2-ring	normal	2-ring
Falcon-512	6.2 Mc	0.26 Mc	0.74 Mc	0.02 Mc	0.04 Mc
MAYO_1	0.24 Mc	0.88 Mc	1.1 Mc	0.17 Mc	0.28 Mc
MAYO_2	0.65 Mc	1.1 Mc	1.5 Mc	0.09 Mc	0.16 Mc

6.3. Comparison with Previous Works on Ring Signatures

Several works introduced post-quantum ring signatures in the past. They provide different tradeoffs in terms of public key size versus signature size. We consider the lattice-based ring signatures Falafl [BKP20], DualRing-LB [Yue+21], Raptor [LAZ19b] and Gandalf [GJK24b]. We additionally consider the isogeny-based ring signature Calamari [BKP20], although the comparison should be made cautiously as isogenies provide different tradeoffs than lattices (in the security assumption, compactness, efficiency).

None of these schemes are based on a standardized scheme, except for Raptor which is based on Falcon. We also note that Gandalf and Raptor only achieve a non-negligible anonymity of the order 2^{-30}, while they could be proven deniable as per Def. 9 – further demonstrating the applicability of our notion.

Only some of these works provide publicly available and optimized C implementations: Raptor [LAZ19b], Calamari and Falafl [BKP20]. DualRing-LB only provides a Python proof of concept[12], and Gandalf had no implementation at the time of writing this article[13].

We provide concrete size and performance numbers in Tab. 5 and Tab. 6. These tables, together with Tab. 2, Tab. 3, highlight that our proposals FalconRS and MayoRS are as compact as previous works based on non-standard schemes, and that our implementations largely outperform all previous available post-quantum ring signature implementations.

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, Katsumata, and Wiggers [HKW25, Sec. 6] also discussed bandwidth, they did not consider SignXKEM and only instantiated RingXKEM with Gandalf [GJK24b].

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

[12] https://github.com/thyuen/dualring.git

[13] It has since been implemented in [GHJ25], and their code is available at https://github.com/vincentvbh/shadowfax/blob/main/Gandalf/.

[14] https://github.com/zhenfeizhang/raptor/tree/37d78152bbca69b27b13a184347ceaec8ffbd69f

Table 5: Sizes of concurrent ring signature schemes, and whether they achieve a negligible anonymity for concretely selected parameters.

Scheme	PK	2-RSig	Negligible anonymity
Gandalf	896 B	1236 B	×
Raptor	900 B	2532 B	×
Calamari	64 B	3662 B	✓
DualRing-LB	2496 B	3877 B	✓
Falafl	4096 B	30 016 B	✓

Table 6: Performance of prior works on ring signatures, to compare with our instantiations from Tab. 3. Experiments executed on a Ryzen Pro 7 5850U @ 3GHz. Numbers are in Megacycles (Mc).

Scheme	Keygen	Sign 2-ring	Verify 2-ring
Raptor[14]	27.1 Mc	5 Mc	2 Mc
Calamari [BKP20]	119.5 Mc	46 581 Mc	41 250 Mc
Falafl [BKP20]	0.1 Mc	163 Mc	76 Mc

Table 7 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 [HKW25], the increase in server storage requirements for the prekey bundles is only about 1 kB in total. The RingXKEM instantiation based on NIST standard Falcon [Pre+22] is also close in bandwidth requirements to the instantiation based on Gandalf [GJK24b] relying on the trapdoor sampler of Antrag [Esp+23] and a preimage sampler of Mitaka [Esp+22].

Table 7: 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	Underlying Crypto
					Individual	L-key storage		
X3DH	✗	X25519	XEd25519 [Per16]	32	128	3296	64	Derived from X25519
PQXDH	✗	DH+Kyber-1024	XEd25519 [Per16]	32	1696	166 496	1632	Derived from X25519
RingXKEM	✓	Kyber-512	Gandalf [GJK24b]	1696	2582	81 526	2804	Based on Antrag and Mitaka
RingXKEM	✓	Kyber-1024	Gandalf [GJK24b]	2464	3350	158 326	4404	Based on Antrag and Mitaka
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 [Pre+22]	1697	1722	80 666	2202	NIST standard
SignXKEM	✓	Kyber-1024	Falcon [Pre+22]	2465	2490	157 466	3802	NIST standard
SignXKEM	✓	Kyber-512	MAYO_1 [Beu+24]	1968	1377	80 321	1857	NIST on-ramp R2
SignXKEM	✓	Kyber-1024	MAYO_1 [Beu+24]	2736	2145	157 121	3457	NIST on-ramp R2
SignXKEM	✓	Kyber-512	MAYO_2 [Beu+24]	6288	1236	80 180	1716	NIST on-ramp R2
SignXKEM	✓	Kyber-1024	MAYO_2 [Beu+24]	7056	2004	156 980	3316	NIST on-ramp R2

Acknowledgments

This paper is partially based on results obtained from a project, JPNP24003, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). We thank the USENIX reviewers for valuable feedbacks on the definition of strong deniability. We also thank Rune Fiedler for sharing his recent paper [FL25], leading us to notice that PQXDH is not HNJL deniable against malicious receivers; as well as Vadim Lyubashevsky and Felix Günther for pointing out an oversight in our proof of global deniability for RingXKEM, leading us to downgrade the permitted leakage for this setting. Lastly, we thank Phillip Gajland, Jonas Janneck, and Eike Kiltz for helping us better understand Gandalf.

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.

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 [BKP20; LAZ19b] adapted with our benchmark code. There are no other datasets or implementations relevant to this paper.

References

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. "Robust Encryption." In: *TCC 2010*. Ed. by Daniele Micciancio. Vol. 5978. LNCS. Springer, Berlin, Heidelberg, Feb. 2010, pp. 480–497. DOI: 10.1007/978-3-642-11799-2_28 (cit. on p. 28).

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. "Quantum Security Proofs Using Semi-classical Oracles." In: *CRYPTO 2019, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. LNCS. Springer, Cham, Aug. 2019, pp. 269–295. DOI: 10.1007/978-3-030-26951-7_10 (cit. on pp. 30, 34, 43, 44).

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. "1-out-of-n Signatures from a Variety of Keys." In: *ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. LNCS. Springer, Berlin, Heidelberg, Dec. 2002, pp. 415–432. DOI: 10.1007/3-540-36178-2_26 (cit. on pp. 20, 50).

[Bac+15] Michael Backes, Aniket Kate, Sebastian Meiser, and Tim Ruffing. "Secrecy Without Perfect Randomness: Cryptography with (Bounded) Weak Sources." In: *ACNS 2015*. Ed. by Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis. Vol. 9092. LNCS. Springer, Cham, June 2015, pp. 675–695. DOI: 10.1007/978-3-319-28166-7_33 (cit. on pp. 4, 8).

[Beu+24] 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 (cit. on pp. 4, 19, 22, 50).

[BF01] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing." In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Berlin, Heidelberg, Aug. 2001, pp. 213–229. DOI: 10.1007/3-540-44647-8_13 (cit. on p. 30).

[BGB04] 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 (cit. on p. 5).

[Bha+24] Karthikeyan Bhargavan, Charlie Jacomme, Franziskus Kiefer, and Rolfe Schmidt. "Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging." In: *USENIX Security 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, Aug. 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/bhargavan (cit. on p. 5).

[BK10] Zvika Brakerski and Yael Tauman Kalai. *A Framework for Efficient Signatures, Ring Signatures and Identity Based Encryption in the Standard Model*. Cryptology ePrint Archive, Report 2010/086. 2010. URL: https://eprint.iacr.org/2010/086 (cit. on p. 19).

[BKP20] Ward Beullens, Shuichi Katsumata, and Federico Pintore. "Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices." In: *ASIACRYPT 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 (cit. on pp. 21–23, 29).

[Bre+20] 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 (cit. on p. 5).

[Bre+22] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. "Post-quantum Asynchronous Deniable Key Exchange and the Signal Handshake." In: *PKC 2022, Part II*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Vol. 13178. LNCS. Springer, Cham, Mar. 2022, pp. 3–34. DOI: 10.1007/978-3-030-97131-1_1 (cit. on pp. 3, 5, 6, 8, 15).

[CCH23] 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 (cit. on p. 7).

[CF11] Cas Cremers and Michele Feltz. *One-round Strongly Secure Key Exchange with Perfect Forward Secrecy and Deniability*. Cryptology ePrint Archive, Report 2011/300. 2011. URL: https://eprint.iacr.org/2011/300 (cit. on p. 3).

[Col+24] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. "K-Waay: Fast and Deniable Post-Quantum X3DH without Ring Signatures." In: *USENIX Security 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, Aug. 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/collins (cit. on pp. 3, 5, 6, 8).

[CPZ20] Melissa Chase, Trevor Perrin, and Greg Zaverucha. "The Signal Private Group System and Anonymous Credentials Supporting Efficient Verifiable Encryption." In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1445–1459. DOI: 10.1145/3372297.3417887 (cit. on p. 15).

[Dag+13] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. "A Cryptographic Analysis of OPACITY - (Extended Abstract)." In: *ESORICS 2013*. Ed. by Jason Crampton, Sushil Jajodia, and Keith Mayes. Vol. 8134. LNCS. Springer, Berlin, Heidelberg, Sept. 2013, pp. 345–362. DOI: 10.1007/978-3-642-40203-6_20 (cit. on p. 3).

[DGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. "Deniable authentication and key exchange." In: *ACM CCS 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM Press, Oct. 2006, pp. 400–409. DOI: 10.1145/1180405.1180454 (cit. on pp. 3, 5, 7, 8).

[DMS04] 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. URL: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html (cit. on p. 6).

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. "Concurrent Zero-Knowledge." In: *30th ACM STOC*. ACM Press, May 1998, pp. 409–418. DOI: 10.1145/276698.276853 (cit. on p. 8).

[DP15] Léo Ducas and Thomas Prest. *Fast Fourier Orthogonalization*. Cryptology ePrint Archive, Report 2015/1014. 2015. URL: https://eprint.iacr.org/2015/1014 (cit. on p. 46).

[Dwo+06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating Noise to Sensitivity in Private Data Analysis." In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Berlin, Heidelberg, Mar. 2006, pp. 265–284. DOI: 10.1007/11681878_14 (cit. on pp. 4, 8).

[Esp+22] Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. "Mitaka: A Simpler, Parallelizable, Maskable Variant of Falcon." In: *EUROCRYPT 2022, Part III*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. LNCS. Springer, Cham, May 2022, pp. 222–253. DOI: 10.1007/978-3-031-07082-2_9 (cit. on pp. 19, 22).

[Esp+23] Thomas Espitau, Thi Thu Quyen Nguyen, Chao Sun, Mehdi Tibouchi, and Alexandre Wallet. "Antrag: Annular NTRU Trapdoor Generation - Making Mitaka as Secure as Falcon." In: *ASIACRYPT 2023, Part VII*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14444. LNCS. Springer, Singapore, Dec. 2023, pp. 3–36. DOI: 10.1007/978-981-99-8739-9_1 (cit. on pp. 19, 22).

[FG25] Rune Fiedler and Felix Günther. "Security Analysis of Signal's PQXDH Handshake." In: *PKC 2025, Part II*. Ed. by Tibor Jager and Jiaxin Pan. Vol. 15675. LNCS. Springer, Cham, May 2025, pp. 137–169. DOI: 10.1007/978-3-031-91823-0_5 (cit. on p. 5).

[FJ24] Rune Fiedler and Christian Janson. "A Deniability Analysis of Signal's Initial Handshake PQXDH." In: *Proceedings on Privacy Enhancing Technologies* 2024 (Oct. 2024), pp. 907–928. DOI: 10.56553/popets-2024-0148 (cit. on pp. 3, 5, 6, 8).

[FL25] Rune Fiedler and Roman Langrehr. "On Deniable Authentication against Malicious Verifiers." In: *To appear in proceedings of CRYPTO 2025: 45th Annual International Cryptology Conference, Santa Barbara, CA, USA*. Santa Barbara, CA, USA, 2025 (cit. on pp. 4, 5, 23).

[Gen+20] Nicholas Genise, Daniele Micciancio, Chris Peikert, and Michael Walter. "Improved Discrete Gaussian and Subgaussian Analysis for Lattice Cryptography." In: *PKC 2020, Part I*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12110. LNCS. Springer, Cham, May 2020, pp. 623–651. DOI: 10.1007/978-3-030-45374-9_21 (cit. on p. 48).

[GHJ25] Phillip Gajland, Vincent Hwang, and Jonas Janneck. *Shadowfax: A Deniability-Preserving AKEM Combiner*. Cryptology ePrint Archive, Paper 2025/154. 2025. URL: https://eprint.iacr.org/2025/154 (cit. on p. 21).

[GJK24a] Phillip Gajland, Jonas Janneck, and Eike Kiltz. *A Closer Look at Falcon*. Cryptology ePrint Archive, Paper 2024/1769. 2024. URL: https://eprint.iacr.org/2024/1769 (cit. on pp. 19, 45, 49, 50).

[GJK24b] Phillip Gajland, Jonas Janneck, and Eike Kiltz. "Ring Signatures for Deniable AKEM: Gandalf's Fellowship." In: *CRYPTO 2024, Part I*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14920. LNCS. Springer, Cham, Aug. 2024, pp. 305–338. DOI: 10.1007/978-3-031-68376-3_10 (cit. on pp. 4, 19–22).

[Goo22] Google. *Messages End-to-End Encryption Overview*. Technical paper. Feb. 2022. URL: https://www.gstatic.com/messages/papers/messages_e2ee.pdf (cit. on p. 3).

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions." In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206. DOI: 10.1145/1374376.1374407 (cit. on p. 50).

[Has+21] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. "An Efficient and Generic Construction for Signal's Handshake (X3DH): Post-Quantum, State Leakage Secure, and Deniable." In: *PKC 2021, Part II*. Ed. by Juan Garay. Vol. 12711. LNCS. Springer, Cham, May 2021, pp. 410–440. DOI: 10.1007/978-3-030-75248-4_15 (cit. on pp. 4, 6, 15, 18, 42).

[Has+22] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. "An Efficient and Generic Construction for Signal's Handshake (X3DH): Post-quantum, State Leakage Secure, and Deniable." In: *Journal of Cryptology* 35.3 (July 2022), p. 17. DOI: 10.1007/s00145-022-09427-1 (cit. on pp. 3, 5, 6, 8, 15, 42).

[HKP22] 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 (cit. on p. 6).

[HKW25] Keitaro Hashimoto, Shuichi Katsumata, and Thom Wiggers. "Bundled Authenticated Key Exchange: A Concrete Treatment of (Post-Quantum) Signal's Handshake Protocol." In: *USENIX Security 2025*. to appear. USENIX, Jan. 13, 2025. URL: https://eprint.iacr.org/2025/040 (cit. on pp. 1, 3–6, 12–16, 21, 22, 42).

[How+20] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. "Isochronous Gaussian Sampling: From Inception to Implementation." In: *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*. Ed. by Jintai Ding and Jean-Pierre Tillich. Springer, Cham, Apr. 2020, pp. 53–71. DOI: 10.1007/978-3-030-44223-1_4 (cit. on p. 47).

[Ica09] Thomas Icart. "How to Hash into Elliptic Curves." In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Berlin, Heidelberg, Aug. 2009, pp. 303–316. DOI: 10.1007/978-3-642-03356-8_18 (cit. on p. 30).

[Jia+22] Shaoquan Jiang, Yeow Meng Chee, San Ling, Huaxiong Wang, and Chaoping Xing. "A new framework for deniable secure key exchange." In: *Inf. Comput.* 285.PB (May 2022). ISSN: 0890-5401. DOI: 10.1016/j.ic.2022.104866. URL: https://doi.org/10.1016/j.ic.2022.104866 (cit. on p. 8).

[JSH24] Kee Jefferys, Maxim Shishmarev, and Simon Harman. *Session: End-To-End Encrypted Conversations With Minimal Metadata Leakage*. Tech. rep. Session, July 2024 (cit. on p. 6).

[KP05] Caroline Kudla and Kenneth G. Paterson. "Modular Security Proofs for Key Agreement Protocols." In: *ASIACRYPT 2005*. Ed. by Bimal K. Roy. Vol. 3788. LNCS. Springer, Berlin, Heidelberg, Dec. 2005, pp. 549–565. DOI: 10.1007/11593447_30 (cit. on pp. 3, 12).

[KS23] Ehren Kret and Rolfe Schmidt. *The PQXDH Key Agreement Protocol*. Protocol documentation. Oct. 18, 2023. URL: https://signal.org/docs/specifications/pqxdh/ (cit. on pp. 3, 5, 6).

[LAZ19a] Xingye Lu, Man Ho Au, and Zhenfei Zhang. *(Linkable) Ring Signature from Hash-Then-One-Way Signature*. Cryptology ePrint Archive, Report 2019/567. 2019. URL: https://eprint.iacr.org/2019/567 (cit. on p. 50).

[LAZ19b] Xingye Lu, Man Ho Au, and Zhenfei Zhang. "Raptor: A Practical Lattice-Based (Linkable) Ring Signature." In: *ACNS 2019*. Ed. by Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung. Vol. 11464. LNCS. Springer, Cham, June 2019, pp. 110–130. DOI: 10.1007/978-3-030-21568-2_6 (cit. on pp. 4, 19, 21, 23).

[LPS23] Helger Lipmaa, Roberto Parisella, and Janno Siim. "Algebraic Group Model with Oblivious Sampling." In: *TCC 2023, Part IV*. Ed. by Guy N. Rothblum and Hoeteck Wee. Vol. 14372. LNCS. Springer, Cham, Nov. 2023, pp. 363–392. DOI: 10.1007/978-3-031-48624-1_14 (cit. on p. 30).

[Lun18] Joshua Lund. *Technology preview: Sealed sender for Signal*. Oct. 2018. URL: https://signal.org/blog/sealed-sender/ (cit. on p. 6).

[Lyu12] Vadim Lyubashevsky. "Lattice Signatures without Trapdoors." In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Berlin, Heidelberg, Apr. 2012, pp. 738–755. DOI: 10.1007/978-3-642-29011-4_43 (cit. on pp. 47, 48, 50).

[Mad+22] 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. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/madathil (cit. on p. 6).

[Mer87] Ralph C. Merkle. "A Digital Signature Based on a Conventional Encryption Function." In: *CRYPTO'87*. Ed. by Carl Pomerance. Vol. 293. LNCS. Springer, Berlin, Heidelberg, Aug. 1987, pp. 369–378. DOI: 10.1007/3-540-48184-2_32 (cit. on p. 29).

[Met23] 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 (cit. on p. 3).

[Mir+09] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil P. Vadhan. "Computational Differential Privacy." In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Berlin, Heidelberg, Aug. 2009, pp. 126–142. DOI: 10.1007/978-3-642-03356-8_8 (cit. on pp. 4, 8).

[MP16] Moxie Marlinspike and Trevor Perrin. *The X3DH Key Agreement Protocol*. Protocol documentation. Nov. 4, 2016. URL: https://signal.org/docs/specifications/x3dh/ (cit. on pp. 3, 5, 6, 12).

[MR04] Daniele Micciancio and Oded Regev. "Worst-Case to Average-Case Reductions Based on Gaussian Measures." In: *45th FOCS*. IEEE Computer Society Press, Oct. 2004, pp. 372–381. DOI: 10.1109/FOCS.2004.72 (cit. on p. 48).

[Nik+19] 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 (cit. on p. 6).

[Nio25] 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 (cit. on p. 5).

[Pas03] Rafael Pass. "On Deniability in the Common Reference String and Random Oracle Model." In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Berlin, Heidelberg, Aug. 2003, pp. 316–337. DOI: 10.1007/978-3-540-45146-4_19 (cit. on pp. 8, 15).

[Per16] Trevor Perrin. *The XEdDSA and VXEdDSA Signature Schemes*. documentation. Oct. 20, 2016. URL: https://signal.org/docs/specifications/xeddsa/ (cit. on p. 22).

[PH10] 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 (cit. on p. 6).

[PM16] Trevor Perrin and Moxie Marlinspike. *The Double Ratchet Algorithm*. Protocol documentation. Nov. 20, 2016. URL: https://signal.org/docs/specifications/doubleratchet/ (cit. on p. 3).

[Pre15] Thomas Prest. "Gaussian sampling in lattice-based cryptography." Theses. Ecole normale supérieure - ENS PARIS, Dec. 2015. URL: https://theses.hal.science/tel-01245066 (cit. on pp. 46, 48).

[Pre17] Thomas Prest. "Sharper Bounds in Lattice-Based Cryptography Using the Rényi Divergence." In: *ASIACRYPT 2017, Part I*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. LNCS. Springer, Cham, Dec. 2017, pp. 347–374. DOI: 10.1007/978-3-319-70694-8_13 (cit. on pp. 47, 48).

[Pre+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. *FALCON*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022 (cit. on pp. 4, 9, 16, 19, 22, 49).

[Sho97] Victor Shoup. "Lower Bounds for Discrete Logarithms and Related Problems." In: *EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Berlin, Heidelberg, May 1997, pp. 256–266. DOI: 10.1007/3-540-69053-0_18 (cit. on pp. 15, 30).

[SV16] Igal Sason and Sergio Verdú. "f -Divergence Inequalities." In: *IEEE Transactions on Information Theory* 62.11 (2016), pp. 5973–6006. DOI: 10.1109/TIT.2016.2603151 (cit. on pp. 4, 9).

[TWG24] Elkana Tovey, Jonathan Weiss, and Yossi Gilad. "Distributed PIR: Scaling Private Messaging via the Users' Machines." In: *ACM CCS 2024*. Ed. by Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie. ACM Press, Oct. 2024, pp. 1967–1981. DOI: 10.1145/3658644.3670350 (cit. on p. 6).

[UG15] Nik Unger and Ian Goldberg. "Deniable Key Exchanges for Secure Messaging." In: *ACM CCS 2015*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. ACM Press, Oct. 2015, pp. 1211–1223. DOI: 10.1145/2810103.2813616 (cit. on p. 3).

[UG18] Nik Unger and Ian Goldberg. "Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging." In: *PoPETs* 2018.1 (Jan. 2018), pp. 21–66. DOI: 10.1515/popets-2018-0003 (cit. on p. 3).

[Ung+15] 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 (cit. on pp. 6, 7).

[Vat+20] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. "On the Cryptographic Deniability of the Signal Protocol." In: *ACNS 2020, Part II*. Ed. by Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi. Vol. 12147. LNCS. Springer, Cham, Oct. 2020, pp. 188–209. DOI: 10.1007/978-3-030-57878-7_10 (cit. on pp. 3, 5–8).

[WB19] Riad S. Wahby and Dan Boneh. "Fast and simple constant-time hashing to the BLS12-381 elliptic curve." In: *IACR TCHES* 2019.4 (2019), pp. 154–179. ISSN: 2569-2925. DOI: 10.13154/tches.v2019.i4.154-179. URL: https://tches.iacr.org/index.php/TCHES/article/view/8348 (cit. on p. 30).

[Wha23] WhatsApp. *WhatsApp Encryption Overview*. Technical white paper. Sept. 27, 2023. URL: https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf (cit. on p. 3).

[Yue+21] Tsz Hon Yuen, Muhammed F. Esgin, Joseph K. Liu, Man Ho Au, and Zhimin Ding. "DualRing: Generic Construction of Ring Signatures with Efficient Instantiations." In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 251–281. DOI: 10.1007/978-3-030-84242-0_10 (cit. on p. 21).

[Zha22] Mark Zhandry. "To Label, or Not To Label (in Generic Groups)." In: *CRYPTO 2022, Part III*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13509. LNCS. Springer, Cham, Aug. 2022, pp. 66–96. DOI: 10.1007/978-3-031-15982-4_3 (cit. on p. 30).

A. Basic Building Blocks

In this section, we provide definitions of basic cryptographic building blocks used to construct Signal handshake protocols.

A.1. Symmetric Key Encryption

We review the definition of symmetric key encryption (SKE).

Definition 10 (Symmetric Key Encryption)**.** A symmetric key encryption scheme SKE with secret key space $\mathcal{K}$ is given by two PPT algorithms SKE.Enc and SKE.Dec as follows:

$\mathsf{SKE.Enc}(K, m) \to \mathsf{ct}$: On input a secret key $K \in \mathcal{K}$ and a message m, outputs a ciphertext ct.
$\mathsf{SKE.Dec}(K, \mathsf{ct}) \to m/\perp$: On input a secret key K and a ciphertext ct, outputs a message m or $\perp$ indicating decryption failure.

Definition 11 (Correctness)**.** An SKE is correct if for all $K \in \mathcal{K}$ and messages m, we have $\mathsf{SKE.Dec}(K, \mathsf{SKE.Enc}(K, m)) = m$.

Definition 12 (IND-CPA Security)**.** We define the advantage of $\mathcal{A}$ against the IND-CPA security game as follows:

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\mathcal{A}}(1^\lambda) := \left| \Pr\left[b = b' \;\middle|\; \begin{array}{c} b \xleftarrow{\$} \{0,1\} \\ K \xleftarrow{\$} \mathcal{K}, \\ (m_0, m_1) \xleftarrow{\$} \mathcal{A}(1^\lambda); \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(K, m_b), \\ b' \xleftarrow{\$} \mathcal{A}(\mathsf{ct}) \end{array} \right] - \frac{1}{2} \right|.$$

An SKE is IND-CPA *secure* if the advantage is negligible for any efficient adversary $\mathcal{A}$.

We also require an SKE to be *robust* [ABN10], i.e., no ciphertext should validly decrypt under two distinct secret keys (even if the decryption results are distinct).

Definition 13 (Robustness)**.** We define the advantage of $\mathcal{A}$ against the robustness game as follows:

$$\mathsf{Adv}^{\mathsf{Robust}}_{\mathsf{SKE},\mathcal{A}}(1^\lambda) := \Pr\left[\begin{array}{c} K \neq K' \\ \wedge\ \mathsf{Dec}(K, \mathsf{ct}) \neq \perp \\ \wedge\ \mathsf{Dec}(K', \mathsf{ct}) \neq \perp \end{array} \middle| (\mathsf{ct}, K, K') \xleftarrow{\$} \mathcal{A}(1^\lambda) \right].$$

An SKE is *robust* if the advantage is negligible for any efficient adversary $\mathcal{A}$.

A.2. Key Encapsulation Mechanisms

We review the definition of key encapsulation mechanism (KEM).

Definition 14 (KEM)**.** A key encapsulation mechanism scheme KEM with session key space $\mathcal{K}$ consists of three PPT algorithms $(\mathsf{KEM.KeyGen}, \mathsf{KEM.Encaps}, \mathsf{KEM.Decaps})$, where:
$\mathsf{KEM.KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{ek}, \mathsf{dk})$: On input the security parameter 1^λ, outputs a pair of keys $(\mathsf{ek}, \mathsf{dk})$.
$\mathsf{KEM.Encaps}(\mathsf{ek}) \xrightarrow{\$} (\mathsf{ss}, \mathsf{ct})$: On input an encapsulation key ek, outputs a session key $\mathsf{ss} \in \mathcal{K}$ and a ciphertext ct.
$\mathsf{KEM.Decaps}(\mathsf{dk}, \mathsf{ct}) \to \mathsf{ss}$: On input a decapsulation key dk and a ciphertext ct, outputs a session key $\mathsf{ss} \in \mathcal{K}$.

Definition 15 (Correctness)**.** A KEM is correct if for all $\lambda \in \mathbb{N}$, it holds that $\Pr[\mathsf{KEM.Decaps}(\mathsf{dk}, \mathsf{ct}) = \mathsf{ss}] \geq 1 - \mathsf{negl}(\lambda)$, where we have $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{KEM.KeyGen}(1^\lambda)$ and $(\mathsf{ss}, \mathsf{ct}) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek})$.

Definition 16 (IND-ATK Security)**.** For $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, we define the advantage of $\mathcal{A}$ against the IND-ATK security game as follows:

$$\mathsf{Adv}^{\mathsf{IND\text{-}ATK}}_{\mathsf{KEM},\mathcal{A}}(1^\lambda) := \left| \Pr\left[b = b' \;\middle|\; \begin{array}{c} b \xleftarrow{\$} \{0,1\}, \\ (\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{KEM.KeyGen}(1^\lambda), \\ (\mathsf{ss}_0, \mathsf{ct}) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek}), \\ \mathsf{ss}_1 \xleftarrow{\$} \mathcal{K}, \\ b' \xleftarrow{\$} \mathcal{A}^{O_{\mathsf{ATK}}}(\mathsf{ek}, \mathsf{ss}_b, \mathsf{ct}) \end{array} \right] - \frac{1}{2} \right|,$$

where

$$O_{\mathsf{ATK}} = \begin{cases} \perp & \text{if } \mathsf{ATK} = \mathsf{CPA} \\ \mathsf{KEM.Decaps}(\mathsf{dk}, \cdot) & \text{if } \mathsf{ATK} = \mathsf{CCA} \end{cases}.$$

When $\mathsf{ATK} = \mathsf{CCA}$, $\mathcal{A}$ is not allowed to query the challenge ciphertext ct to O_{CCA}. A KEM is IND-ATK *secure* if the advantage is negligible for any efficient adversary $\mathcal{A}$.

A.3. Signature Schemes

We review the definition of digital signatures.

Definition 17 (Signature Schemes)**.** A signature scheme with message space $\mathcal{M}$ consists of the following three PPT algorithms $(\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$:
$\mathsf{Sig.KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{vk}, \mathsf{sk})$: On input the security parameter 1^λ, outputs a pair of keys $(\mathsf{vk}, \mathsf{sk})$.
$\mathsf{Sig.Sign}(\mathsf{sk}, \mathsf{M}) \xrightarrow{\$} \sigma$: On input a signing key sk and a message $\mathsf{M} \in \mathcal{M}$, outputs a signature σ.
$\mathsf{Sig.Verify}(\mathsf{vk}, \mathsf{M}, \sigma) \to \{0, 1\}$: On input a verification key vk, a message M and a signature σ, outputs a verification bit.

Definition 18 (Correctness)**.** A signature scheme is correct if for all $\lambda \in \mathbb{N}$ and messages $\mathsf{M} \in \mathcal{M}$, we have $\mathsf{Sig.Verify}(\mathsf{vk}, \mathsf{M}, \sigma) = 1$, where $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}(1^\lambda)$ and $\sigma \xleftarrow{\$} \mathsf{Sig.Sign}(\mathsf{sk}, \mathsf{M})$.

Definition 19 (EUF-CMA Security)**.** We define the advantage of $\mathcal{A}$ against the EUF-CMA security game as follows:

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG},\mathcal{A}}(1^\lambda) := \Pr\left[\begin{array}{c}\mathsf{Sig.Verify}(\mathsf{vk}, \mathsf{M}, \sigma) = 1 \\ \wedge\ \mathsf{M} \notin \mathcal{L}\end{array} \middle| \begin{array}{c}(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}(1^\lambda); \\ (\mathsf{M}, \sigma) \xleftarrow{\$} \mathcal{A}^{O_{\mathsf{Sign}}(\mathsf{sk},\cdot)}(\mathsf{vk})\end{array}\right]$$

where $O_{\mathsf{Sign}}(\cdot) := \mathsf{Sig.Sign}(\mathsf{sk}, \cdot)$ is the signing oracle and $\mathcal{L}$ is the set of messages that $\mathcal{A}$ submitted to O_{Sign}. A signature scheme is EUF-CMA-*secure* if the advantage is negligible for any efficient adversary $\mathcal{A}$.

A.4. Ring Signature Schemes

We recall standard correctness and unforgeability definitions for ring signatures.

Definition 20 (Correctness)**.** A ring signature is correct if for all $\lambda \in \mathbb{N}$, $N = \mathsf{poly}(\lambda)$, $j \in [N]$, and message M, we have $\mathsf{RS.Verify}(\mathsf{RL}, \mathsf{M}, \mathsf{sig}) = 1$, where for $i \in [N]$, $(\mathsf{rvk}_i, \mathsf{rsk}_i) \xleftarrow{\$} \mathsf{RS.KeyGen}(1^\lambda)$, $\mathsf{RL} := \{\mathsf{rvk}_1, \cdots, \mathsf{rvk}_N\}$, and $\mathsf{sig} \xleftarrow{\$} \mathsf{RS.Sign}(\mathsf{rsk}_j, \mathsf{M}, \mathsf{RL})$.

Definition 21 (Unforgeability)**.** We define the advantage of $\mathcal{A}$ against the unforgeability game as a game played between $\mathcal{A}$ and a challenger:

(i) The challenger generates key pairs $(\mathsf{rvk}_i, \mathsf{rsk}_i) \leftarrow \mathsf{RS.KeyGen}(1^\lambda)$ for $i \in [N]$. It sets $\mathsf{VK} := (\mathsf{rvk}_i)_{i\in[N]}$, initializes two empty sets SL and CL, and provides SL to $\mathcal{A}$;

(ii) $\mathcal{A}$ can make signing and corruption queries an arbitrary polynomial number of times:
- $(\mathsf{sign}, i, \mathsf{M}, \mathsf{RL})$: The challenger checks if $\mathsf{rvk}_i \in \mathsf{RL}$, and if so, computes the signature with $\mathsf{sig} \xleftarrow{\$} \mathsf{RS.Sign}(\mathsf{rsk}_i, \mathsf{M}, \mathsf{RL})$. The challenger provides sig to $\mathcal{A}$ and adds $(i, \mathsf{M}, \mathsf{RL})$ to SL;
- $(\mathsf{corrupt}, i)$: The challenger adds rvk_i to CL and returns rsk_i to $\mathcal{A}$.

(iii) $\mathcal{A}$ outputs $(\mathsf{RL}^*, \mathsf{M}^*, \mathsf{sig}^*)$. If $\mathsf{RL}^* \in \mathsf{VK}\backslash\mathsf{CL}$, $(\cdot, \mathsf{M}^*, \mathsf{RL}^*) \notin \mathsf{SL}$, and $\mathsf{RS.Verify}(\mathsf{RL}, \mathsf{M}, \mathsf{sig}) = 1$, then we say the adversary $\mathcal{A}$ wins.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{Unf}}_{\mathsf{RS},\mathcal{A}}(1^\lambda) = \Pr[\mathcal{A}\text{ wins}]$. A ring signature scheme RS is *unforgeable* if the advantage is negligible for any $N = \mathsf{poly}(\lambda)$ and efficient adversary $\mathcal{A}$.

A.5. Merkle Trees

A Merkle tree [Mer87] allows to prove membership in a set by hashing a list of elements $A = (a_0, \cdots, a_N)$ into one hash value root. At a later point, one can efficiently prove to a third party that an element a_i was included in the list A. We rely on a specific construction based on [BKP20] which allows us to hide the position in the list. Looking ahead, this allows us to hide the prekey bundle index being used, that is, to hide how many times a prekey bundle has been used.

Definition 22. A Merkle tree consists of PPT algorithms (MerkleTree, getMerklePath, ReconstructRoot) with access to a hash function $\mathcal{H}_{\mathsf{Coll}} : \{0,1\}^\star \to \{0,1\}^{2\lambda}$.

$\mathsf{MerkleTree}(A) \to (\mathsf{root}, \mathsf{tree})$**:** On input a list of (at most) 2^k elements $A = (a_1, \cdots, a_{2^k})$, with $k \in \mathbb{N}$, it constructs a binary tree of height k with $\{\ell_i = \mathcal{H}_{\mathsf{Coll}}(a_i)\}_{i\in[2^k]}$ as its leaf nodes, and where every internal node h with children h_{left} and h_{right} equals the hash digest of $(h_{\mathsf{left}}, h_{\mathsf{right}})_{\mathsf{lex}}$, where the subscript lex indicates the lexicographical order (or any other total order on binary strings).[15] It then outputs the root root of the Merkle tree, as well as a description of the entire tree tree.

$\mathsf{getMerklePath}(\mathsf{tree}, i) \to \mathsf{path}$**:** On input the description of a Merkle tree tree and an index $i \in [2^k]$, it outputs the list path, which contains the sibling of ℓ_i, as well as the sibling of any ancestor of ℓ_i, ordered by decreasing height.

$\mathsf{ReconstructRoot}(a, \mathsf{path}) \to \mathsf{root}$**:** On input an element a in the list of elements $A = (a_1, \cdots, a_{2^k})$ and $\mathsf{path} = (n_1, \cdots, n_k)$, it outputs a reconstructed root $\mathsf{root}' = h_k$, which is calculated by putting $h_0 = \mathcal{H}_{\mathsf{Coll}}(a)$ and defining h_i for $i \in [k]$ recursively as $h_i = \mathcal{H}_{\mathsf{Coll}}((h_{i-1}, n_i)_{\mathsf{lex}})$.

If the hash function $\mathcal{H}_{\mathsf{Coll}}$ that is used in the Merkle tree is collision resistant, then we have that the Merkle tree construction is binding. Formally, we have the following.

Lemma 1 (Binding)**.** *There is an efficient extractor algorithm that, given the description* tree *of a Merkle tree (having root* root *and constructed using the list of elements A) and* (b, path) *such that* $b \notin A$ *and* $\mathsf{ReconstructRoot}(b, \mathsf{path}) = \mathsf{root}$*, outputs a collision for the hash function* $\mathcal{H}_{\mathsf{Coll}}$*.*

Lastly, the use of the lexicographical order to concatenate two children nodes in the Merkle tree construction implies that the output path of the algorithm getMerklePath hides the index $i \in [N]$ given as input. As we do not formally use this in our work, we refer to [BKP20, Lemma 2.10] for more details.

[15] While it is standard to consider the concatenation $h_{\mathsf{left}}||h_{\mathsf{right}}$, this slight modification allows to show index hiding (cf. [BKP20, Lemma 2.10]).

B. Cryptographic Models

In this section, we give some details on cryptographic models used in proofs, namely the generic group model and the quantum random oracle model.

B.1. Generic Group Model with Oblivious Sampling

We recall the generic group model (GGM) by Shoup [Sho97] (i.e., random representation model) using the notations from Zhandry [Zha22]. Moreover, we extend the GGM with *oblivious sampling* following [LPS23]. This allows us to model adversaries that may sample group elements without knowing the corresponding discrete log.

Let $S \subset \{0,1\}^*$ be a set of strings of cardinality at least $p \in \mathbb{N}$. Let $\Xi : \mathbb{Z}_p \to S$ be a random injective function called the *labeling function*. Let $D : \mathbb{Z}_p \to \{0,1\}$ be an arbitrary efficiently sampleable distribution. All parties, including the adversary, the challenger, and subalgorithms, are able to make the following queries:

Labeling queries On input $x \in \mathbb{Z}_p$ from a party, it outputs the string $\Xi(x) \in S$.

Group operations On input $(s_1, s_2, a_1, a_2) \in S^2 \times \mathbb{Z}_p^2$ from a party, it first checks if there exists $x_1, x_2 \in \mathbb{Z}_p$ such that $\Xi(x_1) = s_1$ and $\Xi(x_2) = s_2$. If not it outputs $\perp$. Otherwise, it outputs $\Xi(a_1x_1 + a_2x_2) \in S$.

Oblivious sampling On invocation, it samples $x \xleftarrow{\$} D$ and outputs $\Xi(x) \in S$.

All queries are assumed to have unit costs. Above, $\Xi(x)$ represents g^x in the real world, where g is a fixed generator of the group. For instance, given $s = \Xi(x)$, a party can compute the representation of g^{ax} for some $a \in \mathbb{Z}_p$, by querying $(s, s, a/2, a/2)$ to the group operation oracle. For simplicity, we sometimes simply write that a party performs a group operation oracle query on input (s, a). Lastly, D allows us to model the party's knowledge of the exponent for an obliviously sampled group element. In this paper, for simplicity, we always assume D is the uniform distribution over $\mathbb{Z}_p$. However, in general, we only require D to be some distribution over $\mathbb{Z}_p$ with sufficient min-entropy.

Admissible encodings are one approach to oblivious sampling. They are efficiently computable functions E from $\mathbb{Z}_p$ to elliptic curve groups that are regular (small preimage sizes) and preimage sampleable (given an element in the image of E, one can efficiently recover its whole preimage). Concrete constructions for admissible encodings are given in e.g. [BF01; Ica09; WB19].

B.2. Quantum Random Oracle Model

We recall the result by Ambainis, Hamburg, and Unruh [AHU19]. As in the classical setting, it allows us to argue that if a reduction programs the (quantum) random oracle on a sufficiently random point, then an adversary learns nothing about the programmed point.

Definition 23 (Punctured oracle)**.** Let $S \subset \mathcal{X}$ be a set. Let $f_S : \mathcal{X} \to \{0,1\}$ be a predicate that returns 1 if and only if $x \in S$. A *punctured oracle* $\mathsf{H} \setminus S$ of $\mathsf{H} : \mathcal{X} \to \mathcal{Y}$ runs as follows: on input x, computes whether $x \in S$ in an auxiliary qubit, measures it, runs $\mathsf{H}(x)$, and returns the result. Let FIND denote the event that any of the measurement returns 1.

Lemma 2 (Semi-classical OW2H)**.** *Let $\mathsf{G}, \mathsf{H} : \mathcal{X} \to \mathcal{Y}$ be random functions, let z be a random value, and let $S \subset \mathcal{X}$ be a random set such that $\forall x \notin S, \mathsf{G}(x) = \mathsf{H}(x)$. $(\mathsf{G}, \mathsf{H}, S, z)$ may have arbitrary joint distribution. Then, for all quantum algorithms $\mathcal{A}$ on input z issuing at most q oracle queries, output either 0 or 1 with the following guarantee:*

$$\left|\Pr[1 \xleftarrow{\$} \mathcal{A}^{|\mathsf{G}\rangle}(z)] - \Pr[1 \xleftarrow{\$} \mathcal{A}^{|\mathsf{H}\rangle}(z)]\right| \leq 2 \cdot \sqrt{q \cdot \Pr[\mathsf{FIND} \mid b \xleftarrow{\$} \mathcal{A}^{|\mathsf{G}\setminus S\rangle}(z)]}. \quad (1)$$

In particular, if for each $x \in \mathcal{X}$, $\Pr[x \in S] \leq \epsilon$ (conditioned on z, on other oracles $\mathcal{A}$ has access to, and on other outputs of H), then we have

$$\Pr[\mathsf{FIND} \mid b \xleftarrow{\$} \mathcal{A}^{|\mathsf{G}\setminus S\rangle}(z)] \leq 4q \cdot \epsilon.$$

C. Combinations of Leakage and Disclosure

Table 8 depicts the combinations of leakage and disclosure which we consider.

Table 8: Secret information available to the distinguishing judge. $\mathcal{H}$ and C denote the accused and (honest-but-curious) accusing users, respectively.

$\mathcal{L}_{\mathsf{leak}}$ for h in $\mathcal{H}$	$\mathcal{D}_{\mathsf{disc}}$ for c in C	leak	disc	The judge gets:
$(\perp, \perp)$	$(\mathsf{isk}_c, \mathsf{st}_c, \mathsf{st}_c^{\mathsf{init}})$	$\perp$	high	No information from accused users. All accuser secrets, including initial state.
$(\mathsf{isk}_h, \perp)$	$(\mathsf{isk}_c, \mathsf{st}_c, \mathsf{st}_c^{\mathsf{init}})$	med	high	Accused users identity secret key. All accuser secrets, including initial state.
$(\mathsf{isk}_h, \mathsf{st}_h)$	$(\mathsf{isk}_c, \mathsf{st}_c, \mathsf{st}_c^{\mathsf{init}})$	high	high	Accused users identity secret key and state. All accuser secrets, including initial state.
$(\mathsf{isk}_h, \perp)$	$(\mathsf{isk}_c, \perp, \perp)$	med	low	Identity secret key of all users.
$(\mathsf{isk}_h, \mathsf{st}_h)$	$(\mathsf{isk}_c, \mathsf{st}_c, \perp)$	high	med	Identity secret key and states of all users.

D. 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 3 (Strong Local Deniability ⇒ (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 $((\vec{\mathsf{prek}}_u)_{u\in C}, \mathsf{st}_{\mathcal{A}} \coloneqq (\mathsf{isk}_u, \mathsf{st}_u^{\mathsf{init}} \coloneqq \mathsf{st}_u)_{u\in C})$. When $\mathcal{A}$ is invoked within oracle $O_{\text{strong-local}}(s, r)$ 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', $\mathsf{Adv}^{\text{strong-local}}_{\mathsf{D}',\mathcal{L}_{\mathsf{leak}}}(1^\lambda)$ 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 $\mathsf{st}_{\mathcal{A}} = (\mathsf{isk}_u, \mathsf{st}_u^{\mathsf{init}})_{u\in C}$. 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. □

E. Proof of Deniability of X3DH and PQXDH

In this section, we provide the proofs of varying levels of deniability attained by the X3DH and PQXDH protocols.

E.1. Local Deniability of X3DH and PQXDH

We prove that X3DH is locally deniable against honest-but-curious accusers (i.e., those who follow the protocol description) even if the distinguisher obtains all secret information of the users.

Lemma 4. *The X3DH protocol is* locally deniable *with respect to leakage function $\mathcal{L}_{\mathsf{high}}$, and disclosure function $\mathcal{D}_{\mathsf{high}}$.*

Proof. Let us first define the simulator $\mathsf{Sim} = (\mathsf{SimPreK}, \mathsf{SimTrans}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_{C})$:

$\mathsf{SimPreK}\left((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}\right)$**:** For $u \in C$, run $(\vec{\mathsf{prek}}_u, \mathsf{st}_u) \leftarrow \mathsf{X3DH.PreKeyBundleGen}(\mathsf{isk}_u)$, and output $((\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}, ((\vec{\mathsf{prek}}_u, \mathsf{st}_u))_{u\in C}, \mathsf{st}_{\mathsf{Sim}})$ with $\mathsf{st}_{\mathsf{Sim}} = ((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u, \mathsf{st}_u)_{u\in C})$. Below, for simplicity, we assume the simulator Sim always extracts the necessary public information from its state $\mathsf{st}_{\mathsf{Sim}}$.

$\mathsf{SimTrans}(\mathsf{isk}, \mathsf{st}_{\mathsf{Sim}}, (s, r, t))$**:** Here, recall isk is either isk_s, isk_r, or $\perp$ depending on $s \in C$, $r \in C$, or $(s, r) \in \mathcal{H} \times \mathcal{H}$, respectively.

If $(s, r) \in \mathcal{H} \times C$: First sample $\mathsf{esk} \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathsf{epk} := [\mathsf{esk}]G$. Then extract the signed prekey secret spksec_r from $\mathsf{st}_r \in \mathsf{st}_{\mathsf{Sim}}$, and compute

$$(\mathsf{ss}_1, \mathsf{ss}_2, \mathsf{ss}_3, \mathsf{ss}_4) := \left([\mathsf{spksec}_r]\overline{\mathsf{ik}}_s, [\mathsf{esk}]\overline{\mathsf{ik}}_r, [\mathsf{esk}]\mathsf{spk}_r, [\mathsf{esk}]\mathsf{opk}_{r,t}\right),$$

where note that such st_r exists as $r \in C$. Finally, compute Alg. 5, Lns. 16 to 18 and output (K, ρ).

If $(s, r) \in C \times \mathcal{H}$: Simply run $(K, \rho) \xleftarrow{\$} \mathsf{X3DH.Send}(\mathsf{isk}_s, \mathsf{ik}_r, \mathsf{prek}_{r,t})$, and output (K, ρ).

$\mathsf{SimSt}_{\mathcal{H}}(\mathsf{isk}_r, \mathsf{st}_r)$**:** Parse $(D_{\mathsf{prek}}, D_{\rho_\perp}) \leftarrow \mathsf{st}_r$ and delete the last unused one-time state, i.e., let t be the smallest index s.t. $D_{\mathsf{prek}}[t] \neq \perp$. If $t \leq L$, let $D_{\mathsf{prek}}[t] \leftarrow \perp$. Output the updated st_r.

$\mathsf{SimSt}_{C}(\mathsf{st}_{\mathsf{Sim}})$**:** For $u \in C$, extract st_u from $\mathsf{st}_{\mathsf{Sim}}$, parse $(D_{\mathsf{prek}}, D_{\rho_\perp}) \leftarrow \mathsf{st}_u$ and delete ephemeral secrets associated to used prekeys, i.e., for $t \in [\mathsf{counter}_u]$, with $t \leq L$, let $D_{\mathsf{prek}}[t] \leftarrow \perp$. Output $(\mathsf{st}_u)_{u\in C}$.

Recall that querying $(s, r) \in \mathcal{H} \times \mathcal{H}$ is only allowed in the global deniability game (see App. E.2 for details), hence, it remains to analyze the distinguishing advantage of D given as auxiliary information $\mathsf{disc}_{\mathsf{D}} := \left((\mathsf{isk}_u)_{u\in C}, (\mathsf{st}_u^{\mathsf{init}})_{u\in C}\right)$ disclosed by accusing users, and $\mathsf{leak}_{\mathsf{D}} = ((\mathsf{isk}_u)_{u\in\mathcal{H}}, (\mathsf{st}_u)_{u\in\mathcal{H}}) = \mathcal{L}_{\mathsf{high}}\left((\mathsf{isk}_u)_{u\in\mathcal{H}}, (\mathsf{st}_u)_{u\in\mathcal{H}}\right)$. First, for the case $(s, r) \in \mathcal{H} \times C$, notice that $\mathsf{ss}_2, \mathsf{ss}_3, \mathsf{ss}_4$ are generated exactly as in the real world. Moreover, the simulated $\mathsf{ss}_1 = [\mathsf{spksec}_r]\overline{\mathsf{ik}}_s$ has the same distribution as the real world $\mathsf{ss}_1 = [\overline{\mathsf{isk}}_s]\mathsf{spk}_r$. Hence, the output of $\mathsf{SimTrans}$ is identical to real world outputs from D's view, even given $\mathsf{disc}_{\mathsf{D}}$ and $\mathsf{leak}_{\mathsf{D}}$. Next, for the case $(s, r) \in C \times \mathcal{H}$, notice that ρ output by $\mathsf{SimTrans}$ is computed exactly by the real sender algorithm. Moreover, by correctness of the $\mathsf{X3DH}$ protocol, the real receiver algorithm outputs the same key K output by $\mathsf{SimTrans}$. Finally, it is straightforward to see that user states updated by $\mathsf{SimSt}_{\mathcal{H}}$ and SimSt_{C} in simulated executions are consistent with real executions. Hence, no D has any distinguishing advantage as desired. This completes the proof. □

The only difference between PQXDH and $\mathsf{X3DH}$ is that PQXDH additionally has a KEM key $\mathsf{ss}_{\mathsf{KEM}}$ in the key derivation. As explained in Sec. 4.2, this does not affect local deniability of PQXDH.

Lemma 5. *The* PQXDH *protocol is* locally deniable *with respect to leakage function* $\mathcal{L}_{\mathsf{high}}$*, and disclosure function* $\mathcal{D}_{\mathsf{high}}$*, under the correctness of the* KEM *scheme. Importantly, this holds even for classical accusers and* quantum *distinguishers.*

Proof. The proof is almost identical to that given for $\mathsf{X3DH}$ (cf. Lemma 4). Indeed, $\mathsf{SimPreK}$ is the same as before. The only difference is $\mathsf{SimTrans}$, where we also take the KEM scheme into consideration. However, since $\mathsf{KEM.Encaps}$ can be run publicly, the honest sender remains deniable and provides no distinguishing advantage to D. Moreover, by the correctness of the KEM scheme, the simulator does not need to have the honest receiver's decapsulation key to know the outcome of $\mathsf{KEM.Decaps}$. Specifically, adding a KEM to $\mathsf{X3DH}$ does not impact deniability. Since, assuming correctness of the underlying primitives, the simulation is perfect in both cases ($(s, r) \in C \times \mathcal{H}$, and $(s, r) \in \mathcal{H} \times C$), deniability holds for both classical and quantum distinguishers. This completes the proof. □

E.2. Global Deniability of X3DH

We prove that $\mathsf{X3DH}$ is globally deniable against honest-but-curious accusers. The protocol remains deniable even against distinguishers obtaining all secret information of the user if the one-time prekeys are never depleted. Otherwise, if the last-resort prekey bundle is used, then it remains deniable against distinguishers only obtaining the long-term identity secret key of the accused (i.e., $\mathsf{leak} = \mathsf{med}$). It is worth mentioning that *only* the sessions between honest senders and receivers *using the last-resort prekey bundle* become undeniable against full leakage — sessions using one-time prekey bundles remain deniable as expected.

Lemma 6. *Assume the key derivation function* KDF *is modeled as a random oracle and the* DDH *assumption holds. Then, the X3DH protocol is* globally deniable *for honest users* $\mathcal{H} \subseteq \mathcal{N}$ *against* honest-but-curious *accusers* $C := \mathcal{N}\backslash\mathcal{H}$ *with respect to disclosure function* $\mathcal{D}_{\mathsf{high}}$*, and leakage function* $\mathcal{L}_{\mathsf{leak}}$ *for* $\mathsf{leak} = \mathsf{high}$ *if the one-time prekey bundles are never depleted; and for* $\mathsf{leak} = \mathsf{med}$ *otherwise.*

Proof. Let us first define the simulator $\mathsf{Sim} = (\mathsf{SimPreK}, \mathsf{SimTrans}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_{C})$. It is defined identically to that of Lemma 4 with the only exception that we have to further specify the description of SimTrans on input $(s, r) \in \mathcal{H} \times \mathcal{H}$ as these cases were excluded in the local deniability game.

$\mathsf{SimTrans}(\mathsf{st}_{\mathsf{Sim}}, (s, r, t))$: As explained above, we only describe the simulation for $(s, r) \in \mathcal{H} \times \mathcal{H}$. Simply sample a random group element $\mathsf{epk} \xleftarrow{\$} \mathbb{G}$, sample random $K \| \tau_{\mathsf{conf}}$ from the range of the KDF, compute Alg. 5, Ln. 18, and output (K, K, ρ). Namely, skip computing the input to the KDF.

It remains to analyze the distinguishing advantage of the distinguisher D, where we consider two cases as in the lemma statement. Since we proved (perfect) local deniability when leak = high, and disc = high in Lemma 4, we only focus on the transcript indistinguishability when D queries $(s, r) \in \mathcal{H} \times \mathcal{H}$ to its oracle.

Case 1. Let us first consider the case where the one-time prekey bundles are not depleted: we always have a one-time prekey $\mathsf{opk}_{u,t}$ for $t \in [L]$ and the sender always computes the secret key K by including ss_4 in the KDF. In this case, D is given as auxiliary information $\mathsf{disc}_{\mathsf{D}} := \left((\mathsf{isk}_u)_{u \in C}, (\mathsf{st}_u^{\mathsf{init}})_{u \in C}\right)$ disclosed by accusing users, and $\mathsf{leak}_{\mathsf{D}} := \left((\mathsf{isk}_u)_{u \in \mathcal{H}}, (\mathsf{st}_u)_{u \in \mathcal{H}}\right) = \mathcal{L}_{\mathsf{high}}\left((\mathsf{isk}_u)_{u \in \mathcal{H}}, (\mathsf{st}_u)_{u \in \mathcal{H}}\right)$. The main observation is that once the receiver $r \in \mathcal{H}$ runs $\mathsf{X3DH.Receive}(\mathsf{isk}_r, \mathsf{st}_r, (\mathsf{ik}_s, t, \rho))$, the one-time prekey secret osk_t is deleted from its state st_r. Namely, D does not have $\mathsf{osk}_{r,t}$ nor esk to compute $\mathsf{ss}_4 = [\mathsf{esk}]\mathsf{opk}_{r,t} = [\mathsf{osk}_{r,t}]\mathsf{epk}$ after the session is established. We formalize this argument below.

For the sake of contradiction, assume D has a non-negligible advantage ϵ and makes at most Q_{Global} queries to the oracle O_{Global}. We first make a syntactical modification to the game when `mode = real`; recall that in this case, the game first ran $(K, \rho) \xleftarrow{\$} \mathsf{X3DH.Send}(\mathsf{isk}_s, (\mathsf{ik}_r, \mathsf{prek}_{r,t}))$ and $(K', \mathsf{st}_r) \xleftarrow{\$} \mathsf{X3DH.Receive}(\mathsf{isk}_r, \mathsf{st}_r, (\mathsf{ik}_s, t, \rho))$. We modify this so that the game no longer runs X3DH.Receive but simply outputs K instead of K' and an updated st_r, where $\mathsf{st}_r[t] \leftarrow \bot$ if $t \neq \bot$. Due to correctness, this does not alter the view of D. For the sake of explanation, we denote this artificial receive algorithm by $\mathsf{X3DH.Receive}'$ below. We also define a modified sender algorithm $\mathsf{X3DH.Send}'$ defined identical to X3DH.Send except that it samples ss_4 randomly in $\mathbb{G}$ as opposed to computing it as $[\mathsf{esk}]\mathsf{opk}_{r,t}$. Note that $\mathsf{X3DH.Receive}'$ runs as expected on the output of $\mathsf{X3DH.Send}'$, while X3DH.Receive would output $\bot$ with overwhelming probability.

Now, let us define $Q_{\mathsf{Global}} + 1$ hybrid games. We define $\mathsf{Game}_{Q_{\mathsf{Global}}+1}$ identical to the original global deniability game. The i-th ($i \in [Q_{\mathsf{Global}}]$) game Game_i is defined identical to Game_{i+1} except that when D queries O_{Global} for the $Q_{\mathsf{Global}} - i + 1$-th time, if the input is $(s, r) \in \mathcal{H} \times \mathcal{H}$, and if `mode = real`, the game computes the real transcript by running $\mathsf{X3DH.Send}'$ and $\mathsf{X3DH.Receive}'$ as opposed to running the original functions X3DH.Send and X3DH.Receive. Namely, in Game_1, $\mathsf{X3DH.Send}'$ and $\mathsf{X3DH.Receive}'$ are executed if `mode = real` and SimTrans is executed if `mode = sim`, when D queries O_{Global} on input $(s, r) \in \mathcal{H} \times \mathcal{H}$. Notice that by the definition of X3DH.Send, unless D queries the random oracle on input $\mathsf{ss}_1 \| \mathsf{ss}_2 \| \mathsf{ss}_3 \| \mathsf{ss}_4$, the views in `mode = real` and `sim` are identical. However, since ss_4 is sampled uniformly at random from $\mathbb{Z}_p$ and hidden from D, this cannot occur with all but negligible probability. Hence, the advantage of D in Game_1 is negligible. Then, by assumption, there must exist an index $i^* \in [Q_{\mathsf{Global}}]$ such that the advantage of D changes by more than a non-negligible fraction $\epsilon / Q_{\mathsf{Global}}$ between Game_{i^*} and Game_{i^*+1}. It remains to construct a DDH solver exploiting this fact.

Let $\mathcal{B}$ be an adversary against the DDH problem, given as input $(A, B, C) = ([a]G, [b]G, [c]G)$, where c is $c = ab$ if $b_{\mathsf{DDH}} = 0$ and $c \xleftarrow{\$} \mathbb{Z}_p$ if $b_{\mathsf{DDH}} = 1$. $\mathcal{B}$ first samples a challenge bit $b \xleftarrow{\$} \{0, 1\}$, random $(s^*, r^*, t^*) \xleftarrow{\$} \mathcal{H} \times \mathcal{H} \times [L]$, and prepares all the keys of the users as in $\mathsf{Game}_{Q_{\mathsf{Global}}+1}$ except that it sets the t^*-th one-time prekey opk_{r^*,t^*} of user r^* as A. $\mathcal{B}$ then simulates Game_{i^*+1} up till the $(Q_{\mathsf{Global}} - i^*)$-th query to O_{Global} using the keys it has prepared at the beginning of the game. When D queries O_{Global} for the $(Q_{\mathsf{Global}} - i^* + 1)$-th time on input (s, r), $\mathcal{B}$ checks if $(s, r) = (s^*, r^*)$ and the t^*-th one-time prekey bundle of the receiver r^* is going to be used. If not, it aborts the game. Otherwise, if $\mathcal{B}$ sampled $b = 0$ as the challenge bit, it first sets $\mathsf{epk} = B$ rather than sampling it on its own (see Alg. 5, Ln. 7). It then computes $\mathsf{ss}_1, \mathsf{ss}_2, \mathsf{ss}_3$ using the keys it sampled at the beginning of the game, where note that $\mathcal{B}$ knows all the discrete logarithms except for those of opk_{r^*,t^*} and epk. Finally, it sets $\mathsf{ss}_4 = C$, computes Alg. 5, Lns. 16 to 18, and outputs (K, ρ). The rest of the game is identical to Game_{i^*+1}. When D outputs a guess bit b', $\mathcal{B}$ outputs $[\![b = b']\!]$ as its guess for b_{DDH}.

It remains to analyze the advantage of $\mathcal{B}$. With probability $1/N^2L$, the guess (s^*, r^*, t^*) made by $\mathcal{B}$ is correct. Moreover, the above game simulated by $\mathcal{B}$ is identical to Game_{i^*+1} if $C = [ab]G$ and Game_{i^*} if $C = [c]G$ for a random $c \xleftarrow{\$} \mathbb{Z}_p$. From our assumption, since the advantage D has against Game_{i^*} and Game_{i^*+1} differs by a non-negligible amount $\epsilon / Q_{\mathsf{Global}}$, $\mathcal{B}$ solves the DDH problem with a non-negligible advantage of at least $\epsilon / (2N^2 L Q_{\mathsf{Global}})$. However, this contradicts the DDH assumption, thus completing the proof.

Case 2. The second case is where a last-resort prekey bundle is used. Unlike in Case 1, we can no longer embed the DDH problem into ss_4 as $\mathsf{opk}_{u,\bot} = \bot$. To this end, we restrict the leakage given to D as leak = med in order to embed the DDH problem into $\mathsf{ss}_3 = [\mathsf{esk}]\mathsf{spk}_r = [\mathsf{spksec}_r]\mathsf{epk}$ instead. Following an almost identical hybrid argument as in Case 1, we can prove that no distinguisher D can have non-negligible advantage in the global deniability game under the DDH assumption. □

E.3. Global Deniability of PQXDH

We prove two types of global deniability for PQXDH. While the accuser is always assumed to be classical, the distinguisher may be classical or quantum, where the latter captures HNJL deniability.

Lemma 7. *Assume the key derivation function* KDF *is modeled as a random oracle. Then, the* PQXDH *protocol is* globally deniable *with respect to disclosure function* $\mathcal{D}_{\mathsf{high}}$*, and leakage function* $\mathcal{L}_{\mathsf{leak}}$*, for:*

- leak = high *if one-time prekey bundles are never depleted; and* leak = med *otherwise, under the* DDH *assumption;*
- leak = high *if one-time prekey bundles are never depleted; and* leak = med *otherwise, under the correctness of the* KEM *scheme and the existence of an efficient* DL *solver.*

The latter case can handle deniability against a classical accuser and a quantum *distinguisher.*

Proof. The first (classical) part of the proof follows immediately from the proof of local deniability of the PQXDH protocol (cf. Lemma 5) and the proof of global deniability of the X3DH protocol (cf. Lemma 6). Indeed, from Lemma 5, we only need to specify the description of SimTrans on input $(s, r) \in \mathcal{H} \times \mathcal{H}$ as these cases were excluded in the local deniability game. An exact same proof following Lemma 6 suffices as deniability holds regardless of the KEM scheme, as anybody can produce a valid ciphertext.

For the HNJL part of the proof, where the distinguisher is quantum, local deniability is immediate. Indeed, the simulator's output is exactly the output of a real execution, no computational assumptions are required for deniability to hold. Hence, even a quantum D cannot distinguish real from simulated executions. For the case $(s, r) \in \mathcal{H} \times \mathcal{H}$, using the proof strategy of Lemma 6, we need to argue that a *quantum* D cannot guess the input $(\mathsf{ss} \,\|\mathsf{ss}_{\mathsf{KEM}}\,, \mathsf{content})$ to the KDF (modeled as a random oracle), with all but negligible probability. Observe that content is known to D; and a quantum D, who can compute discrete logarithms, could efficiently compute inputs $\mathsf{ss}_1, \mathsf{ss}_2, \mathsf{ss}_3$ (and ss_4, if $\mathsf{opk}_{r,t} \neq \perp$), and hence ss. However, so long as the KEM is proven post-quantum secure, $\mathsf{ss}_{\mathsf{KEM}}$ is indistinguishable from random from D's view, and can take any value in the KEM's session key space (which is exponentially large compared the number of oracle queries D can perform). Hence, D cannot query the random oracle on input $(\mathsf{ss} \,\|\mathsf{ss}_{\mathsf{KEM}}\,, \mathsf{content})$ with all but negligible probability. This holds both in the classical setting, and in the QROM, by applying the OW2H lemma [AHU19]. Indeed, let us denote q the maximum number of (quantum) oracle queries made by D, k the size of the KEM's session key space, and S_{BAD} the set of values $(\mathsf{ss} \,\|\mathsf{ss}_{\mathsf{KEM}}\,, \mathsf{content})$ for which D's view differs in the real and simulated execution modes. Note that $|S_{BAD}| \leq t$. Then for any input x, $\Pr[x \in S_{BAD}] \leq \frac{t}{k-q+1}$. Now, by applying Lemma 2, we can upper bound the probability p_{find} that D, given access to the quantum random oracle, queries an element in S_{BAD}: $p_{\mathsf{find}} \leq \frac{4qt}{k-q+1}$. This allows us to bound the probability that D can tell appart real and simulated executions in the QROM by $p_{\mathsf{dist}} \leq 4q\sqrt{\frac{t}{k-q+1}}$, which is negligible in the security parameter. □

E.4. Strong Local and Global Deniability of X3DH and PQXDH

Strong local deniability of X3DH and PQXDH almost directly follows from their global counterparts so we only provide their security statement below. As explained in Sec. 4.2, we can only show PQXDH is classically strong local deniable.

Lemma 8. *Assume the underlying group used by the X3DH and PQXDH protocols is a generic group of prime order p, and the key derivation function* KDF *is modeled as a random oracle. Then, the two protocols are* strongly local deniable *for honest users* $\mathcal{H} \subseteq \mathcal{N}$ *against* malicious *accusers* $\mathcal{C} \coloneqq \mathcal{N} \backslash \mathcal{H}$ *with respect to leakage function* $\mathcal{L}_{\mathsf{high}}$*. Importantly, the accusers and distinguisher are classical.*

Below, we prove that X3DH and PQXDH are (classically) strongly global deniable. The leakages supported are the same as for standard (non-strong) global deniability (see Apps. E.2 and E.3).

Lemma 9. *Assume the underlying group used by the X3DH and PQXDH protocols is a generic group of prime order p and the key derivation function* KDF *is modeled as a random oracle. Then, the two protocols are* strongly global deniable *for honest users* $\mathcal{H} \subseteq \mathcal{N}$ *against* malicious *accusers* $\mathcal{C} \coloneqq \mathcal{N} \backslash \mathcal{H}$ *with respect to leakage function* $\mathcal{L}_{\mathsf{leak}}$ *for* leak = high *if the one-time prekeys are never depleted and for* leak = med *otherwise. Importantly, the accusers and distinguisher are classical.*

Proof. We prove the statement in two parts: one proof for X3DH and another for PQXDH.

First half (X3DH). Let $\mathcal{A}$ be the malicious accuser. We first define the simulator $\mathsf{Sim}_{\mathcal{A}} = (\mathsf{SimPreK}_{\mathcal{A}}, \mathsf{SimTrans}_{\mathcal{A}}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_{\mathcal{A}})$. Recall in the GGM with oblivious sampling (see App. B.1), every party, including the simulator $\mathsf{Sim}_{\mathcal{A}}$ and the distinguisher D have access to the GGM oracles.

At a high level, $\mathsf{Sim}_{\mathcal{A}}$ will internally execute the accuser $\mathcal{A}$ and when $\mathcal{A}$ queries the GGM and random oracles, $\mathsf{Sim}_{\mathcal{A}}$ will simply relay them to its respective oracles. Assume the GGM oracle is queried a total of Q times by all parties, including the challenger of the game. Since $\mathsf{Sim}_{\mathcal{A}}$ can observe all the oracle queries made by $\mathcal{A}$, when $\mathcal{A}$ outputs a label $s \in S \subset \{0,1\}^*$, representing a group element $\mathbb{G}$, $\mathsf{Sim}_{\mathcal{A}}$ can efficiently compute the unique labels $(s_i, s'_i)_{i \in [Q']}$ for $Q' \leq Q$ and *non-zero* coefficients $(\alpha_i, \alpha'_i)_{i \in [Q']}$ in $\mathbb{Z}_p$ such that $\Xi^{-1}(s) = \sum_{i \in [Q']} (\alpha_i \cdot \Xi^{-1}(s_i) + \alpha'_i \cdot \Xi^{-1}(s'_i))$, where s_i (resp. s'_i) is an output of the GGM labeling (resp. oblivious sampling) oracle. Here, recall $\Xi : \mathbb{Z}_p \to S$ is the random injective function sampled within the GGM. Below, given a label $s \in S$ from $\mathcal{A}$, we call s a Good_L label if s does not contain any coefficients α'_i, and call it a Bad_L label otherwise. That is, if s contains labels output by the GGM oblivious sampling oracle, then we deem it a

Bad_L label, (informally) indicating that neither $\mathsf{Sim}_{\mathcal{A}}$ nor $\mathcal{A}$ knows $\Xi^{-1}(s) \in \mathbb{Z}_p$. Moreover, it is clear that for any Good_L label $s \in S$, $\mathsf{Sim}_{\mathcal{A}}$ can efficiently compute the unique $x \in \mathbb{Z}_p$ such that $\Xi(x) = s$. We will use these facts to describe the simulator $\mathsf{Sim}_{\mathcal{A}}$ below.

$\mathsf{SimPreK}_{\mathcal{A}}((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}})$**:** It first internally executes the adversary $((\vec{\mathsf{prek}}_u)_{u\in C}, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}})$. It then outputs $((\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}, (\vec{\mathsf{prek}}_u)_{u\in C}, \mathsf{st}_{\mathcal{A}}, \mathsf{st}_{\mathsf{Sim}})$ with $\mathsf{st}_{\mathsf{Sim}} = ((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in\mathcal{N}}, \mathsf{st}_{\mathcal{A}}, \mathsf{aux}_{\mathsf{Sim}})$, where $\mathsf{aux}_{\mathsf{Sim}}$ includes all the queries $\mathcal{A}$ has made to the GGM and random oracles so far. Below, we assume $\mathsf{aux}_{\mathsf{Sim}}$ is updated every time the adversary or the simulator makes oracle queries.

$\mathsf{SimTrans}_{\mathcal{A}}(\mathsf{isk}, \mathsf{st}_{\mathsf{Sim}}, (s, r, t))$**:** Here, recall isk is either isk_s, isk_r, or $\perp$ depending on $s \in C$, $r \in C$, or $(s, r) \in \mathcal{H} \times \mathcal{H}$, respectively.

If $(s, r) \in \mathcal{H} \times C$: First retrieve the prekey bundle $(\mathsf{spk}_r, \sigma_{\mathsf{spk}_r}, \mathsf{opk}_{r,t}) \leftarrow \mathsf{prek}_{r,t}$ of the malicious receiver, sample $\mathsf{esk} \xleftarrow{\$} \mathbb{Z}_p$ and query the GGM labelling oracle to receive $\mathsf{epk} := \Xi(\mathsf{esk}) \in S$. Then check if spk_r is a Good_L label using $\mathsf{aux}_{\mathsf{Sim}} \in \mathsf{st}_{\mathsf{Sim}}$. If so, compute the unique signed prekey secret $\mathsf{spksec}_r \in \mathbb{Z}_p$ such that $\Xi(\mathsf{spksec}_r) = \mathsf{spk}_r$. Then query the GGM group operation oracle on the tuples $(\overline{\mathsf{ik}}_s, \mathsf{spksec}_r)$, $(\overline{\mathsf{ik}}_r, \mathsf{esk})$, $(\mathsf{spk}_r, \mathsf{esk})$, and $(\mathsf{opk}_{r,t}, \mathsf{esk}) \in \mathbb{Z}_p \times S$ to obtain labels $(s_1, s_2, s_3, s_4) \in S^4$. Set

$$(\mathsf{ss}_1, \mathsf{ss}_2, \mathsf{ss}_3, \mathsf{ss}_4) := (s_1, s_2, s_3, s_4),$$

compute the remaining Alg. 5, Lns. 16 to 18, and output $(K, \rho, \mathsf{st}_{\mathsf{Sim}})$. Above, we note that if $\mathsf{opk}_{r,t} = \perp$ (i.e., the last-resort prekey bundle), then the simulator simply ignores generation of s_4.

Otherwise, if spk_r is a Bad_L label, Sample random $(K\|\tau_{\mathsf{conf}})$ from the range of the KDF, compute Alg. 5, Lns. 17 and 18, and output $(K, \rho, \mathsf{st}_{\mathsf{Sim}})$. Namely, skip computing the input to the KDF.

If $(s, r) \in C \times \mathcal{H}$: First execute the malicious sender internally $(\rho, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}(\mathsf{isk}_s, \mathsf{st}_{\mathcal{A}}, (s, r, t))$ using the accuser's state $\mathsf{st}_{\mathcal{A}} \in \mathsf{st}_{\mathsf{Sim}}$. Here, the simulator updates $\mathsf{st}_{\mathsf{Sim}}$ by storing the updated $\mathsf{st}_{\mathcal{A}}$. Then parse $(\mathsf{epk}, \tau_{\mathsf{conf}}) \leftarrow \rho$ and check if $\mathcal{A}$ ever made a random oracle query on input $(\mathsf{ss}, \mathsf{content})$, where $\mathsf{content} := \mathsf{ik}_s\|\mathsf{ik}_r\|\mathsf{prek}_{r,t}\|\mathsf{epk}$, such that $K\|\tau_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss}, \mathsf{content})$ using $\mathsf{aux}_{\mathsf{Sim}} \in \mathsf{st}_{\mathsf{Sim}}$. If not, simply output $(\perp, \rho, \mathsf{st}_{\mathsf{Sim}})$. Otherwise, retrieve the first such tuple (ss, K) under any canonical ordering and output $(K, \rho, \mathsf{st}_{\mathsf{Sim}})$.

If $(s, r) \in \mathcal{H} \times \mathcal{H}$: First sample $\mathsf{esk} \xleftarrow{\$} \mathbb{Z}_p$ and query the GGM labelling oracle to receive $\mathsf{epk} := \Xi(\mathsf{esk}) \in S$. Then sample random $(K\|\tau_{\mathsf{conf}})$ from the range of the KDF, compute Alg. 5, Lns. 17 and 18, and output (K, K, ρ). Namely, skip computing the input to the KDF.

$\mathsf{SimSt}_{\mathcal{H}}(\mathsf{isk}_r, \mathsf{st}_r)$**:** Parse $(D_{\mathsf{prek}}, D_{\rho_\perp}) \leftarrow \mathsf{st}_r$ and delete the last unused one-time state, i.e., let t be the smallest index s.t. $D_{\mathsf{prek}}[t] \neq \perp$. If $t \leq L$, let $D_{\mathsf{prek}}[t] \leftarrow \perp$. Output the updated st_r.

$\mathsf{SimSt}_{\mathcal{A}}(\mathsf{st}_{\mathsf{Sim}})$**:** Extract $\mathsf{st}_{\mathcal{A}}$ from $\mathsf{st}_{\mathsf{Sim}}$. Output $\mathsf{st}_{\mathcal{A}}$.

It remains to analyze the distinguishing advantage of D. Notice that there are only three cases in which the simulator deviates from the algorithms executed in the real world.

Case (i): When $(s, r) \in \mathcal{H} \times C$ and spk_r is a Bad_L label. Since $\mathcal{A}$ (and hence D) does not know spksec_r or esk, the Diffie-Hellman value ss_3 remains hidden. Specifically, the output of the KDF remains indistinguishable from random.

Case (ii): When $(s, r) \in C \times \mathcal{H}$ and $\mathcal{A}$ makes several random oracle queries on input $(\mathsf{ss}, \mathsf{content})$ such that $K\|\tau_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss}, \mathsf{content})$ for a given $\mathsf{content} := \mathsf{ik}_s\|\mathsf{ik}_r\|\mathsf{prek}_{r,t}\|\mathsf{epk}$. Note that if such an ss was never queried, the honest receiver will not accept the handshake message ρ as the verification of Alg. 6, Ln. 19 will fail (with all but negligible probability). In particular, the simulation becomes identical to the real world.

Case (iii): When $(s, r) \in \mathcal{H} \times \mathcal{H}$, $\mathcal{A}$ (and hence D) does not know esk. If the sender uses the last-resort prekey bundle, then assuming the state of accused users is not leaked to D, the Diffie-Hellman value ss_3 remains hidden. Note that this is the only case where leaking honest users' state will cause the simulation to fail, as it would allow D to recompute the input to the KDF from ρ. It could then trivially see that the simulator sampled $(K\|\tau_{\mathsf{conf}})$ randomly (as opposed to using the KDF). Hence, for global deniability, we can only simulate for leakage $\mathsf{leak} = \mathsf{med}$ if the one-time prekeys are depleted.

More formally, the fact that Case (ii) cannot impact the distinguisher D's advantage follows immediately from the fact that KDF is a random oracle. Indeed, if $\mathsf{ss} \neq \mathsf{ss}'$, and denoting that $K\|\tau_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss}, \mathsf{content})$, and that $K'\|\tau'_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss}', \mathsf{content})$, since KDF is a random oracle, it holds that $\tau_{\mathsf{conf}} \neq \tau'_{\mathsf{conf}}$ with overwhelming probability. Moreover, if the honest receiver accepts, then the malicious sender must have queried $(\mathsf{ss}, \mathsf{content})$ such that $K\|\tau_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss}, \mathsf{content})$. Indeed, due to the randomness of the KDF, for any fixed value of content, the probability that $\mathcal{A}$ can find an ss and τ_{conf} — without querying $\mathsf{KDF}(\mathsf{ss}, \mathsf{content})$ — satisfying $K^*\|\tau_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss}, \mathsf{content})$, for arbitrary K^*, is negligible. Hence, the simulator can output

the same session key K that will be output by the honest receiver. Finally, if the honest receiver does not accept (i.e., $K = \perp$), then the output of the simulator is identical to that of the accuser $\mathcal{A}$ in the real world.

It remains to formally prove that Case (i) and Case (iii) also cannot impact the advantage of D. To this end, we construct a reduction $\mathcal{R}$ that simulates the view of the distinguisher in the strong global deniability game. As typical with GGM based proofs, the reduction $\mathcal{R}$ simulates the game by using indeterminates $\overline{x}$ instead of $x \in \mathbb{Z}_p$ and encodes the "polynomial" $\overline{x} \in \mathbb{Z}_p[X]$ instead of x. Moreover, rather than preparing a random injective labeling function $\Xi : \mathbb{Z}_p \to S$, $\mathcal{R}$ samples a uniformly random label $s \in S$ on-the-fly, conditioned on that it has never been sampled. Note that the indeterminates are associated to the labels in S for which $\mathcal{A}$ and D do not know the corresponding discrete logarithm. Concretely, the reduction $\mathcal{R}$ maintains a list L where each tuple consists of a polynomial over $\mathbb{Z}_p$ and a label S. At the end of the game, $\mathcal{R}$ plugs in random points in $\mathbb{Z}_p$ into the indeterminates and computes all the polynomials. As long as the distinct polynomials do not compute to the same value, the simulation remains indistinguishable in the GGM. We finally prove that such a collision occurs with negligible probability by relying on the Schwartz–Zippel lemma.

More formally, there are only three ways in which a new indeterminate is created by $\mathcal{R}$.

1. When generating the one-time prekeys at the beginning of the game, $\mathcal{R}$ prepares, for $u \in \mathcal{H}$ and $t \in [L]$, a new indeterminate $\overline{\mathsf{osk}_{u,t}}$, samples a random label $\mathsf{opk}_{u,t} \xleftarrow{\$} S$ not included in the list L, and uses $\mathsf{opk}_{u,t}$ instead. At the time the distinguisher D receives $\mathsf{aux}_{\mathsf{D}}$, the reduction $\mathcal{R}$ samples $\mathsf{osk}_{u,t} \xleftarrow{\$} \mathbb{Z}_p$ and provides it to D if it has not been used yet (i.e., if it is not erased from state). It then evaluates all the polynomials in the list L by plugging in $\overline{\mathsf{osk}_{u,t}} = \mathsf{osk}_{u,t}$. It also prepares new indeterminates for the signed prekey spk_u and proceeds similarly.

2. When the adversary makes a GGM oblivious sampling query, $\mathcal{R}$ prepares a new indeterminate $\overline{x}$, samples a random label $s \xleftarrow{\$} S$ not included in the list L, and outputs s. It then updates the list $L \leftarrow L \cup (\overline{x}, s)$. By definition, s is a Bad_L label.

3. When the simulator is required to sample $\mathsf{esk} \xleftarrow{\$} \mathbb{Z}_p$ when $(s, r) \in \mathcal{H} \times C$ or $\mathcal{H} \times \mathcal{H}$, $\mathcal{R}$ instead first prepares a new indeterminate $\overline{\mathsf{esk}}$ and samples a random label $\mathsf{epk} \xleftarrow{\$} S$ not included in the list L. We assume the simulator continues its execution using $\overline{\mathsf{esk}}$ and the reduction simulates the GGM group operation queries accordingly.

In addition, the reduction $\mathcal{R}$ slightly alters the behavior of the simulator when $(s, r) \in \mathcal{H} \times C$ and spk_r is a Bad_L label, and when $(s, r) \in \mathcal{H} \times \mathcal{H}$. Recall that in these cases, the original simulator simply skipped computing the input to the KDF and sampled random $(K \| \tau_{\mathsf{conf}})$ from the range of the KDF. This reflected the fact that the simulator could not compute some of the Diffie-Hellman inputs to the KDF. In contrast, the reduction $\mathcal{R}$ will always compute this. For instance, when it needs to compute the Diffie-Hellman tuple between two Bad_L labels $\mathsf{opk}_{r,t}$ and spk_s, it first computes the polynomial $\overline{\mathsf{osk}_{r,t}} \cdot \overline{\mathsf{spk}_s}$, samples a random label $s \xleftarrow{\$} S$ not included in the list L, and uses s. It then updates the list $L \leftarrow L \cup (\overline{\mathsf{osk}_{r,t}} \cdot \overline{\mathsf{spk}_s}, s)$. It then runs the KDF on the computed labels. It is clear that unless the distinguisher D evaluates the KDF on the computed label, the reduction $\mathcal{R}$'s simulation is identical to the behavior of the simulator. This corresponds to the case where neither of Bad_L labels become Good_L labels once giving $\mathsf{aux}_{\mathsf{D}}$ to the distinguisher D. In particular, this proves that Case (i) and Case (iii) have negligible impact on D.

Lastly, at the end of the game, the reduction $\mathcal{R}$ samples random $x \xleftarrow{\$} \mathbb{Z}_p$ for all indeterminate $\overline{x}$, respectively, and plugs it in the polynomials in the list L. If any of the distinct polynomials in the list L evaluate to the same value, then $\mathcal{R}$ aborts. It can be checked that unless $\mathcal{R}$ aborts, it perfectly simulates the strong global deniability game. Moreover, noticing that all the polynomials in the list L are of degree at most two, the probability $\mathcal{R}$ aborts is bounded by $(2Q_{\mathcal{A}} + Q_{\mathsf{D}})^2/p$ due to the Schwartz-Zippel lemma, where $Q_{\mathcal{A}}$ and Q_{D} are the number of GGM oracle queries made by $\mathcal{A}$ and D, respectively. Since p is exponentially large, the probability is upper bounded by a negligible function. This completes the proof for $\mathsf{X3DH}$.

Second half (PQXDH). The proof for PQXDH is almost identical to above. Indeed, $\mathsf{SimPreK}$, $\mathsf{SimSt}_{\mathcal{H}}$ and $\mathsf{SimSt}_{\mathcal{A}}$ are the same as before. The only difference is $\mathsf{SimTrans}$, where we also take the KEM scheme into consideration. When the sender is honest, i.e., $s \in \mathcal{H}$, we can rely on the same proof as in the $\mathsf{X3DH}$ case since $\mathsf{KEM.Encaps}$ can be run publicly. It is only when the sender is malicious (i.e., $s \in C$) that $\mathsf{SimTrans}$ performs an extra check. This stems from the fact that a malicious sender may send a malformed ciphertext and the simulator must be able to simulate the honest receiver's behavior without the decapsulation key. While it is almost identical to the $\mathsf{SimTrans}$ constructed for the case of $\mathsf{X3DH}$, we provide its description for completeness.

$\mathsf{SimTrans}_{\mathcal{A}}(\mathsf{isk}, \mathsf{st}_{\mathsf{Sim}}, (s, r, t))$**:** As explained above, we only focus on $(s, r) \in C \times \mathcal{H}$. For this case, the simulation first executes the malicious sender internally $(\rho, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}(\mathsf{isk}_s, \mathsf{st}_{\mathcal{A}}, (s, r, t))$ using the accuser's state $\mathsf{st}_{\mathcal{A}} \in \mathsf{st}_{\mathsf{Sim}}$. Here, the simulator updates $\mathsf{st}_{\mathsf{Sim}}$ by storing the updated $\mathsf{st}_{\mathcal{A}}$. It then parses $(\mathsf{epk}, \mathsf{ct}, \tau_{\mathsf{conf}}) \leftarrow \rho$ and checks if $\mathcal{A}$ ever made a random oracle query on input $(\mathsf{ss} \| \mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$ where $\mathsf{content} := \mathsf{ik}_s \| \mathsf{ik}_r \| \mathsf{prek}_{r,t} \| \mathsf{epk} \| \mathsf{ct}$, such that it computed $K \| \tau_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss} \| \mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$ using $\mathsf{aux}_{\mathsf{Sim}} \in \mathsf{st}_{\mathsf{Sim}}$. If not, it simply outputs $(\perp, \rho, \mathsf{st}_{\mathsf{Sim}})$. Otherwise, it retrieves the first such tuple $(\mathsf{ss} \| \mathsf{ss}_{\mathsf{KEM}}, K)$ under any canonical ordering and outputs $(K, \rho, \mathsf{st}_{\mathsf{Sim}})$.

Following a similar argument to before, we can check that the above simulation cannot alter the advantage of the distinguisher D with all but a negligible probability. In particular, if the honest receiver accepts the handshake message ρ, then the malicious sender must have queried $(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$ such that $K\|\tau_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$. Indeed, due to the randomness of the KDF, for any fixed $\mathsf{content}$, the probability that $\mathcal{A}$ can find an $\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}$ and τ_{conf} — without querying $\mathsf{KDF}(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$ — that satisfy the computation $K^*\|\tau_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$, for arbitrary K^*, is negligible. Hence, the simulator can output the same session key K that will be output by the honest receiver. Note that the simulation does not (and can not, as it does not know the receiver secrets) check that, denoting $\mathsf{ss}^*_{\mathsf{KEM}} := \mathsf{KEM.Decaps}(\mathsf{dk}_{r,t}, \mathsf{ct})$, it holds that $\mathsf{ss}^*_{\mathsf{KEM}} = \mathsf{ss}_{\mathsf{KEM}}$. However, observe that if $\mathsf{ss}^*_{\mathsf{KEM}} \neq \mathsf{ss}_{\mathsf{KEM}}$, then, denoting $K^*\|\tau^*_{\mathsf{conf}} := \mathsf{KDF}(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$, with all but negligible probability, we will have $\tau^*_{\mathsf{conf}} \neq \tau_{\mathsf{conf}}$, and therefore the honest receiver would not accept (i.e., $K = \bot$). Finally, if the honest receiver does not accept, then the output of the simulator is identical to the accuser $\mathcal{A}$ in the real world (due to Alg. 2, Lns. 21 and 24). This completes the second half of the proof. □

E.5. Strong HNJL Deniability of PQXDH for Accused Receivers

We here prove the strong HNJL deniability of the protocol PQXDH. Since the leakage functions supported in the strong local and global settings are the same, we only focus on strong global deniability.

Lemma 10. *Assume the key derivation function* KDF *used by the* PQXDH *protocol is modeled as a random oracle, and the* KEM *scheme is correct. Then, the* PQXDH *protocol is* strongly deniable *for accused receivers, against* malicious *accusers, restricted to be classical senders, with respect to* quantum distinguishers *given access to leakage function* $\mathcal{L}_{\mathsf{high}}$*, under the assumption that the accuser only accesses the random oracle classically, and where:*

- *For local deniability,* $\mathsf{leak} = \mathsf{high}$*;*
- *For global deniability,* $\mathsf{leak} = \mathsf{high}$ *if one-time prekey bundles are never depleted; and* $\mathsf{leak} = \mathsf{med}$ *otherwise, under the correctness of the* KEM *scheme.*

Proof. Let $\mathcal{A}$ be the malicious accuser. We first define the simulator $\mathsf{Sim}_{\mathcal{A}} = (\mathsf{SimPreK}_{\mathcal{A}}, \mathsf{SimTrans}_{\mathcal{A}}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_{\mathcal{A}})$. At a high level, $\mathsf{Sim}_{\mathcal{A}}$ will internally execute the accuser $\mathcal{A}$ and when $\mathcal{A}$ queries the random oracle, $\mathsf{Sim}_{\mathcal{A}}$ will simply relay the query to its respective oracle. We describe $\mathsf{Sim}_{\mathcal{A}}$ below.

$\mathsf{SimPreK}_{\mathcal{A}}((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in\mathcal{C}}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}})$**:** First internally execute the adversary $((\vec{\mathsf{prek}}_u)_{u\in\mathcal{C}}, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in\mathcal{C}}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}})$. Finally, output $((\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{C}}, \mathsf{st}_{\mathcal{A}}, \mathsf{st}_{\mathsf{Sim}})$ with $\mathsf{st}_{\mathsf{Sim}} = ((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in\mathcal{N}}, \mathsf{st}_{\mathcal{A}}, \mathsf{aux}_{\mathsf{Sim}})$, where $\mathsf{aux}_{\mathsf{Sim}}$ includes all the queries $\mathcal{A}$ has made to the random oracle so far. Below, we assume $\mathsf{aux}_{\mathsf{Sim}}$ is updated every time the adversary or the simulator makes oracle queries.

$\mathsf{SimTrans}_{\mathcal{A}}(\mathsf{isk}, \mathsf{st}_{\mathsf{Sim}}, (s, r, t))$**:** Here, isk is either isk_s or $\bot$, since we only consider the case $r \in \mathcal{H}$.

If $(s, r) \in \mathcal{C} \times \mathcal{H}$: First execute the malicious sender internally $(\rho, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}(\mathsf{isk}_s, \mathsf{st}_{\mathcal{A}}, (s, r, t))$ using the accuser's state $\mathsf{st}_{\mathcal{A}} \in \mathsf{st}_{\mathsf{Sim}}$. Here, the simulator updates $\mathsf{st}_{\mathsf{Sim}}$ by storing the updated $\mathsf{st}_{\mathcal{A}}$. Then parse $(\mathsf{epk}, \mathsf{ct}, \tau_{\mathsf{conf}}) \leftarrow \rho$ and check if $\mathcal{A}$ ever made a random oracle query on input $(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$, where $\mathsf{content} := \mathsf{ik}_s\|\mathsf{ik}_r\|\mathsf{prek}_r\|\mathsf{epk}\|\mathsf{ct}$, such that $K\|\tau_{\mathsf{conf}} = \mathsf{KDF}(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$ using $\mathsf{aux}_{\mathsf{Sim}} \in \mathsf{st}_{\mathsf{Sim}}$. If not, simply output $(\bot, \rho, \mathsf{st}_{\mathsf{Sim}})$. Otherwise, under any canonical ordering, retrieve the first such tuple $(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, K, \tau_{\mathsf{conf}})$, and output $(K, \rho, \mathsf{st}_{\mathsf{Sim}})$.

If $(s, r) \in \mathcal{H} \times \mathcal{H}$: Since both users are honest, we can use the same simulation (and indistinguishability proof) here as in standard HNJL global deniability (see Lemma 7).

$\mathsf{SimSt}_{\mathcal{H}}(\mathsf{isk}_r, \mathsf{st}_r)$**:** Parse $(D_{\mathsf{prek}}, D_{\rho_\bot}) \leftarrow \mathsf{st}_r$ and delete the last unused one-time state, i.e., let t be the smallest index s.t. $D_{\mathsf{prek}}[t] \neq \bot$. If $t \leq L$, let $D_{\mathsf{prek}}[t] \leftarrow \bot$. Output the updated st_r.

$\mathsf{SimSt}_{\mathcal{A}}(\mathsf{st}_{\mathsf{Sim}})$**:** Extract $\mathsf{st}_{\mathcal{A}}$ from $\mathsf{st}_{\mathsf{Sim}}$. Output $\mathsf{st}_{\mathcal{A}}$.

It remains to analyze the distinguishing advantage of D. Regarding the simulation of prekeys, prekey bundles output by $\mathsf{SimPreK}_{\mathcal{A}}$ have exactly the same distribution as those output in the real world. It is also straightforward to see that the outputs of $\mathsf{SimSt}_{\mathcal{H}}$ and $\mathsf{SimSt}_{\mathcal{A}}$ in simulated executions are consistent with user states in real executions.

Regarding $\mathsf{SimTrans}_{\mathcal{A}}$, for $(s, r) \in \mathcal{H} \times \mathcal{H}$, we can use the same analysis as in Lemma 7. Then there is only one other situation in which the simulator deviates from the algorithms executed in the real world. This is when $(s, r) \in \mathcal{C} \times \mathcal{H}$ and $\mathcal{A}$ makes a random oracle query on input $(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$ such that $K\|\tau_{\mathsf{conf}} = \mathsf{KDF}(\mathsf{ss}\|\mathsf{ss}_{\mathsf{KEM}}, \mathsf{content})$ for a given $\mathsf{content} := \mathsf{ik}_s\|\mathsf{ik}_r\|\mathsf{prek}_r\|\mathsf{epk}\|\mathsf{ct}$, and where τ_{conf} is equal to that output by $\mathcal{A}$ in ρ. Note that if such an input was never queried, the honest receiver will not accept the handshake message ρ as the equality check on Alg. 6, Ln. 19 will fail (with all but negligible probability). In particular, the simulation becomes identical to the real world.

The formal proof that the above simulation cannot alter the advantage of the distinguisher D with all but a negligible probability is almost identical to that given for $\mathsf{X3DH}$ (cf. Lemma 9), and hence we here omit the details of the proof. □

E.6. PQXDH Modelling Gap

In our proposed BAKE model, all elements of a prekey bundle run out simultaneously. However, the actual specification of PQXDH is slightly different, as the last resort KEM key $\mathsf{ek}_{r,\perp}$ may be used *before* one runs out of one time prekeys $\mathsf{opk}_{r,t}$, and conversely, one may run out of $\mathsf{opk}_{u,t}$'s but still have one time prekeys $\mathsf{ek}_{r,t}$ available. Furthermore, a sender which receives ek has no way of knowing if it is a last resort $\mathsf{ek}_{r,\perp}$ or a one-time $\mathsf{ek}_{r,t}$. We here clarify our simplification in the model does not harm the deniability results for PQXDH.

PQXDH's global deniability against *classical* distinguishers is only affected if *both* one-time keys $\mathsf{opk}_{r,t}$ run out, *and* the last-resort $\mathsf{ek}_{r,\perp}$ is used. Otherwise, even with full leakage from the accused user, the input to the KDF cannot be predicted by the distinguisher, and the session key is indistinguishable from a random key. If *both* one-time components of the prekey bundle run out, global deniability holds if leak = med, since st_r contains the decapsulation key $\mathsf{dk}_{r,\perp}$ and the DH secret spksec_r (which is the secret leveraged in the DDH assumption when one runs out of keys $\mathsf{opk}_{r,t}$). This holds for both standard and strong deniability against classical distinguishers.

On the other hand, deniability against *quantum* distinguishers only relies on the distinguishers' inability to decapsulate the KEM. If the last resort KEM encapsulation key $\mathsf{ek}_{r,\perp}$ is used then HNJL deniability only holds if leak = med, since the client state contains the decapsulation key $\mathsf{dk}_{r,\perp}$. Running out of $\mathsf{opk}_{r,t}$ keys does not affect HNJL deniability.

Finally, we discuss the consequences of the sender not knowing if the receiver's ek is a last-resort key. Note that local deniability holds for the sender even if an accusing receiver purposefully uses their last resort key, as only the *global* deniability of PQXDH is affected by prekeys running out, i.e., when both the sender and the receiver are accused. In this context, one would expect that both parties, where possible, regularly upload fresh prekey bundles, and delete old states, to guarantee deniability.

F. Single to Multi-Challenge Deniability for Ring Signatures

In the following lemma we show that $(\overline{\mu}, \overline{\delta})$-deniability against adversaries making Q challenge queries follows from (μ, δ)-deniability where the adversary is limited to a single challenge query.

Lemma 11 (Single to Multi-Challenge Deniability)**.** *If a ring signature scheme* RS *is* (μ, δ)*-deniable against adversaries performing a single signing query, then* RS *is* $(\overline{\mu}, \overline{\delta})$*-deniable against adversaries performing* Q *queries, where* $\overline{\mu} = \mu^Q$*, and* $\overline{\delta} \leq \delta \cdot Q \cdot \mu^{Q-1}$.

Consequently, for deniability to hold in practice, using the NIST's recommended upper-bound $q_s = 2^{64}$ on the maximal number of signatures which may be queried from a single signer, it should hold that $\mu \leq 1 + q_s^{-1}$, so that $\mu^{q_s} \leq 2$. Note that it is intuitively natural that the adversary's success probability increases as it is allowed more queries, effectively extending its runtime. This does not diminish the "bit-security" of the scheme, which is generally understood as $\log(\frac{T}{\epsilon})$, where T is the adversary's runtime and ϵ its success probability. Intuitively, the multiplicative factor μ bounds the additional advantage gained from increased queries, ensuring that the scheme's overall security level remains consistent.

The proof of Lemma 11 is given below.

Proof. Let us consider the deniability game for ring signatures $\mathsf{Game}^{\mathsf{Deny}}_{\mathsf{RS},b,\mathcal{A}}(1^\lambda, Q)$, where the adversary can adaptively make Q queries to the challenge oracle, and thus get Q signatures signed with rsk_b. For $j \in \{0, \ldots, Q\}$ we define the hybrid game Hybrid_j, in which the adversary queries its signing oracle Q times and, for $0 \leq i \leq Q - j$, the i-th query is signed using rsk_0; whereas for $i \geq Q - j + 1$, the i-th query is signed using rsk_1. Namely, Hybrid_0 is identical to $\mathsf{Game}^{\mathsf{Deny}}_{\mathsf{RS},0,\mathcal{A}}(1^\lambda, Q)$ whereas Hybrid_Q is identical to $\mathsf{Game}^{\mathsf{Deny}}_{\mathsf{RS},1,\mathcal{A}}(1^\lambda, Q)$. Let $\mathcal{B}$ be an adversary against the single-query deniability of RS, and let $\mathcal{A}$ denote an adversary for the Q query deniability game. We first show that, under the single-query deniability of RS, $\mathcal{A}$ cannot distinguish whether it is in Hybrid_j or in Hybrid_{j+1}. Algorithm $\mathcal{B}$ receives as input a set of RS keypairs $(\mathsf{rvk}_u, \mathsf{rsk}_u)_{u \in [N]}$, which it forwards to $\mathcal{A}$. For $i \in [Q]$, consider the i-th query made by $\mathcal{A}$ to its signing oracle (simulated by $\mathcal{B}$). If $i < Q - j$ then $\mathcal{B}$ signs with identity key of index 0; if $i > Q - j$ then $\mathcal{B}$ signs with identity key of index 1; and for the $Q - j$-th query, $\mathcal{B}$ simply forwards the query to its own challenger, and sends the obtained signature back to $\mathcal{A}$.

Now if $\mathcal{A}$ outputs 0 (resp. 1), then $\mathcal{B}$ guesses that the RS signing key of index 0 (resp. of index 1) was used to sign its challenge signature, and outputs $b' := 0$ (resp. $b' := 1$). It follows that the probability $\mathcal{B}$ outputs 0 when its challenge bit was 0 (resp. 1) is exactly the probability that $\mathcal{A}$ outputs 0 in Hybrid_j (resp. Hybrid_{j+1}). Hence, by the (μ, δ)-deniability of RS in the single query setting, it holds that $\Pr[0 \xleftarrow{\$} \mathcal{A} \mid \mathsf{Hybrid}_j] \leq \mu \cdot \Pr[0 \xleftarrow{\$} \mathcal{A} \mid \mathsf{Hybrid}_{j+1}] + \delta$. By a simple recursion, we obtain:

$$\Pr[0 \xleftarrow{\$} \mathcal{A} \mid \mathsf{Hybrid}_0] \leq \mu^Q \cdot \Pr[0 \xleftarrow{\$} \mathcal{A} \mid \mathsf{Hybrid}_Q] + \delta \cdot Q \cdot \mu^{Q-1}.$$

Hence, RS is $(\mu^Q, \delta \cdot Q \cdot \mu^{Q-1})$-deniable against adversaries performing Q signing queries. □

G. Proofs of Deniability for RingXKEM

In this section, we provide the proofs of varying levels of deniability attained by the RingXKEM protocol.

G.1. Standard Local Deniability of RingXKEM

We here prove the standard local deniability of RingXKEM.

Lemma 12. *If the underlying ring signature scheme* RS *is* (μ, δ)-deniable*, then the* RingXKEM *protocol is* (μ, δ)-local deniable *with respect to leakage function* $\mathcal{L}_{\mathsf{high}}$*, and disclosure function* $\mathcal{D}_{\mathsf{high}}$.

Proof. Let us first define the simulator Sim:

$\mathsf{SimPreK}\left((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}\right)$**:** For $u \in C$, run the PreKeyBundleGen algorithm, with the only difference that on Ln. 13 of the algorithm, the simulator does not discard the ring signature secret key, i.e., it computes $(\widehat{\mathsf{rvk}}_u, \widehat{\mathsf{rsk}}_u) \xleftarrow{\$} \mathsf{RS.KeyGen}(1^\lambda)$. Note that this is the only BAKE protocol considered in this article for which we use the fact that SimPreK deviates from the description of PreKeyBundleGen. This being said, the resulting prekey bundles and user state are still computed exactly as in a real execution:

$$\left(\vec{\mathsf{prek}}_u \coloneqq (\mathsf{prek}_{u,t})_{t\in[L]\cup\{\perp\}}, \mathsf{st}_u \coloneqq (D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_u, D_{\rho_\perp})\right).$$

Finally, output $((\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}, (\vec{\mathsf{prek}}_u, \mathsf{st}_u)_{u\in C}, \mathsf{st}_{\mathsf{Sim}})$ with the latter value $\mathsf{st}_{\mathsf{Sim}} \coloneqq ((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u, \mathsf{st}_u)_{u\in C}, (\widehat{\mathsf{rvk}}_u, \widehat{\mathsf{rsk}}_u)_{u\in C})$. Below, for simplicity, we assume the simulator Sim always extracts the necessary public information from its state $\mathsf{st}_{\mathsf{Sim}}$.

$\mathsf{SimTrans}(\mathsf{isk}, \mathsf{st}_{\mathsf{Sim}}, (s, r, t))$**:** Here, recall isk is either isk_s or isk_r depending on $s \in C$ or $r \in C$, respectively.

If $(s, r) \in \mathcal{H} \times C$: The simulator extracts $\mathsf{isk}_r \coloneqq (\mathsf{dk}_r, \mathsf{rsk}_r)$ and $(\widehat{\mathsf{rvk}}_r, \widehat{\mathsf{rsk}}_r)$ from $\mathsf{st}_{\mathsf{Sim}}$. It runs Alg. 9, Lns. 7 to 10 as per the real sender algorithm (these steps do not require any sender secrets). It then uses the *receiver's* prekey ring signature secret key $\widehat{\mathsf{rsk}}_r$ to compute the ring signature $\sigma \xleftarrow{\$} \mathsf{RS.Sign}(\widehat{\mathsf{rsk}}_r, \mathsf{content}, \{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\})$. Finally, it computes Alg. 9, Lns. 12 and 13, as per the real sender algorithm, and outputs (K, ρ).

If $(s, r) \in C \times \mathcal{H}$: The simulator extracts isk_s from $\mathsf{st}_{\mathsf{Sim}}$. It then simply runs $(K, \rho) \xleftarrow{\$} \mathsf{Send}(\mathsf{isk}_s, \mathsf{ik}_r, \mathsf{prek}_{r,t})$ and outputs (K, ρ).

$\mathsf{SimSt}_{\mathcal{H}}(\mathsf{isk}_r, \mathsf{st}_r)$**:** Parse $(D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_r, D_{\rho_\perp}) \leftarrow \mathsf{st}_r$ and delete the last unused one-time state, i.e., let t be the smallest index s.t. $D_{\mathsf{kem}}[t] \neq \perp$. If $t \leq L$, let $D_{\mathsf{kem}}[t] \leftarrow \perp$. Output the updated st_r.

$\mathsf{SimSt}_C(\mathsf{st}_{\mathsf{Sim}})$**:** For $u \in C$, extract st_u from $\mathsf{st}_{\mathsf{Sim}}$, parse $(D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_u, D_{\rho_\perp}) \leftarrow \mathsf{st}_u$ and delete ephemeral secrets associated to used prekeys, i.e., for $t \in [\mathsf{counter}_u]$, with $t \leq L$, let $D_{\mathsf{kem}}[t] \leftarrow \perp$. Output $(\mathsf{st}_u)_{u\in C}$.

It remains to analyze the distinguishing advantage of the distinguisher D. Regarding the simulation of prekeys, it is easy to see that prekey bundles output by SimPreK have exactly the same distribution as those output by the real PreKeyBundleGen algorithm. The only difference is that the simulator stores ring signing keys $(\widehat{\mathsf{rsk}}_u)_{u\in C}$ in its state $\mathsf{st}_{\mathsf{Sim}}$ for future use. Since D does not have access to $\mathsf{st}_{\mathsf{Sim}}$, a real execution of PreKeyBundleGen is perfectly indistinguishable from the simulator's execution of SimPreK from D's view.

It is also straightforward to see that the outputs of $\mathsf{SimSt}_{\mathcal{H}}$ and SimSt_C in simulated executions are consistent with states updated in real executions.

Let us now consider the simulation of transactions. Since $\mathsf{leak} = \mathsf{high}$ and $\mathsf{disc} = \mathsf{high}$, D is given as auxiliary inputs $\mathsf{disc}_{\mathsf{D}} \coloneqq (\mathsf{isk}_u, \mathsf{st}_u^{\mathsf{init}})_{u\in C}$ disclosed by accusing users, and $\mathsf{leak}_{\mathsf{D}} \coloneqq (\mathsf{isk}_u, \mathsf{st}_u)_{u\in\mathcal{H}} = \mathcal{L}_{\mathsf{high}}\left((\mathsf{isk}_u)_{u\in\mathcal{H}}, (\mathsf{st}_u)_{u\in\mathcal{H}}\right)$.

If $(s, r) \in C \times \mathcal{H}$: Notice that ρ output by SimTrans is computed by the real sender algorithm. Moreover, by correctness of the RingXKEM protocol, with all but negligible probability, the real receiver algorithm outputs the same key K output by SimTrans. Hence, the output of SimTrans is identical to the real world from D's view, even given $\mathsf{disc}_{\mathsf{D}}$ and $\mathsf{leak}_{\mathsf{D}}$.

If $(s, r) \in \mathcal{H} \times C$: Observe that the only difference with a real execution of the protocol is that the simulator uses the receiver's key $\widehat{\mathsf{rsk}}_r$ instead of the sender's rsk_s to compute the ring signature σ. However — in both real and simulated executions — the signature is for the ring $\{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\}$, i.e. the verification keys associated to rsk_s and $\widehat{\mathsf{rsk}}_r$. Hence, by deniability of RS, this difference cannot be detected by D except with negligible probability. Let us formalize this argument.

Let $\mathcal{B}$ be an adversary against the $(\mu(\lambda, Q), \delta(\lambda))$-deniability of RS, and let D denote an algorithm attempting to distinguish $\mathsf{Game}^{\mathsf{local}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}}}(1^\lambda, \mathtt{real})$ from $\mathsf{Game}^{\mathsf{local}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{leak}}}(1^\lambda, \mathtt{sim})$. $\mathcal{B}$ samples a set of RS keypairs $((\mathsf{rvk}_1, \mathsf{rsk}_1), (\widehat{\mathsf{rvk}}_1, \widehat{\mathsf{rsk}}_1), \ldots,$

$(\mathsf{rvk}_N, \mathsf{rsk}_N), (\widehat{\mathsf{rvk}}_N, \widehat{\mathsf{rsk}}_N))$, and a challenge bit $\mathtt{mode} \xleftarrow{\$} \{\mathtt{real}, \mathtt{sim}\}$. It then prepares the identity keys and prekeys of all users as per Alg. 8, using the aforementioned ring signature keys (and storing the secret keys $\{\widehat{\mathsf{rsk}}_i\}_{i\in[N]}$).

For $i \in [Q]$, consider the i-th query made by D to $\mathcal{O}_{\mathsf{Local}}$. We denote the corresponding input to $\mathcal{O}_{\mathsf{Local}}$ as (s, r, t). If $(s, r) \in \mathcal{H} \times C$, then $\mathcal{B}$ computes $(\mathsf{ss}_r, \mathsf{ct}_r) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek}_r)$; $(\widehat{\mathsf{ss}}_{r,t}, \widehat{\mathsf{ct}}_{r,t}) \xleftarrow{\$} \mathsf{KEM.Encaps}(\widehat{\mathsf{ek}}_{r,t})$; and $\mathsf{content} := \mathsf{ik}_s \| \mathsf{ik}_r \| \mathsf{prek}_{r,t} \| \mathsf{ct}_r \| \widehat{\mathsf{ct}}_{r,t}$. Let us denote id_0 and id_1 the identities associated to rvk_s and $\widehat{\mathsf{rvk}}_r$ respectively. $\mathcal{B}$ then sends $(\mathsf{content}, \mathsf{id}_0, \mathsf{id}_1, \{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\})$ to its challenger, and receives $\sigma^* \xleftarrow{\$} \mathsf{RS.Sign}(\mathsf{rsk}_{\mathsf{id}_b}, \mathsf{content}, \{\mathsf{rvk}_s, \mathsf{rvk}_r\})$ for some randomly sampled challenge bit $b \in \{0, 1\}$.

Observe that if $\mathcal{B}$'s challenge bit b is 0 (resp. 1), then D's view is exactly that of $\mathsf{Game}^{\mathsf{local}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{high}}}(1^\lambda, \mathtt{real})$ (resp. $\mathsf{Game}^{\mathsf{local}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{high}}}(1^\lambda, \mathtt{sim})$). Finally, $\mathcal{B}$ outputs $b' := 0$ if D outputs $\mathtt{mode}' = \mathtt{real}$, and 1 if $\mathtt{mode}' = \mathtt{sim}$. Hence, $\Pr[b' = 0 \mid b = 1] = \Pr[\mathtt{mode}' = \mathtt{real} \mid \mathsf{Game}^{\mathsf{local}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{high}}}(1^\lambda, \mathtt{sim})]$ and $\Pr[b' = 0 \mid b = 0] = \Pr[\mathtt{mode}' = \mathtt{real} \mid \mathsf{Game}^{\mathsf{local}}_{\mathsf{D},\mathcal{H},\mathcal{L}_{\mathsf{high}}}(1^\lambda, \mathtt{real})]$. By the (μ, δ)-deniability of RS, we can conclude the proof, since, for $\beta \in \{0, 1\}$:

$$\Pr[b' = 0 \mid b = 0] \leq \mu(\lambda, Q) \cdot \Pr[b' = 0 \mid b = 1] + \delta(\lambda)$$
$$\Leftrightarrow \Pr[\mathtt{mode}' = \mathtt{real} \mid \mathtt{mode} = \mathtt{real}] \leq \mu(\lambda, Q) \cdot \Pr[\mathtt{mode}' = \mathtt{real} \mid \mathtt{mode} = \mathtt{sim}] + \delta(\lambda). \quad \square$$

G.2. Standard Global Deniability of RingXKEM

We here prove the standard global deniability of RingXKEM.

Lemma 13. *If the ring signature scheme* RS *is* (μ, δ)-deniable*, then the* RingXKEM *protocol is* (μ, δ)-global deniable *with respect to leakage function* $\mathcal{L}_{\mathsf{med}}$*, and disclosure function* $\mathcal{D}_{\mathsf{high}}$.

Proof. For global deniability we simulate prekey bundles of all users, so that the simulator knows the secret keys for the RS verification keys in prekey bundles. Let us first define the simulator $\mathsf{Sim} = (\mathsf{SimPreK}, \mathsf{SimTrans}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_C)$. We only need to consider the case $(s, r) \in \mathcal{H} \times \mathcal{H}$, since local deniability was proven in the previous section.

$\mathsf{SimPreK}\left((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}\right)$**:** We treat each user u different depending on their role.

> For $u \in C$: Run the RingXKEM.PreKeyBundleGen algorithm, with the only difference that on Ln. 13 of the algorithm, the simulator does not discard the ring signature secret key, i.e., it computes $(\widehat{\mathsf{rvk}}_u, \widehat{\mathsf{rsk}}_u) \xleftarrow{\$} \mathsf{RS.KeyGen}(1^\lambda)$. The resulting prekey bundle $(\vec{\mathsf{prek}}_u := (\mathsf{prek}_{u,t})_{t\in[L]\cup\{\perp\}}, \mathsf{st}_u := (D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_u, D_{\rho_\perp}))$ is thus computed exactly as in a real execution.
>
> For $u \in \mathcal{H}$: Parse the prekey bundle as $(\widehat{\mathsf{ek}}_{u,t}, \mathsf{path}_{u,t}, \mathsf{root}_u, \sigma_{u,\mathsf{root}}, \widehat{\mathsf{rvk}}_u)_{t\in[L]\cup\{\perp\}} \leftarrow \vec{\mathsf{prek}}_u$. Sample $(\widehat{\mathsf{rvk}}^*_u, \widehat{\mathsf{rsk}}^*_u) \xleftarrow{\$} \mathsf{RS.KeyGen}(1^\lambda)$, and replace $\widehat{\mathsf{rvk}}_u$ with $\widehat{\mathsf{rvk}}^*_u$, i.e. set $\widehat{\mathsf{rvk}}_u := \widehat{\mathsf{rvk}}^*_u$.
>
> Finally, output $((\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}, (\vec{\mathsf{prek}}_u, \mathsf{st}_u)_{u\in C}, \mathsf{st}_{\mathsf{Sim}})$ with $\mathsf{st}_{\mathsf{Sim}} := ((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u, \mathsf{st}_u)_{u\in C}, (\widehat{\mathsf{rvk}}_u, \widehat{\mathsf{rsk}}_u)_{u\in\mathcal{N}})$. Below, for simplicity, we assume the simulator Sim always extracts the necessary public information from its state $\mathsf{st}_{\mathsf{Sim}}$.

$\mathsf{SimTrans}(\mathsf{st}_{\mathsf{Sim}}, (s, r, t))$**:** We only consider $(s, r) \in \mathcal{H} \times \mathcal{H}$, hence the simulator has no user secrets. It first extracts $(\widehat{\mathsf{rvk}}_r, \widehat{\mathsf{rsk}}_r)$ from $\mathsf{st}_{\mathsf{Sim}}$, and runs Alg. 9, Lns. 7 to 10 as per the real sender algorithm (these steps do not require any sender secrets). It uses the *receiver's* prekey ring signature secret key $\widehat{\mathsf{rsk}}_r$ to compute the ring signature $\sigma \xleftarrow{\$} \mathsf{RS.Sign}(\widehat{\mathsf{rsk}}_r, \mathsf{content}, \{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\})$. Finally, it computes Alg. 9, Lns. 12 and 13, as per the real sender algorithm, and outputs (K, ρ).

It remains to analyze the distinguishing advantage of D when $(s, r) \in \mathcal{H} \times \mathcal{H}$.

Regarding the simulation of prekeys, we first observe that, since the honest receiver's state $\mathsf{st}_r := (D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_r, D_{\rho_\perp})$ contains the original (non-simulated) verification key $\widehat{\mathsf{rvk}}_r$, if the accused receiver's state leaks, the distinguisher can easily distinguish real and simulated executions. Hence, we restrict the leakage function to $\mathcal{L}_{\mathsf{med}}$, thereby ensuring that prekey bundles output by SimPreK have exactly the same distribution as those output by the real RingXKEM.PreKeyBundleGen algorithm. The only difference is that, via this simulation, the simulator is able to store the secret ring signing keys $(\widehat{\mathsf{rsk}}_u)_{u\in\mathcal{N}}$ in its state $\mathsf{st}_{\mathsf{Sim}}$ for future use. Since D does not have access to $\mathsf{st}_{\mathsf{Sim}}$, to it, a real execution of RingXKEM.PreKeyBundleGen is perfectly indistinguishable from the simulator's execution of SimPreK.

Let us now consider the simulation of transactions. Since $\mathsf{leak} = \mathsf{med}$ and $\mathsf{disc} = \mathsf{high}$, D is given as auxiliary inputs $\mathsf{disc}_{\mathsf{D}} := (\mathsf{isk}_u, \mathsf{st}^{\mathsf{init}}_u)_{u\in C}$ disclosed by accusing users, and $\mathsf{leak}_{\mathsf{D}} := (\mathsf{isk}_u)_{u\in\mathcal{H}} = \mathcal{L}_{\mathsf{med}}\left((\mathsf{ik}_u)_{u\in\mathcal{H}}, (\mathsf{st}_u)_{u\in\mathcal{H}}\right)$. We only need to consider the case $(s, r) \in \mathcal{H} \times \mathcal{H}$. Observe that the only difference with a real execution of the protocol is that the simulator uses the receiver's key $\widehat{\mathsf{rsk}}_r$ instead of the sender's rsk_s to compute the ring signature σ. However, — in both real and simulated executions — the signature is for the ring $\{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\}$, i.e. the verification keys associated to rsk_s and $\widehat{\mathsf{rsk}}_r$. By the deniability of RS, this difference cannot be detected by D except with negligible probability. The formal proof for this is essentially identical to that of local deniability when $s \in \mathcal{H}$, we thus omit the details of the proof here. $\square$

G.3. Strong Local and Global Deniability of RingXKEM for Accused Receivers

We here prove the strong local and global deniability of the protocol RingXKEM.

Lemma 14. *Assume the key derivation function* KDF *used by the* RingXKEM *protocol is modeled as a random oracle, the* KEM *scheme is correct, and the* SKE *scheme is correct and robust. Then, the* RingXKEM *protocol is* strongly local (resp. strongly global) deniable *for accused receivers, against* malicious *accusers, restricted to be senders, with respect to leakage function $\mathcal{L}_{\mathsf{high}}$ (resp. $\mathcal{L}_{\mathsf{med}}$), under the assumption that the accuser only accesses the random oracle classically.*

Proof. Let $\mathcal{A}$ be the malicious accuser. We first define the simulator $\mathsf{Sim}_{\mathcal{A}} = (\mathsf{SimPreK}_{\mathcal{A}}, \mathsf{SimTrans}_{\mathcal{A}}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_{\mathcal{A}})$. At a high level, $\mathsf{Sim}_{\mathcal{A}}$ will internally execute the accuser $\mathcal{A}$ and when $\mathcal{A}$ queries the random oracle, $\mathsf{Sim}_{\mathcal{A}}$ will simply relay the query to its respective oracle. We describe $\mathsf{Sim}_{\mathcal{A}}$ below.

$\mathsf{SimPreK}_{\mathcal{A}}((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}})$**:** First internally execute the adversary $((\vec{\mathsf{prek}}_u)_{u\in C}, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}})$. Then, for $u \in \mathcal{H}$, parse $(\widehat{\mathsf{ek}}_{u,t}, \mathsf{path}_{u,t}, \mathsf{root}_u, \sigma_{u,\mathsf{root}}, \widehat{\mathsf{rvk}}_u)_{t\in[L]\cup\{\perp\}} \leftarrow \vec{\mathsf{prek}}_u$.

If proving *global* deniability: For $u \in \mathcal{H}$, sample $(\widehat{\mathsf{rvk}}^*_u, \widehat{\mathsf{rsk}}^*_u) \xleftarrow{\$} \mathsf{RS.KeyGen}(1^\lambda)$, and replace $\widehat{\mathsf{rvk}}_u$ with $\widehat{\mathsf{rvk}}^*_u$, i.e. $\widehat{\mathsf{rvk}}_u := \widehat{\mathsf{rvk}}^*_u$. Let $\widehat{\mathsf{rsk}}_u := \widehat{\mathsf{rsk}}^*_u$.

Else (local): For $u \in \mathcal{H}$, let $\widehat{\mathsf{rsk}}_u := \perp$.

End if.

Finally, output $((\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}, (\vec{\mathsf{prek}}_u)_{u\in C}, \mathsf{st}_{\mathcal{A}}, \mathsf{st}_{\mathsf{Sim}})$ with $\mathsf{st}_{\mathsf{Sim}} = ((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in\mathcal{N}}, \mathsf{st}_{\mathcal{A}}, (\widehat{\mathsf{rvk}}_u, \widehat{\mathsf{rsk}}_u)_{u\in\mathcal{H}}, \mathsf{aux}_{\mathsf{Sim}})$, where $\mathsf{aux}_{\mathsf{Sim}}$ includes all the queries $\mathcal{A}$ has made to the random oracle so far. Below, we assume $\mathsf{aux}_{\mathsf{Sim}}$ is updated every time the adversary or the simulator makes oracle queries.

$\mathsf{SimTrans}_{\mathcal{A}}(\mathsf{isk}, \mathsf{st}_{\mathsf{Sim}}, (s, r, t))$**:** Here, isk is either isk_s or $\perp$, since we only consider the case $r \in \mathcal{H}$.

If $(s, r) \in C \times \mathcal{H}$: First execute the malicious sender internally $(\rho, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}(\mathsf{isk}_s, \mathsf{st}_{\mathcal{A}}, (s, r, t))$ using the accuser's state $\mathsf{st}_{\mathcal{A}} \in \mathsf{st}_{\mathsf{Sim}}$. Here, the simulator updates $\mathsf{st}_{\mathsf{Sim}}$ by storing the updated $\mathsf{st}_{\mathcal{A}}$. Then parse $(\mathsf{ct}_r, \widehat{\mathsf{ct}}_r, \mathsf{ct}_{\mathsf{ske}}) \leftarrow \rho$ and check if $\mathcal{A}$ ever made a random oracle query on input $(\mathsf{ss}_r \| \widehat{\mathsf{ss}}_{r,t}, \mathsf{content})$, where $\mathsf{content} := \mathsf{ik}_s \| \mathsf{ik}_r \| \mathsf{prek}_r \| \mathsf{ct}_r \| \widehat{\mathsf{ct}}_r$, such that $K \| K_{\mathsf{ske}} := \mathsf{KDF}(\mathsf{ss}_r \| \widehat{\mathsf{ss}}_{r,t}, \mathsf{content})$ using $\mathsf{aux}_{\mathsf{Sim}} \in \mathsf{st}_{\mathsf{Sim}}$. If not, simply output $(\perp, \rho, \mathsf{st}_{\mathsf{Sim}})$. Otherwise, under any canonical ordering, retrieve the first such tuple $(\mathsf{ss}_r, \widehat{\mathsf{ss}}_{r,t}, K, K_{\mathsf{ske}})$, for which, denoting $\sigma := \mathsf{SKE.Dec}(K_{\mathsf{ske}}, \mathsf{ct}_{\mathsf{ske}})$, it holds that $[\![\mathsf{RS.Verify}(\{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\}, \mathsf{content}, \sigma) = 1]\!]$. If none of the queried tuples satisfy this condition, then simply output $(\perp, \rho, \mathsf{st}_{\mathsf{Sim}})$. Otherwise, output $(K, \rho, \mathsf{st}_{\mathsf{Sim}})$.

If $(s, r) \in \mathcal{H} \times \mathcal{H}$: Since both users are honest, we can use the same simulation (and indistinguishability proof) here as in standard global deniability.

$\mathsf{SimSt}_{\mathcal{H}}(\mathsf{isk}_r, \mathsf{st}_r)$**:** Parse $(D_{\mathsf{kem}}, \widehat{\mathsf{rvk}}_r, D_{\rho_\perp}) \leftarrow \mathsf{st}_r$ and delete the last unused one-time state, i.e., let t be the smallest index s.t. $D_{\mathsf{kem}}[t] \neq \perp$. If $t \leq L$, let $D_{\mathsf{kem}}[t] \leftarrow \perp$. Output the updated st_r.

$\mathsf{SimSt}_{\mathcal{A}}(\mathsf{st}_{\mathsf{Sim}})$**:** Extract $\mathsf{st}_{\mathcal{A}}$ from $\mathsf{st}_{\mathsf{Sim}}$. Output $\mathsf{st}_{\mathcal{A}}$.

It remains to analyze the distinguishing advantage of D. Regarding the simulation of prekeys, for strong *local* deniability, prekey bundles output by $\mathsf{SimPreK}_{\mathcal{A}}$ have exactly the same distribution as those output in the real world. For strong *global* deniability, this holds so long as honest receiver states do not leak (as was the case for standard global deniability), i.e., $\mathsf{leak} = \mathsf{med}$. The only difference is that the simulator is able to store the secret keys $(\widehat{\mathsf{rsk}}_u)_{u\in\mathcal{H}}$ in its state $\mathsf{st}_{\mathsf{Sim}}$ for future use. Since D does not have access to $\mathsf{st}_{\mathsf{Sim}}$, a real execution of RingXKEM.PreKeyBundleGen is perfectly indistinguishable from the simulator's execution of $\mathsf{SimPreK}_{\mathcal{A}}$ from D's view. It is also straightforward to see that the outputs of $\mathsf{SimSt}_{\mathcal{H}}$ and $\mathsf{SimSt}_{\mathcal{A}}$ in simulated executions are consistent with user states in real executions.

Regarding $\mathsf{SimTrans}_{\mathcal{A}}$, notice that there are only two cases in which the simulator deviates from the algorithms executed in the real world.

Case (i) When $(s, r) \in C \times \mathcal{H}$ and $\mathcal{A}$ makes several random oracle queries on input $(\mathsf{ss}_r \| \widehat{\mathsf{ss}}_{r,t}, \mathsf{content})$ such that $K \| K_{\mathsf{ske}} := \mathsf{KDF}(\mathsf{ss}_r \| \widehat{\mathsf{ss}}_{r,t}, \mathsf{content})$ for a given $\mathsf{content} := \mathsf{ik}_s \| \mathsf{ik}_r \| \mathsf{prek}_r \| \mathsf{ct}_r \| \widehat{\mathsf{ct}}_r$. Note that if such an input was never queried, the honest receiver will not accept the handshake message ρ as, by robustness of the SKE, the decryption of Alg. 10, Ln. 15 will fail (with all but negligible probability). In particular, the simulation becomes identical to the real world.

Case (ii) When $(s, r) \in \mathcal{H} \times \mathcal{H}$, since neither of the parties is in C, they both follow the protocol description, and the same arguments as for standard global deniability allow arguing indistinguishability of real and simulated executions.

The formal proof that the above simulation cannot alter the advantage of the distinguisher D with all but a negligible probability is almost identical to that given for X3DH (cf. Lemma 9), and hence we here omit the details of the proof. □

G.3.1. Attack on Strong Sender Deniability

Hashimoto et al. noticed that there are plausible attacks on strong sender deniability [Has+22, Remark 7.11]. Namely, a malicious receiver accuser can craft a malicious RS verification key $\widehat{\mathsf{rvk}}_r$ such that it can later prove that it does not know the signing key. This means that if a sender produces a ring signature σ_S on the ring $\{\mathsf{rvk}_s, \widehat{\mathsf{rvk}}_r\}$, the receiver can later convince a judge that it could not have generated σ_S, thus breaking deniability. While it may be possible to thwart such attacks by including a proof of signing key possession in the prekey bundle, we do not consider them in our work due to the high overhead of such proofs.

Note that this attack does not extend to X3DH or PQXDH. Indeed, in X3DH and PQXDH, if the malicious receiver provides a pre-key for which it does not know the associated secret key, then it is not able to compute the input ss to the KDF, from which keys are derived. This means that the accuser cannot exchange encrypted messages with the accused sender, and that the distinguisher cannot distinguish a session key honestly generated by the sender from a randomly sampled session key. Conversely, in RingXKEM, the malicious receiver can generate its prekeys with honest KEM keys, but an RS public key for which the secret is unknown. The KEM decapsulation key allows for decryption of the input for the KDF, so not knowing the RS signing key does not prevent the judge from recomputing the session key output by the KDF, whereas it prevents the sender from denying the ring signature.

H. Deniability of SignXKEM

At PKC 2021, Hashimoto et al. proposed a post-quantum, signed Signal handshake protocol. As signatures typically provide the opposite of deniability (namely, non-repudiation), they suggested encrypting the signature to hide it from accusers [Has+21; Has+22]. In this section, we build on their proposal and develop the SignXKEM protocol. We add the Merkle Tree optimization described in [HKW25, Section 5.1]. We will first instantiate SignXKEM as a BAKE protocol using the syntax from Def. 1, after which we will discuss its deniability.

H.1. The SignXKEM protocol

SignXKEM uses a KEM KEM and a (plain) signature scheme Sig for its identity keys and prekeys. Additionally, it uses a symmetric encryption scheme SKE and a Merkle Tree. The functions SignXKEM.IdKeyGen, SignXKEM.PreKeyBundleGen, SignXKEM.Send, and SignXKEM.Receive are given by the Algs. 8 to 10 respectively.

The basic functionality of the SignXKEM 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 a signature to authenticate the sender. The signature is encrypted using the derived encryption key, which allows us to obtain deniability as described in App. H.2.

To authenticate the prekey bundle, one could simply sign it; however, as post-quantum signature schemes typically have large signatures, this leads to significant storage requirements. The Merkle tree optimization was first proposed for RingXKEM (discussed in Sec. 5) by Hashimoto, Katsumata, and Wiggers [HKW25]. By constructing a Merkle tree from all KEM public keys $\widehat{\mathsf{dk}}_{u,t}$, we can replace prekey bundles' individual signatures by the path to the root and a single signature on the root of the tree. This amortizes the storage cost of post-quantum signatures over all prekey bundles that are uploaded by a single call to SignXKEM.IdKeyGen. Note that although we include the Merkle tree path in our description of prekey bundles $\mathsf{prek}_{u,t}$, this path and the tree's root can be recomputed by the server from the uploaded prekey bundles (and thus do not need to be stored).

H.2. Summary: Deniability of SignXKEM

We summarize the level of deniability that SignXKEM satisfies. See Tab. 1 for a complete overview. The formal statements are provided in Apps. H.3 and H.4.

Local and global deniability. For SignXKEM, the permitted levels of leakage and disclosure allowing to achieve local and global deniability depend on whether the last resort prekey is used. In particular, standard local and global deniability hold for $(\mathsf{leak}, \mathsf{disc}) = (\mathsf{high}, \mathsf{med})$ if one-time prekeys never run out; otherwise, one must restrict to $(\mathsf{leak}, \mathsf{disc}) = (\mathsf{med}, \mathsf{low})$.

This is because, to ensure that the signature authenticating the sender remains hidden from the distinguisher D, we once again rely on the fact that the input to the KDF, modelled as a random oracle, cannot be guessed by D. If this holds, then D cannot recompute the key K_{ske} allowing to decrypt the SKE ciphertext and recover the signature. However, for SignXKEM, if D knows the receiver's *initial* state, it can easily recompute the inputs to the KDF, whereas if D only gets the *updated* states, and only one-time prekey bundles are used — whose states are deleted after use — then D cannot guess the KDF input with all but negligible probability. Hence, so long as one-time prekeys are not depleted, and the distinguisher only gets *updated states*, (i.e., $(\mathsf{leak}, \mathsf{disc}) = (\mathsf{high}, \mathsf{med})$) local and global deniability hold. On the other hand, if the last-resort prekey is used, the associated state is not deleted, and hence even the receiver's *updated* state would allow D to compute the KDF's input. In this case, deniability holds for $(\mathsf{leak}, \mathsf{disc}) = (\mathsf{med}, \mathsf{low})$.

Both local and global deniability of SignXKEM hold even if both the accuser and the distinguisher are *quantum*, so long as KEM is PQ secure.

Strong local and global deniability. As was the case for RingXKEM, for strong (local and global) deniability, we are only able to prove deniability for the *receiver*; this holds with respect to the leakage function $\mathcal{L}_{\mathsf{leak}}$ for $\mathsf{leak} = \mathsf{high}$ if the one-time prekey bundles are never depleted, and $\mathsf{leak} = \mathsf{med}$ otherwise. Again, as for RingXKEM, we can only prove strong deniability for the receiver in the *classical* ROM, while the malicious accuser and distinguisher may be quantum. The proof strategy is similar to that of strong deniability for PQXDH.

H.3. Local and Global Deniability of SignXKEM

We prove that SignXKEM is local and global deniable for weak disclosure functions (where the initial state of accusing users is not revealed). Since the disclosure and leakage functions supported in the local and global settings are the same, we only focus on global deniability. The non-trivial part of the proof is when the sender is honest. In this case, we must argue that the sender's signature implicitly included in the handshake message ρ cannot be later recovered by the distinguisher, as otherwise, deniability cannot hold. To this end, we use the fact that for weak disclosure functions (i.e., $\mathcal{D}_{\mathsf{disc}}$ with $\mathsf{disc} \in \{\mathsf{low}, \mathsf{med}\}$) the distinguisher only obtains the accuser's *updated* states. Specifically, once the (honest-but-curious) receiver deletes the decapsulation key included in the one-time prekey bundle, the handshake message from the honest sender should look random. This is why SignXKEM does not satisfy deniability for disclosure function $\mathcal{D}_{\mathsf{high}}$: given high, the distinguisher can decrypt the encrypted signatures.

Below, we rely on the OW2H lemma [AHU19] (see App. B.2 for definition) to formally prove this against a quantum accuser and distinguisher in the QROM.

Lemma 15. *Assume the key derivation function* KDF *is modeled as a random oracle and the* KEM *scheme is correct and* IND-CPA *secure, and the* SKE *scheme is correct,* IND-CPA *secure, and robust. Then, the* SignXKEM *protocol is* globally deniable *with respect to leakage function $\mathcal{L}_{\mathsf{leak}}$ and disclosure function $\mathcal{D}_{\mathsf{disc}}$ for* $\mathsf{leak} = \mathsf{high}$ *and* $\mathsf{disc} = \mathsf{med}$ *if the one-time prekey bundles are never depleted; and for* $\mathsf{leak} = \mathsf{med}$ *and* $\mathsf{disc} = \mathsf{low}$ *otherwise.*

Proof. Let us first define the simulator $\mathsf{Sim} = (\mathsf{SimPreK}, \mathsf{SimTrans}, \mathsf{SimSt}_{\mathcal{H}}, \mathsf{SimSt}_{C})$.

$\mathsf{SimPreK}\left((\mathsf{ik}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u)_{u\in C}, (\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}\right)$**:** For $u \in C$, run $(\vec{\mathsf{prek}}_u, \mathsf{st}_u) \leftarrow \mathsf{SignXKEM.PreKeyBundleGen}(\mathsf{isk}_u)$. Then simply output $((\vec{\mathsf{prek}}_u)_{u\in\mathcal{H}}, (\vec{\mathsf{prek}}_u, \mathsf{st}_u)_{u\in C}, \mathsf{st}_{\mathsf{Sim}})$ with $\mathsf{st}_{\mathsf{Sim}} = ((\mathsf{ik}_u, \vec{\mathsf{prek}}_u)_{u\in\mathcal{N}}, (\mathsf{isk}_u, \mathsf{st}_u)_{u\in C})$. Below, for simplicity, we assume the simulator Sim always extracts the necessary public information from its state $\mathsf{st}_{\mathsf{Sim}}$.

$\mathsf{SimTrans}(\mathsf{isk}, \mathsf{st}_{\mathsf{Sim}}, (s, r, t))$**:** Here, recall isk is either isk_s, isk_r, or $\perp$ depending on $s \in C$, $r \in C$, or $(s, r) \in \mathcal{H} \times \mathcal{H}$.

If $s \in \mathcal{H}$: It first runs $(\mathsf{ss}_r, \mathsf{ct}_r) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek}_r)$ and $(\mathsf{ss}_{r,t}, \mathsf{ct}_{r,t}) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek}_{r,t})$, and samples random keys (K, K_{ske}) from the range of the KDF. It then samples a random message M having the same length as a signature σ_s and runs $\mathsf{ct}_{\mathsf{ske}} \xleftarrow{\$} \mathsf{SKE.Enc}(K_{\mathsf{ske}}, \mathsf{M})$. Lastly, it outputs $(K, \rho := (\mathsf{ct}_r, \mathsf{ct}_{r,t}, \mathsf{ct}_{\mathsf{ske}}))$. Note that even if $r \in C$, the simulator does not require isk_r.

If $(s, r) \in C \times \mathcal{H}$: It simply runs $(K, \rho) \xleftarrow{\$} \mathsf{SignXKEM.Send}(\mathsf{isk}_s, \mathsf{ik}_r, \mathsf{prek}_{r,t})$, and outputs (K, ρ).

$\mathsf{SimSt}_{\mathcal{H}}(\mathsf{isk}_r, \mathsf{st}_r)$**:** Parse $(D_{\mathsf{kem}}, D_{\rho_\perp}) \leftarrow \mathsf{st}_r$ and delete the last unused one-time state, i.e., let t be the smallest index s.t. $D_{\mathsf{kem}}[t] \neq \perp$. If $t \leq L$, let $D_{\mathsf{kem}}[t] \leftarrow \perp$. Output the updated st_r.

$\mathsf{SimSt}_{C}(\mathsf{st}_{\mathsf{Sim}})$**:** For $u \in C$, extract st_u from $\mathsf{st}_{\mathsf{Sim}}$, parse $(D_{\mathsf{kem}}, D_{\rho_\perp}) \leftarrow \mathsf{st}_u$ and delete ephemeral secrets associated to used prekeys, i.e., for $t \in [\mathsf{counter}_u]$, with $t \leq L$, let $D_{\mathsf{kem}}[t] \leftarrow \perp$. Output $(\mathsf{st}_u)_{u\in C}$.

It remains to analyze the distinguishing advantage of the distinguisher D, where we consider two cases as in the lemma statement.

Case 1. Let us first consider the case where the one-time prekey bundles are not depleted: we always have a one-time prekey $\mathsf{prek}_{u,t}$ for $t \in [L]$. In this case, D is given as auxiliary information $\mathsf{disc}_{\mathsf{D}} := \left((\mathsf{isk}_u)_{u\in C}, (\mathsf{st}_u)_{u\in C}\right)$ disclosed by accusing users, and $\mathsf{leak}_{\mathsf{D}} := \left((\mathsf{isk}_u)_{u\in\mathcal{H}}, (\mathsf{st}_u)_{u\in\mathcal{H}}\right) = \mathcal{L}_{\mathsf{high}}\left((\mathsf{isk}_u)_{u\in\mathcal{H}}, (\mathsf{st}_u)_{u\in\mathcal{H}}\right)$.

If $(s, r) \in C \times \mathcal{H}$: notice that ρ output by SimTrans is computed by the real sender algorithm. Moreover, by correctness of the SignXKEM protocol, the real receiver algorithm outputs the same key K output by SimTrans with all but negligible probability. Hence, the output of SimTrans is identical to the real world to a distinguisher D even given $\mathsf{disc}_{\mathsf{D}}$ and $\mathsf{leak}_{\mathsf{D}}$.

If $s \in \mathcal{H}$: the main observation is that for disclosure function $\mathcal{D}_{\mathsf{med}}$, D only obtains the accuser's *updated* states. Specifically, whether r be in C or in $\mathcal{H}$, once the (honest-but-curious / honest) receiver runs $\mathsf{SignXKEM.Receive}(\mathsf{isk}_r, \mathsf{st}_r, (\mathsf{ik}_s, t, \rho))$, the one-time prekey secret $\mathsf{dk}_{r,t}$ is deleted from its state st_r. Namely, D does not have $\mathsf{dk}_{r,t}$ to compute $\mathsf{ss}_{r,t} := \mathsf{KEM.Decaps}(\mathsf{dk}_{r,t}, \mathsf{ct}_{r,t})$ after the session is established. We formalize this argument below.

For the sake of contradiction, assume D has a non-negligible advantage ϵ and makes at most Q_{Global} queries to the oracle O_{Global}. We first make a syntactical modification to the game when $\mathtt{mode} = \mathtt{real}$. In this mode, the game ran $(K, \rho) \xleftarrow{\$} \mathsf{SignXKEMSend}(\mathsf{isk}_s, (\mathsf{ik}_r, \mathsf{prek}_{r,t}))$ and $(K', \mathsf{st}_r) \xleftarrow{\$} \mathsf{SignXKEMReceive}(\mathsf{isk}_r, \mathsf{st}_r, (\mathsf{ik}_s, t, \rho))$. We modify this so that the game no longer runs $\mathsf{SignXKEMReceive}$ but simply outputs K instead of K' and an updated st_r, where $\mathsf{st}_r[t] \leftarrow \perp$ if $t \neq \perp$. Due to the correctness of $\mathsf{SignXKEM}$, the view of D is identical with all but negligible probability. We denote this artificial receive algorithm by $\mathsf{SignXKEMReceive}'$. We also define a modified sender algorithm $\mathsf{SignXKEMSend}'$ defined identical to $\mathsf{SignXKEMSend}$ except that, after computing $(\mathsf{ss}_{r,t}, \mathsf{ct}_{r,t}) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek}_{r,t})$, it samples $\mathsf{ss}'_{r,t}$ randomly over KEM's session key space and overwrites $\mathsf{ss}_{r,t}$ with $\mathsf{ss}'_{r,t}$. Note that $\mathsf{SignXKEMReceive}'$ runs as expected on the output of $\mathsf{SignXKEMSend}'$, while $\mathsf{SignXKEMReceive}$ would output $\perp$ with overwhelming probability.

Now, let us define $Q_{\mathsf{Global}} + 1$ hybrid games, where $\mathsf{Game}_{Q_{\mathsf{Global}}+1}$ is defined identical to the original global deniability game. The i-th ($i \in [Q_{\mathsf{Global}}]$) game Game_i is defined identical to Game_{i+1} except that when D queries O_{Global} for the $Q_{\mathsf{Global}} - i + 1$-th time, if $s \in \mathcal{H}$ and $\mathtt{mode} = \mathtt{real}$, the game computes the real transcript by running $\mathsf{SignXKEMSend}'$ and $\mathsf{SignXKEMReceive}'$ as opposed to running $\mathsf{SignXKEMSend}$ and $\mathsf{SignXKEMReceive}$. Namely, in Game_1, $\mathsf{SignXKEMSend}'$ and $\mathsf{SignXKEMReceive}'$ are executed if $\mathtt{mode} = \mathtt{real}$ and $\mathsf{SimTrans}$ is executed if $\mathtt{mode} = \mathtt{sim}$, when D queries O_{Global} on input a sender $s \in \mathcal{H}$.
Notice that by the definition of $\mathsf{SignXKEMSend}'$, unless D queries the random oracle on input $\mathsf{ss}_r \| \mathsf{ss}_{r,t}, \mathsf{context}$, the views in $\mathtt{mode} = \mathtt{real}$ and $\mathtt{sim}$ are identical. Indeed, assuming no such query to the random oracle is made by D, in both modes the keys (K, K_{ske}) are random in the range of the KDF. It follows that, from D's view, K_{ske} can take any value in the SKE key space. Furthermore, the robustness of SKE ensures that $\mathsf{ct}_{\mathsf{ske}}$ can only be decrypted using the same key K_{ske} as was used for encryption. Hence, D has negligible probability of guessing K_{ske} which would allow decrypting $\mathsf{ct}_{\mathsf{ske}}$, and, from the $\mathsf{IND\text{-}CPA}$ security of SKE, D has negligible advantage in distinguishing $\mathsf{ct}_{\mathsf{ske}}$ as computed when $\mathtt{mode} = \mathtt{real}$ and when $\mathtt{mode} = \mathtt{sim}$. Now, since $\mathsf{ss}_{r,t}$ is sampled uniformly at random from KEM's session key space (which is exponentially large compared to t and the number of oracle queries made by D) and hidden from D, D cannot query the random oracle on input $\mathsf{ss}_r \| \mathsf{ss}_{r,t}, \mathsf{context}$ with all but negligible probability. This holds both in the classical setting, and in the QROM, by applying the OW2H lemma [AHU19].

Indeed, let us denote q the maximum number of (quantum) oracle queries made by D, k the size of the KEM's session key space, and S_{BAD} the set of values $\mathsf{ss}_r \| \mathsf{ss}_{r,t}, \mathsf{context}$ for which D's view differs in the real and simulated execution modes. Note that $|S_{BAD}| \leq t$. Then for any input x, $\Pr[x \in S_{BAD}] \leq \frac{t}{k-q+1}$. Now, by applying Lemma 2, we can upper bound the probability p_{find} that D, given access to the quantum random oracle, queries an element in S_{BAD}: $p_{\mathsf{find}} \leq \frac{4qt}{k-q+1}$. This allows us to bound the probability that D can distinguish real and simulated executions in the QROM by $p_{\mathsf{dist}} \leq 4q\sqrt{\frac{t}{k-q+1}}$, which is negligible in the security parameter.

Hence, the advantage of D in Game_1 is negligible. Then, by assumption, there must exist an index $i^* \in [Q_{\mathsf{Global}}]$ such that the advantage of D changes by more than a non-negligible fraction $\epsilon / Q_{\mathsf{Global}}$ between Game_{i^*} and Game_{i^*+1}. It remains to construct an adversary breaking KEM's $\mathsf{IND\text{-}CPA}$ security exploiting this fact.

Let $\mathcal{B}$ be an adversary against the $\mathsf{IND\text{-}CPA}$ security of KEM, given as input $(\mathsf{ek}^*, (\mathsf{ss}^*, \mathsf{ct}_0^*))$, where $(\mathsf{ek}^*, \mathsf{dk}^*) \xleftarrow{\$} \mathsf{KEM.KeyGen}(1^\lambda)$; $(\mathsf{ss}^*, \mathsf{ct}_0^*) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek}^*)$ if the challenge bit $b_{\mathsf{KEM}} = 0$ and $\mathsf{ss}^* \xleftarrow{\$} \mathcal{K}$ if $b_{\mathsf{KEM}} = 1$.

$\mathcal{B}$ first samples a challenge bit $\mathtt{mode} \xleftarrow{\$} \{\mathtt{real}, \mathtt{sim}\}$, random $(s^*, r^*, t^*) \xleftarrow{\$} \mathcal{H} \times \mathcal{N} \times [L]$, and prepares all the keys of the users as in $\mathsf{Game}_{Q_{\mathsf{Global}}+1}$ except that it sets the t^*-th one-time prekey ek_{r^*,t^*} of user r^* as ek^*. $\mathcal{B}$ then simulates Game_{i^*+1} up till the $(Q_{\mathsf{Global}} - i^*)$-th query to O_{Global} using the keys it has prepared at the beginning of the game. When D queries O_{Global} for the $(Q_{\mathsf{Global}} - i^* + 1)$-th time on input (s, r), $\mathcal{B}$ checks if $(s, r) = (s^*, r^*)$ and the t^*-th one-time prekey bundle of the receiver r^* is going to be used. If not, it aborts the game. Otherwise, if $\mathcal{B}$ sampled $\mathtt{mode} = \mathtt{real}$, it sets $(\mathsf{ss}_{r^*,t^*}, \mathsf{ct}_{r^*,t^*}) := (\mathsf{ss}^*, \mathsf{ct}_0^*)$ rather than using the $\mathsf{KEM.Encaps}$ algorithm, computes Alg. 9, Lns. 7 and 9 to 14, and outputs (K, ρ). The rest of the game is identical to Game_{i^*+1}. When D outputs a guess bit $\mathtt{mode}'$, $\mathcal{B}$ outputs $[\![\mathtt{mode} = \mathtt{mode}']\!]$ as its guess for b_{KEM}.

It remains to analyze the advantage of $\mathcal{B}$. With probability $1/N^2L$, the guess (s^*, r^*, t^*) made by $\mathcal{B}$ is correct. Moreover, the above game simulated by $\mathcal{B}$ is identical to Game_{i^*+1} if $(\mathsf{ss}_0^*, \mathsf{ct}_0^*) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{ek}^*)$, and Game_{i^*} if $\mathsf{ss}_1^* \xleftarrow{\$} \mathcal{K}$. From our assumption, since the advantage that D has against Game_{i^*} and Game_{i^*+1} differs by a non-negligible amount $\epsilon / Q_{\mathsf{Global}}$, $\mathcal{B}$'s advantage in the $\mathsf{IND\text{-}CPA}$ security game of the KEM is at least $\epsilon / (2N^2 L Q_{\mathsf{Global}})$, which is non-negligible. However, this contradicts the assumption that KEM is $\mathsf{IND\text{-}CPA}$ secure, thus completing the proof for the case where one-time prekey bundles are not depleted.

Case 2. Let us now consider the case where the last resort prekey bundle is used. Since the last resort prekey secret $\mathsf{dk}_{r,\perp}$ is not deleted from r's state after having been used, if D were to obtain r's state st_r, it would be able to decapsulate the KEM keys and distinguish real and simulated executions. Consequently, if the last resort prekey bundle is used, global deniability is only proven to hold for leakage function $\mathcal{L}_{\mathsf{med}}$ and disclosure function $\mathcal{D}_{\mathsf{low}}$. Indeed, these do not reveal any user states, so $\mathsf{dk}_{r,\perp}$ remains unknown to D, and the same proof methodology as in *Case 1* can be applied.

This completes the proof. □

H.4. Strong Local and Global Deniability of SignXKEM for Accused Receivers

Since the leakage functions supported in the strong local and global settings are the same, we focus on global deniability.

It is clear that SignXKEM does not satisfy strong deniability since it is not even (standard) deniable if an accusing receiver's initial state is revealed to the distinguisher. However, it does satisfy some level of strong deniability for honest users that are receivers, that is, the accuser $\mathcal{A}$ is the sender (see Rem. 1). Thus, if one is using SignXKEM only as a receiver, then their deniability is guaranteed.

The proof for receiver deniability follows almost immediately from the proof given for strong local and global deniability of RingXKEM in App. G.3. To prove receiver deniability, the simulator must process the (possibly maliciously generated) handshake message $\rho = (\mathsf{ct}_r, \mathsf{ct}_{r,t}, \mathsf{ct}_{\mathsf{ske}})$ without the receiver's KEM decapsulation keys. The robustness of the SKE scheme guarantees that there is only one SKE key K_{ske} that properly decrypts $\mathsf{ct}_{\mathsf{ske}}$. Since the KDF is modeled as a random oracle, the simulator searches for any input with output $K\|K_{\mathsf{ske}}$. If so, it outputs K, so that the same K_{ske} is used by the sender and the receiver. Since the proof is almost identical to that of RingXKEM, we omit the proof of the following lemma statement. We note that similarly to RingXKEM, we limit the quantum accuser to only access the random oracle classically (see discussion at the end of Sec. 5.3).

Lemma 16. *Assume the key derivation function* KDF *is modeled as a quantum random oracle, the* KEM *scheme is correct, and the* SKE *scheme is correct and robust. Then, the* SignXKEM *protocol is* strongly global deniable *against* malicious *accusers, restricted to be senders, with respect to the leakage function* $\mathcal{L}_{\mathsf{leak}}$ *for* $\mathsf{leak} = \mathsf{high}$ *if the one-time prekey bundles are never depleted, and* $\mathsf{leak} = \mathsf{med}$ *otherwise, under the assumption that the accuser only accesses the random oracle classically.*

I. Omitted Details for Ring Signature Constructions

In this section, we include omitted details in the constructions of FalconRS and MayoRS.

I.1. Falcon-based Ring Signature

In order to formalize the deniability of FalconRS, we introduce more thoroughly the different components underlying Falcon, and recall useful preliminaries for lattices.

I.1.1. Lattices

In this section, we recall useful definitions and lemmas for instanciating Falcon. We follow notations from [GJK24a].

Max-log distance. We will make use of the max log distance between distributions through this section.

Definition 24 (Max-log distance)**.** Let $\mathcal{P}, Q$ two distributions over the same countable support $\mathsf{Supp}(\mathcal{P})$. The max log distance between $\mathcal{P}$ and Q is

$$\Delta_{\mathsf{ML}}(\mathcal{P}, Q) = \max_{x \in \mathsf{Supp}(\mathcal{P})} |\log \mathcal{P}(x) - \log Q(x)|$$

Linear algebra. We use lower (resp. upper) case bold fonts $\mathbf{v}$ (resp. $\mathbf{M}$) for vectors (resp. matrices). The vectors are in the column form and we use v_i (resp. $\mathbf{m}_i$) to indicate the i-th entry (resp. column) of $\mathbf{v}$ (resp. $\mathbf{M}$).

For $\mathbf{v} \in \mathbb{Z}^n$, we note $\|\mathbf{v}\|_2$ its euclidean norm. For $\mathbf{M} \in \mathbb{Z}^{m \times m}$, we note $\|\mathbf{M}\|_{\mathsf{GS}}$ the maximal 2-norm of the Gram-Schmidt orthogonalization of the column vectors of $\mathbf{M}$.

Throughout this work, for a fixed power-of-two n, we consider the polynomial ring $\mathcal{R} = \mathbb{Z}_q[x]/(x^n + 1)$. We also note $\mathcal{R}_q = \mathcal{R}/(q\mathcal{R})$. For a vector $\mathbf{u} \in \mathcal{R}_q^\ell$, $\overline{\mathbf{u}} \in \mathbb{Z}^{n\ell} = \begin{bmatrix} u_1 \\ \vdots \\ u_{n\ell} \end{bmatrix}$ the coefficient embedding of $\mathbf{u}$. We define the norm of $\mathbf{u} \in \mathcal{R}_q^\ell$ as $\|\mathbf{u}\| \coloneqq \|\overline{\mathbf{u}}\|_2$.

Definition 25 (Anticirculant matrix)**.** For a polynomial $u \in \mathcal{R}$, the anticirculant matrix of u is defined as

$$\mathcal{A}(u) = \begin{bmatrix} u_1 & u_2 & \dots & u_n \\ -u_n & u_1 & \dots & u_{n-1} \\ \vdots & & \ddots & \\ -u_2 & -u_3 & \dots & u_1 \end{bmatrix}$$

Lattice. A lattice in m-dimension Euclidean space $\mathbb{R}^m$ is a discrete set

$$\Lambda(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$$

of all integral combinations of n linearly independent vectors $\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}^m$.

We are specifically interested in NTRU lattices, that underly Falcon. They are defined over $\mathbb{R}^{2n}$ by polynomials $f, g \in \mathcal{R}$, and $h = g \cdot f^{-1}$ as the coefficient embedings of the module

$$\Lambda_h = \left\{ (u, v) \in \mathcal{R}^2, \text{ s.t. } u \cdot h + v = 0 \bmod q \right\}$$

Equivalently, Λ_h is generated by the columns of a matrix $\mathbf{B}_h = \begin{bmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathcal{A}(h) & q\mathbf{I}_n \end{bmatrix}$. A trapdoor for this lattice is a short basis $\mathbf{B} = \begin{bmatrix} \mathcal{A}(f) & \mathcal{A}(F) \\ \mathcal{A}(g) & \mathcal{A}(G) \end{bmatrix}$ where the polynomials $f, g, F, G \in \mathcal{R}$ are relatively short and $f \cdot G - g \cdot F = 0 \bmod q$.

Falcon relies on a trapdoor sampler, that generates a short matrix $\mathbf{B}$.

Definition 26 (NTRU trapdoor generation)**.** An NTRU trapdoor generation algorithm $\mathsf{TpdGen}()$ samples a public key $h \in \mathcal{R}_q \backslash \{0\}$, and a trapdoor $(f, g, F, G) \in \mathcal{R}^4$ such that $\mathbf{B}_h$ and $\mathbf{B}$ generate the same matrix and $\|\mathbf{B}_{f,h}\|_{\mathsf{GS}} \leq \alpha \cdot \sqrt{q}$.

We assume the ring $\mathcal{R}$, a target quality $\alpha \geq 1$, and a modulus q are provided as public parameters.

Gaussian sampling. For a positive real σ, let $\rho_\sigma(\mathbf{z}) = \exp(-\frac{-\|\mathbf{z}\|_2}{2\sigma^2})$. The discrete Gaussian distribution of standard deviation σ over a lattice Λ, centered in $\mathbf{c}$, is defined by its probability distribution function $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}(\mathbf{z}) = \frac{\rho_\sigma(\mathbf{z}-\mathbf{c})}{\sum_{\mathbf{z}' \in \Lambda} \rho_\sigma(\mathbf{z}'-\mathbf{c})}$ for $\mathbf{z} \in \Lambda$. We may omit $\mathbf{c}$ when it is 0.

Falcon's security hinges on the sampling of Gaussian vectors over the public lattice generated by $\mathbf{B}_h$, by leveraging the trapdoor $\mathbf{B}$.

Falcon's Gaussian sampler. Falcon relies on the FFT Gaussian sampler [DP15; Pre15], and adapts it to cope with real-world constraints, by tail-cutting internal variables, using floating-point arithmetic, and introducing a polynomial approximation. Looking forward, while these approximations introduce some bias, we will prove the deniability of our scheme by bounding the bias of tail cuts with statical arguments, and operation approximations with relative errors.

I.1.2. Formal Security Analysis

The proof of deniability of our scheme requires analyzing in depth the preimage sampler of Falcon, and notably the impact of tail cuts, floating points, and polynomial approximations.

We wish to prove that FalconRS is deniable for one signature ($Q = 1$), and rely on Lemma 11 to prove deniability for Q signatures. We will show for this that tail cuts create a small discrepancy between the support of the concrete preimage sampler and the arbitrary elements sampled in Ln. 7 that translates into a small term δ. The remaining biases introduced by floating point and approximations can then be analyzed with relative errors on distributions, and translate into the term μ of our deniability notion.

Before moving to a formal statement, we introduce notations for the preimage sampler, corresponding to the successive introduction of optimizations. We note PreSmp the concrete preimage of Falcon, with tail cuts, floating-point, polynomial approximation; $\mathsf{PreSmp}^{\mathsf{tl}}$ where only tail cuts are introduced; and finally $\mathsf{PreSmp}^{\mathsf{ideal}}$ the idealized preimage sampler, with no practical optimization.

Theorem 1 (Deniability of FalconRS)**.** *FalconRS is* (μ, δ)-deniable *for* $Q = 1$*, assuming that* H *is a random oracle, where* $\mu = \exp(\Delta_f)^{2C} \cdot \exp(\Delta_{\mathsf{FFO}})^{2C} \cdot \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{2C}$ *and* $\delta = 2^{-\lambda} + (1+\mu) \cdot \delta_{\mathsf{tl}}$*, where*

- *Noting* $p_{\mathsf{rej}} \coloneqq \max_{\mathbf{B}} \Pr[\|(u_j)_j, v + c\| > \beta; u_1 \leftarrow \chi_u,$ $(u_0, v) \leftarrow \mathsf{PreSmp}(\mathbf{B}, \sigma, -c), c \xleftarrow{\$} \mathcal{R}_q]$ *and take* C *as the smallest integer verifying* $\left(C \cdot 2^{-\kappa} + p_{\mathsf{rej}}\right)^C < 2^{-\lambda}$.
- $\Delta_f = \max_{\mathbf{B},\sigma,c} \Delta_{\mathsf{ML}}(\mathsf{PreSmp}(\mathbf{B}, \sigma, c), \mathsf{PreSmp}^{\mathsf{tl}}(\mathbf{B}, \sigma, c))$
- $\Delta_{\mathsf{FFO}} = \max_{\mathbf{B},\sigma,c} \Delta_{\mathsf{ML}}(\mathsf{PreSmp}^{\mathsf{ideal}}(\mathbf{B}, \sigma, \mathsf{c}), \mathcal{D}_{\Lambda(\mathbf{B}),\sigma,\mathsf{c}})$
- $\delta_{\mathsf{tl}} = s + o(s^2)$ *where*
  - $s = 2C \cdot \left(2e^{-\frac{\eta'^2}{2\sigma^2}} + 2n \cdot 2e^{-\frac{\eta^2}{2\sigma_{\max}^2}} \cdot \frac{1+\varepsilon/(2n)}{1-\varepsilon/(2n)}\right)$
  - η, η' *are the bounds used for the tail cuts performed in FalconRS respectfully for* χ_u *and within* PreSmp
- $\varepsilon \in [0, 1)$ *is such that* $\sigma \geq \alpha\sqrt{q} \cdot \eta_\varepsilon(\mathbb{Z}^{2n})$.

Proof. We proceed with a series of hybrids starting from the game $\mathsf{Game}^{\mathsf{Deny}}_{\mathsf{RS},0,\mathcal{A}}(1^\lambda, Q = 1)$.

Hybrid_1. This is the original deniability game from Alg. 7 $\mathsf{Game}^{\mathsf{Deny}}_{\mathsf{RS},0,\mathcal{A}}(1^\lambda, Q)$, where the first signer signs the message.

Hybrid_2. In this hybrid, we constrain the number of restarts during signing to be at most C. Let us note $p_{\mathsf{rej}} \coloneqq \max_{\mathbf{B}} \Pr[\|(u_j)_j, v + c\| > \beta; z_1 \leftarrow \chi_u, (u_0, v) \leftarrow \mathsf{PreSmp}(\mathbf{B}, \sigma, -c), c \xleftarrow{\$} \mathcal{R}_q]$ and take C as the smallest integer verifying $\left(C \cdot 2^{-\kappa} + p_{\mathsf{rej}}\right)^C < 2^{-\lambda}$.

We can bound the probability of restart conditionned on previous restarts. After i restarts, the probability of starting again is bounded by:

- Sampling again a salt that was previously sampled: this happens with probability less than $i \cdot 2^\kappa$.
- When the salt is fresh, then c is sampled independently of the previous rejects and we can bound the probability of reject by p_{rej}.

The probability of performing more than C restarts can thus be bounded by $\prod_{i=0}^{C-1}(i \cdot 2^\kappa + p_{\mathsf{rej}}) < (C \cdot 2^\kappa + p_{\mathsf{rej}})^C$.
By definition of C, the probability of performing more than C restarts is thus bounded by $2^{-\lambda}$ and we deduce:

$$\left|\mathsf{Adv}^{\mathsf{Hybrid}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Hybrid}_2}_{\mathcal{A}}\right| \leq 2^{-\lambda}$$

Hybrid_3. In this hybrid, we remove the use of floating-points and polynomial approximation in the sampler, i.e. we replace the use of PreSmp with $\mathsf{PreSmp}^{\mathsf{tl}}$.

Previous works [How+20; Pre17] noted that using polynomial approximations and floating-point simply introduced a small relative error compared to the original sampler.

For each of the C uses of the sampler, we note we have $\Delta_f = \max_{\mathbf{B},\sigma,c} \Delta_{\mathsf{ML}}(\mathsf{PreSmp}(\mathbf{B}, \sigma, c), \mathsf{PreSmp}^{\mathsf{tl}}(\mathbf{B}, \sigma, c))$, and we obtain

$$\mathsf{Adv}^{\mathsf{Hybrid}_2}_{\mathcal{A}} \leq \exp(\Delta_f)^C \cdot \mathsf{Adv}^{\mathsf{Hybrid}_3}_{\mathcal{A}}$$

Hybrid_4. In this hybrid, we remove the tail cuts performed in the signing procedure, and replace the tailcut distributions by ideal ones. In particular, we replace $\mathsf{PreSmp}^{\mathsf{tl}}$ with the idealized $\mathsf{PreSmp}^{\mathsf{ideal}}$.

Tail cuts are performed in two places: (i) in the sampling of the u_i at Ln. 7, and (ii) within the preimage sampler $\mathsf{PreSmp}^{\mathsf{tl}}$.

The tail cuts used in Falcon's preimage sampler are all within a sub-primitive called $\mathsf{SamplerZ}$, that samples a Gaussian over $\mathbb{Z}$, for which we recall the ideal functionality in Alg. 12. In order to sample a preimage from $\mathcal{D}_{\Lambda(\mathbf{B}),\sigma,c}$ for a target c, the FFO sampler calls twice $\mathsf{SamplerZ}$ for each leaf of a secret tree. $\mathsf{SamplerZ}$ takes as input a standard deviation σ and a center μ and should return an element sampled from $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$. For concrete implementation, the half-Gaussian χ is implemented using a CDT table that is tailcut. Let us note $\mathsf{SamplerZ}^{\mathsf{tc}}$ the sampler where χ is tailcut to an interval $[0, \eta]$. Equivalently, it implies that $\mathsf{SamplerZ}^{\mathsf{tc}}$ samples elements from the distribution $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ tailcut to $\lfloor\mu\rfloor + [-\eta, \eta + 1]$.

Algorithm 12 Falcon Gaussian sampler for $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$

1: **function** $\mathsf{SamplerZ}^{\mathsf{ideal}}(\mu, \sigma')$ ▷ *Assume that* $\sigma' \in [\sigma_{\min}, \sigma_{\max}]$
2: $r \leftarrow \mu - \lfloor\mu\rfloor$
3: $\mathsf{ccs} \leftarrow \sigma_{\min}/\sigma'$
4: **while** true **do**
5: $z_0 \xleftarrow{\$} \chi$ ▷ *Half-Gaussian distribution of parameter* $\sigma_{\max}$
6: $b \xleftarrow{\$} \{0, 1\}$
7: $z \coloneqq b + (2b - 1) \cdot z_0$
8: $x \coloneqq \frac{(z-r)^2}{2\sigma'^2} - \frac{z_0^2}{2\sigma_{\max}^2}$
9: **if** $\mathcal{B}_{\mathsf{ccs}\cdot\exp(-x)} = 1$ **then**
10: **return** $z + \lfloor\mu\rfloor$

We rely on a lemma allowing to bound the statistical distance between $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ and its tailcut version.

Lemma 17 (Adapted from Lemma 4.3 and 4.4 of [Lyu12]). *For any vector* $\mathbf{v} \in \mathbb{R}^m$, $\boldsymbol{\mu} \in \mathbb{R}^m$ *and* $r > 0, \sigma \geq \eta_\varepsilon(\mathbb{Z}^m)$, *we have*

$$\Pr[|\langle \mathbf{z} - \boldsymbol{\mu}, \mathbf{v}\rangle| \geq r; \mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma,\boldsymbol{\mu}}] \leq 2e^{-\frac{r^2}{2\|\mathbf{v}\|^2\sigma^2}} \cdot \frac{1+\varepsilon}{1-\varepsilon}$$

As a special case for $m = 1, \mathbf{v} = 1$, *we get:*

$$\Pr[|z - \mu| \geq r; \mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma,\mu}] \leq 2e^{-\frac{r^2}{2\sigma^2}} \cdot \frac{1+\varepsilon}{1-\varepsilon}$$

If $\mu = 0$, *These formulas hold for* $\varepsilon = 0$.

Proof. Looking at the proof of Lemma 4.3 in [Lyu12], we can see that it straightforwardly adapts to the case of non-centered Gaussians up to an additional factor $\frac{\rho_\sigma(\mathbb{Z}^m)}{\rho_\sigma(\mathbb{Z}^m-\boldsymbol{\mu})}$.

Additionally, $\frac{\rho_\sigma(\mathbb{Z}^m)}{\rho_\sigma(\mathbb{Z}^m-\boldsymbol{\mu})} \leq \frac{1+\varepsilon}{1-\varepsilon}$ by [MR04, Lemma 4.4]. □

We can bound the probability of the tail cuts occuring in our scheme.

- First, the tailcut of χ_u has a probability bounded by $2e^{-\frac{\eta'^2}{2\sigma^2}}$, and note that this distribution is sampled at most C times.
- The tail cuts in SamplerZ have a probability bounded by $2e^{-\frac{\eta^2}{2\sigma_{\max}^2}} \cdot \frac{1+\varepsilon}{1-\varepsilon}$, noticing that $\sigma_{\max} \geq \sigma_{\min} \geq \sigma/(\alpha\sqrt{q}) \geq \eta_\varepsilon(\mathbb{Z}^{2n}) \geq 2n \cdot \eta_\varepsilon(\mathbb{Z})$, and these distributions are sampled at most $C \cdot 2n$ times.

We deduce a bound on $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}$:

$$\begin{aligned}
&\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3} \\
&\leq \frac{1}{1 - C \cdot \left(2e^{-\frac{\eta'^2}{2\sigma^2}} + 2n \cdot 2e^{-\frac{\eta^2}{2\sigma_{\max}^2}} \cdot \frac{1+\varepsilon/(2n)}{1-\varepsilon/(2n)}\right)} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4} \\
&\leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4} + s + o(s^2)
\end{aligned}$$

with $s = C \cdot \left(2e^{-\frac{\eta'^2}{2\sigma^2}} + 2n \cdot 2e^{-\frac{\eta^2}{2\sigma_{\max}^2}} \cdot \frac{1+\varepsilon/(2n)}{1-\varepsilon/(2n)}\right) < 1$.

Hybrid_5. In this hybrid, we replace the preimage output of $\mathsf{PreSmp}^{\mathsf{ideal}}(\mathbf{B}_0, \sigma, -c')$ with $\mathcal{D}_{\Lambda(\mathbf{B}_0),\sigma,-c'}$.

We note that the FFO sampler used within Falcon does not perfectly sample from $\mathcal{D}_{\Lambda(\mathbf{B}_0),\sigma,-c'}$ for a target c'. Indeed, the use of discrete Gaussians internally slightly bias the sampler, due to their slighly varying mass depending on their center. We recall Lemma 18 bounding this difference for the Klein sampler, and argue by analogy that the FFO sampler distribution is close to $\mathcal{D}_{\Lambda(\mathbf{B}_0),\sigma,-c'}$.

Lemma 18 (Relative error of Klein sampler [Pre15; Pre17])**.** *Let n a positive integer, and $\varepsilon \in (0, 1/4)$. Then, the relative error of the Klein sampler $\mathsf{KleinSmp}(\mathbf{B}, \sigma, c)$ for any basis $\mathbf{B} \in \mathbb{Z}^{m\times n}$, standard deviation $\sigma \geq \eta_\varepsilon(\mathbb{Z}^n) \cdot \|\mathbf{B}\|_{\mathsf{GS}}$, and arbitrary syndrome $c \in \mathbb{Z}^m$ is bounded by*

$$\Delta_{\mathsf{ML}}(\mathsf{KleinSmp}(\mathbf{B}, \sigma, c), \mathcal{D}_{\Lambda(\mathbf{B}),\sigma,c}) \leq n \cdot \log\left(\frac{1+\varepsilon/n}{1-\varepsilon/n}\right) \approx 2\varepsilon.$$

We obtain:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4} \leq \exp(\Delta_{\mathsf{FFO}})^C \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_5},$$

where we have that

$$\Delta_{\mathsf{FFO}} = \max_{\mathbf{B},\sigma,c} \Delta_{\mathsf{ML}}(\mathsf{PreSmp}^{\mathsf{ideal}}(\mathbf{B}, \sigma, \mathsf{c}), \mathcal{D}_{\Lambda(\mathbf{B}),\sigma,\mathsf{c}}).$$

Hybrid_6. In this hybrid, we exchange the roles of the users u_0 and u_1, i.e. instead of running the preimage sampler for user u_0, we sample a Gaussian from $\mathcal{D}_{\mathbb{Z}^n,\sigma}$ for user u_1 and run $\mathcal{D}_{\Lambda(\mathbf{B}_1),\sigma,-(c-h_0z_0)}$.

Concretely, we need to show that the signature distribution is close in these two cases, that is that the two distributions below are at a close distance:

- $(z_0, z_1, v + (c - h_0z_0))$ where $z_0 \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$, $(z_1, v) \leftarrow \mathcal{D}_{\Lambda(\mathbf{B}_1),\sigma,-(c-h_0z_0)}$
- $(z_0, z_1, v + (c - h_1z_1))$ where $z_1 \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$, $(z_0, v) \leftarrow \mathcal{D}_{\Lambda(\mathbf{B}_0),\sigma,-(c-h_1z_1)}$

We rely on Theorem 4.1 from [Gen+20].

Lemma 19 (Adapted from Theorem 4.1 of [Gen+20])**.** *For any $\varepsilon \in [0, 1)$, a full-rank lattice Λ, and $\sigma \geq \eta_\varepsilon(\Lambda)$, and matrix $\mathbf{T}$, any c in $\mathbb{R}^n$, the max-log distance between the distributions*

- (x_0, x_1) *where* $x_1 \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$ *and* $x_1 \leftarrow \mathbf{T} \cdot x_0 + \mathcal{D}_{\Lambda,\sigma,c-\mathbf{T}\cdot x_0}$
- $(x_0, x_1) \leftarrow \mathcal{D}_{\Lambda',\sigma,c}$ *where* $\Lambda' := \{(x_0, \mathbf{T} \cdot x_0 + x_1); x_0 \in \mathbb{Z}^n, x_1 \in \Lambda\}$

is bounded by $\log \frac{1+\varepsilon}{1-\varepsilon}$.

We conclude by applying twice Lemma 19 successively with the lattice of user 0 and then of user 1, and $\mathbf{T}$ successively being equal to $\mathcal{A}(-h_1)$ and $\mathcal{A}(-h_0)$. We conclude by noticing that the lattice Λ' in Lemma 19 is equal in both cases:

$$\begin{aligned}
&\{(z_0, z_1, v - h_0 z_0) \text{ s.t. } h_1 z_1 + v = 0 \bmod q\} \\
&\quad = \{(z_0, z_1, v) \text{ s.t. } \sum_i h_i z_i + v = 0 \bmod q\} \\
&\quad = \{(z_0, z_1, v - h_1 z_1) \text{ s.t. } h_0 z_0 + v = 0 \bmod q\}
\end{aligned}$$

The max-log distance between the two signature distributions is then $2 \log \frac{1+\varepsilon}{1-\varepsilon}$.
Finally, by the properties of the max-log distance:

$$\mathsf{Adv}^{\mathsf{Hybrid}_5}_{\mathcal{A}} \leq \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{2C} \cdot \mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}}$$

Hybrid_7. From this point, hybrids consist in re-establishing the concrete signing for user u_1. This hybrid replaces the distribution $\mathcal{D}_{\Lambda(\mathbf{B}_1),\sigma,-c'}$ by $\mathsf{PreSmp}(\mathbf{B}_1, \sigma, -c')$.
As for Hybrid_5, we obtain a bound

$$\mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}} \leq \exp(\Delta_{\mathsf{FFO}})^C \cdot \mathsf{Adv}^{\mathsf{Hybrid}_7}_{\mathcal{A}}$$

Hybrid_8. This hybrid re-introduces the tail cuts.
Similarly to the analysis in Hybrid_4, we bound the probability of the tails that are cut. Then, by expressing the tailcut distribution as following the ideal distribution conditioned on a smaller support, we obtain

$$\mathsf{Adv}^{\mathsf{Hybrid}_7}_{\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{Hybrid}_8}_{\mathcal{A}} + C\left(2e^{-\frac{\eta'^2}{2\sigma^2}} + 2n \cdot 2e^{-\frac{\eta^2}{2\sigma_{\min}^2}} \cdot \frac{1+\varepsilon/(2n)}{1-\varepsilon/(2n)}\right).$$

Hybrid_9. This hybrid reintroduces floating-point and polynomial approximations.
As in Hybrid_3, we get

$$\mathsf{Adv}^{\mathsf{Hybrid}_8}_{\mathcal{A}} \leq \exp(\Delta_f)^C \cdot \mathsf{Adv}^{\mathsf{Hybrid}_9}_{\mathcal{A}}$$

Hybrid_{10}. We finally remove the constraint on the number of rejection restarts. The advantage of the adversary can only be increased by this change:

$$\mathsf{Adv}^{\mathsf{Hybrid}_9}_{\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{Hybrid}_{10}}_{\mathcal{A}}$$

Hybrid_{10} is exactly the $\mathsf{Game}^{\mathsf{Deny}}_{\mathsf{RS},1,\mathcal{A}}(1^\lambda, Q = 1)$ and we obtain the theorem statement by collecting all the bounds.

□

I.1.3. Concrete Parameters

We rely on the same ring, key generation, and preimage sampler as Falcon-512:

- the degree of the polynomial ring is $n = 512$, and modulo is $q = 12289$.
- the key quality is $\alpha = 1.17$.
- the signing standard deviation is $\sigma = 165.736617183$ (corresponding to $\varepsilon \approx 2^{-35.5}$ [GJK24a]).
- the salt are sampled from $\{0,1\}^\kappa$ with $\kappa = 320$.
- The preimage sampler uses as implementation constants $\sigma_{\min} = 1.277\,833\,697$, $\sigma_{\max} = 1.8205$. For the internal randomness tailcut, $\eta = 18$.

For a ring of two users, we select the check bound as $\beta = 1.1 \cdot \sqrt{3n}\sigma$. We also pick the new tailcut parameter $\eta' \coloneqq \lceil \sqrt{70 \cdot \log(2)} \cdot \sqrt{2} \cdot \sigma \rceil \approx 1633$. We then evaluate the security of our scheme with the lattice-estimator.
As for the deniability of our scheme, we evaluate the different elements of Theorem 1,

- Following the analysis of [Pre+22, Section 2.5.2], the use of floating points and polynomial approximations introduces a small error: $\exp(\Delta_f) \leq 1 + 2^{-31}$.
- By analogy with the Klein sampler, the bias of the FFO sampler is of the order $\exp(\Delta_{\mathsf{FFO}}) \leq 1 + 2\varepsilon \approx 1 + 2^{-34.5}$.
- The probability of rejection can be evaluated with Theorem 1 to $p_{\mathsf{rej}} \approx 2^{-21.4}$. Then, it is sufficient to take $C = 6$ to ensure $(C \cdot 2^{-\kappa} + p_{\mathsf{rej}})^C \leq 2^{-128}$ in Theorem 1.

Lemma 20. *With the same notations as Theorem 1,* $p_{\mathsf{rej}} \leq \exp(\Delta_f) \cdot \exp(\Delta_{\mathsf{FFO}}) \cdot (\frac{1+\varepsilon}{1-\varepsilon}) \cdot p_\beta + \delta$ *where we define* $p_\beta = \Pr\left[\|(z_j)_j, v\| > \beta; (z_0, z_1, v) \leftarrow \mathcal{D}_{\mathbb{Z}^{3n},\sigma}\right]$.

Remark that p_β *can be bounded by* $k^{3n} \cdot \exp(\frac{3n}{2}(1-k^2))$ *where* $k = \beta/(\sqrt{3n}\sigma)$ *using [Lyu12, Lemma 4.4].*

Proof. As in the proof of Theorem 1, we can bound the effect of floating point, polynomial approximation, tail cuts, and the bias of the FFO sampler: $p_{\mathsf{rej}} \leq \Delta_f \cdot \Delta_{\mathsf{FFO}} \cdot (\frac{1+\varepsilon}{1-\varepsilon}) \cdot p_{\mathsf{rej}}^{\mathsf{ideal}} + \delta$ where we define $p_{\mathsf{rej}}^{\mathsf{ideal}} = \max_{\mathbf{B}} \Pr[\|(z_j)_j, v + c\| > \beta; z_1 \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}, (z_0, v + c) \leftarrow \mathcal{D}_{\Lambda(\mathbf{B}),\sigma,-c}, c \xleftarrow{\$} \mathcal{R}_q]$.

Now, we rely on a lemma showing that the syndrome $c = h_1 \cdot z_1 + v$ is close to uniform when (z_1, v) are Gaussians of parameter σ, to remove the conditioning on c in $p_{\mathsf{rej}}^{\mathsf{ideal}}$.

Lemma 21 (Adapted from Lemma 5.2 [GPV08], Cor. 2 [GJK24a])**.** *For* $\varepsilon \in (0, \frac{1}{2})$, $\sigma \geq \eta_\varepsilon(\Lambda_h)$, *consider the distributions* $\mathcal{P} = \mathcal{U}(\mathcal{R}_q)$ *and* Q *the distribution of* $u \cdot h + v$, *where* $u, v \leftarrow \mathcal{D}_{\mathcal{R},\sigma}$. *Then,*

$$\Delta_{\mathsf{ML}}(\mathcal{P}, Q) \leq \log \frac{1+\varepsilon}{1-\varepsilon}$$

Applying the above lemma, we deduce that $p_{\mathsf{rej}}^{\mathsf{ideal}} \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot p_\beta$, which concludes the proof. □

Following the above analysis of the terms of Theorem 1, we deduce that FalconRS is deniable with concrete terms:

- $\mu = 1 + 2^{-27}$
- $\delta = 2^{-57}$

I.2. MAYO-based Ring Signature

We now formally introduce our ring signature based on MAYO. We reuse the same parameter names as the original MAYO specification [Beu+24]. Viewing MAYO as a hash-then-sign signature scheme, we can apply generic transformations from [AOS02] to obtain an efficient ring signature.

Recall that MAYO is designed over quadratic maps. It first chooses a map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ with trapdoor tp as public key, from which is derived a larger map $\mathcal{P}^* : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$. To sign message M we compute a target $\mathbf{t} = H(\mathsf{M}, \mathsf{salt})$ (where salt is a random value), and then we sample a preimage u such that $\mathcal{P}^*(\mathbf{u}) = \mathbf{t}$, leveraging the trapdoor tp.

For the matter of abstraction and re-use within a ring signature, we proceed as for FalconRS and denote $\mathsf{PreSmp}(\mathsf{tp}, \mathbf{t}) \to \mathbf{u}$ the random procedure of MAYO sampling a pre-image $\mathbf{u}$ given a target $\mathbf{t}$, that is sampling $\mathbf{u}$ such that $\mathcal{P}^*(\mathbf{u}) = \mathbf{t}$.

We finally describe in Alg. 13 the ring signature MayoRS obtained from MAYO using Abe, Ohkubo, and Suzuki transform.

I.2.1. Security Analysis

Theorem 2 (Unforgeability of MayoRS)**.** *MayoRS is unforgeable assuming the unforgeability of MAYO.*

Formally, let $\mathcal{A}$ *be an adversary against the unforgeability of MayoRS making at most* Q_s *calls to the signing oracle, at most* Q_H *random oracle queries, and with at most* N *generated keypairs. Then, there exists an adversary* $\mathcal{B}$ *against the unforgeability of MAYO, running in time* $\mathcal{T}(\mathcal{B}) \approx \mathcal{T}(\mathcal{A})$, *such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RS-UF}} \leq N \cdot Q_H \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{UF}} + 2^{-\kappa} \cdot Q_s \cdot (Q_s + Q_H) + 2^{-2\lambda} \cdot Q_H^2$$

Proof. We proceed with a series of hybrids. Our proof roughly follows ideas presented in [LAZ19a], but improves the security loss by a factor Q_H by guessing only one random oracle query, and then programming all subsequent ones, instead of guessing a pair. Additionally, we cover the introduction of a seed used for compressing the signatures when preimages are comparatively larger than seeds.

Hybrid_1**.** This is the original unforgeability game for a ring signature.

Hybrid_2**.** In this hybrid, we create a MAYO signing oracle OSign_i for each keypair $(\mathsf{sk}_i, \mathsf{pk}_i)_{i\in[N]}$, and we reexpress the signing oracle of MayoRS as a function of the individual OSign_i. We assume the existence of separate random oracles H_i used within OSign_i, and only accessible by the challenger.

First, we fix an arbitrary message μ^*.

Then, whenever the ring signature signing oracle is called on input $(i, \mathsf{RL}, \mathsf{M})$, we sample random elements $\mathbf{u}_j \xleftarrow{\$} \mathbb{F}_q^n$ for $j \neq i$, $(\mathsf{salt}', \mathbf{u}_i) \leftarrow \mathsf{OSign}_i(\mu^*)$, as well as $\mathbf{c}_0 \xleftarrow{\$} \mathbb{F}_q^m$, and we sequentially compute $\mathbf{c}_{j+1 \bmod N} = H(j+1 \bmod N, \mathsf{RL}, \mathsf{M}, \mathbf{c}_j + \mathcal{P}_j(\mathbf{u}_j))$ for $j \in [N-1]$.

Finally, we compute the seed $\mathsf{seed} = H(0, \mathsf{RL}, \mathsf{M}, \mathbf{c}_{N-1} + \mathcal{P}_{N-1}(\mathbf{u}_{N-1}))$, sample a salt $\mathsf{salt} \leftarrow \{0,1\}^\kappa$, and program the random oracle $H(0, \mathsf{seed}, \mathsf{salt}) = \mathbf{c}_0$.

Algorithm 13 MAYO-based ring signature scheme

function MayoRS.KeyGen(1^λ)
  $\mathcal{P}, \mathsf{tp} \xleftarrow{\$} \mathsf{TpdGen}()$
  return $(\mathsf{rvk} \coloneqq \mathcal{P}, \mathsf{rsk} \coloneqq \mathsf{tp})$

function MayoRS.Sign($\mathsf{rsk}_i, \mathsf{M}, \mathsf{RL} \coloneqq \{\mathsf{rvk}_j\}_j$)
  $\mathsf{tp}_i \coloneqq \mathsf{rsk}_i$; $\{\mathcal{P}_j\}_j \coloneqq \{\mathsf{rvk}_j\}_j$
  $\mathsf{salt} \xleftarrow{\$} \{0,1\}^\kappa$; $\mathbf{t} \xleftarrow{\$} \mathbb{F}_q^m$
  $\mathsf{ctx} \coloneqq (i+1 \bmod N, \mathsf{RL}, \mathsf{M}, \mathbf{t})$
  if $[\![i+1 \bmod N = 0]\!]$ **then**
    $\mathsf{seed} \coloneqq H(0, \mathsf{RL}, \mathsf{M}, \mathbf{t}) \in \{0,1\}^{2\lambda}$; $\mathsf{ctx} \coloneqq (0, \mathsf{seed}, \mathsf{salt})$
  $\mathbf{c}_{i+1 \bmod N} =: H(\mathsf{ctx}) \in \mathbb{F}_q^m$
  for $j = i+1, ..., N-1, 0, ..., i-1$ **do**
    $\mathbf{u}_j \xleftarrow{\$} \mathbb{F}_q^{kn}$
    $\mathsf{ctx} \coloneqq (j+1 \bmod N, \mathsf{RL}, \mathsf{M}, \mathbf{c}_j + \mathcal{P}_j^*(\mathbf{u}_j))$
    if $[\![j+1 \bmod N = 0]\!]$ **then**
      $\mathsf{seed} \coloneqq H(\mathsf{ctx}) \in \{0,1\}^{2\lambda}$; $\mathsf{ctx} \coloneqq (0, \mathsf{seed}, \mathsf{salt})$
    $\mathbf{c}_{j+1 \bmod N} \coloneqq H(\mathsf{ctx}) \in \mathbb{F}_q^m$
  $\mathbf{u}_i \coloneqq \mathsf{PreSmp}(\mathsf{tp}_i, \mathbf{t} - \mathbf{c}_i)$
  return $\mathsf{sig} \coloneqq (\mathsf{salt}, \mathsf{seed}, \{\mathbf{u}_j\}_j)$

function MayoRS.Verify($\mathsf{RL} \coloneqq \{\mathsf{rvk}_j\}_j, \mathsf{M}, \mathsf{sig}$)
  $(\mathsf{salt}, \mathsf{seed}, \{\mathbf{u}_i\}_i) \coloneqq \mathsf{sig}$; $\{\mathcal{P}_i\}_i \coloneqq \{\mathsf{rvk}_i\}_i$
  $\mathbf{c}_0 \coloneqq H(0, \mathsf{seed}, \mathsf{salt}) \in \mathbb{F}_q^m$
  for $j = 0, ..., N-2$ **do**
    $\mathbf{c}_{j+1 \bmod N} =: H(j+1 \bmod N, \mathsf{RL}, \mathsf{M}, \mathbf{c}_j + \mathcal{P}_j^*(\mathbf{u}_j)) \in \mathbb{F}_q^m$
  return $[\![\mathsf{seed} = H(0, \mathsf{RL}, \mathsf{M}, \mathbf{c}_{N-1} + \mathcal{P}_j^*(\mathbf{u}_{N-1}))]\!]$

The view of the adversary is identical unless the random oracle H has been queried on $(0, \mathsf{seed}, \mathsf{salt})$ before it is programmed, which happens with probability bounded by $2^{-\kappa} \cdot Q_s Q_H$, or in case the same salt' is provided twice by OSign_i, which happens with probability bounded by $2^{-\kappa} Q_s^2$.

Hence,

$$\left| \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2} \right| \leq 2^{-\kappa} \cdot Q_s \cdot (Q_s + Q_H)$$

Hybrid_3. In this hybrid, we assert that if $H(0, \mathsf{seed}, \mathsf{salt})$ is queried for some seed, then seed cannot be returned later by the random oracle.

The advantage loss is related to the preimage resistance of the hash-function H. Given that seed is sampled in $\{0,1\}^{2\lambda}$, the probability to find a preimage for a seed is less than $Q_H \cdot 2^{-2\lambda}$. As the adversary can query up to Q_H seeds, we obtain:

$$\left| \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3} \right| \leq 2^{-2\lambda} \cdot Q_H^2$$

Hybrid_4. In this hybrid, we guess the index of the hash in the adversary's forgery that was queried first. We start by sampling $u \xleftarrow{\$} [Q_H]$. We abort the game in case the index u does not correspond to the hash used in the forgery that was queried first to the random oracle.

This hybrid inccurs a loss Q_H:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3} \leq Q_H \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}$$

We assert that the first hash used in a forgery will be of the form $H(i^*, \mathsf{RL}^*, \mathsf{M}^*, \mathbf{e}^*)$. Indeed, the previous hybrid ensured that the only hash taking a input with a different form (i.e. taking as input seed) cannot be the first one called by the adversary.

Looking forward, observe that a forged ring signature is composed of vectors $\mathbf{u}_i^* \in \mathbb{F}_q^n$ such that for any $i \in [N]$, we have $\mathcal{P}_i(\mathbf{u}_i^*) = \mathbf{e}_i - \mathbf{c}_i$, where $\mathbf{c}_i$ is a hash of the form $H(i-1 \bmod N, \cdot)$, and $\mathbf{e}_i$ is used as input of a hash of the form $H(i, \mathsf{RL}, \mathsf{M}, \mathbf{e}_i)$. By guessing the value of $\mathbf{e}_i$ before the corresponding $\mathbf{c}_i$ is defined, we will then be able to program the random oracle to choose $\mathbf{c}_i$ so as to target any value for $\mathbf{e}_i - \mathbf{c}_i$.

Hybrid_4. In this hybrid, we additionally guess the value of $i^* \in [N]$.

This reduces the adversary's advantage by an additional factor N:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3} \leq N \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}$$

Hybrid_5**.** In this hybrid, after $\mathbf{e}^*$ is defined, we program all the calls to the random oracle so that the values $\mathbf{e}^* - \mathbf{c}_i$ is equal to a target of the underlying signature for user i^*. Concretely, whenever the random oracle is called on some input X and has to output a uniform $\mathbf{c} \in \mathbb{F}_q^m$, we take an arbitrary message μ, different from μ^* and any other message previously chosen. We also choose an arbitrary salt. We then program $H(X) \coloneqq \mathbf{e}^* - H_i(\mu, \mathsf{salt})$.

As H_i is a random oracle only accessible to the challenger, the view of the adversary is identically distributed in this game:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_5}$$

Conclusion. Finally, we observe that when $\mathcal{A}$ wins the game Hybrid_4, they output a signature verifying $\mathcal{P}_{i^*}(\mathbf{u}_{i^*}) = \mathbf{e}^* - \mathbf{c}_i = H_i(\mu, \mathsf{salt})$ for some μ, salt that were sampled in Hybrid_5, i.e. a forgery for the i^*-th keypair.

We can hence reduce the probability of the adversary winning this game to the unforgeability of MAYO by replacing the i^*-th keypair by the one given by the unforgeability challenger against MAYO, and by aborting in case the corruption oracle is called on index i^*.

Formally, there exists an adversary $\mathcal{B}$ against the unforgeability of MAYO, running in time $\mathcal{T}(\mathcal{A}) \approx \mathcal{T}(\mathcal{B})$, such that:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_5} \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{UF}}$$

□

Theorem 3 (Deniability of MayoRS)**.** *MayoRS is* $(1, 2N \cdot B)$*-deniable where* $B = \frac{q^{k-(n-o)}}{q-1} + \frac{q^{m-ko}}{q-1}$*, assuming one challenge query is allowed.*

Proof. In order to prove the deniability of MayoRS, we will show that we can replace the uniform sampling of non-signers, by the preimage sampling function of MAYO to make the signature procedure independent of the actual signer.

Looking in more detail at the preimage sampler of MAYO, we can see that given a target $\mathbf{t}$, it samples signatures of the form $(\mathbf{V} + \mathbf{X}\mathbf{O}^t, \mathbf{X})$, where $\mathbf{V} \in \mathbb{F}_q^{k \times (n-o)}$ is uniformly distributed, under the condition that an associated matrix $\mathbf{A} \in \mathbb{F}_q^{m \times ko}$ (derived from $\mathbf{V}$ and the keypair pk, sk) is full-rank, and $\mathbf{X}$ is uniformly sampled among the possible preimages of $\mathbf{t}$ of the form $(\mathbf{V} + \mathbf{X}\mathbf{O}^t, \mathbf{X})$.

It can be seen that when $\mathbf{t}$ is uniformly distributed and obtaining $\mathbf{V}, \mathbf{X}$ with the preimage sampler, then the joint distribution of $(\mathsf{pk}, \mathsf{sk}, \mathbf{V}, \mathbf{X}, \mathbf{t})$ is identical to first sampling $(\mathsf{pk}, \mathsf{sk}, \mathbf{V})$ such that the associated matrix $\mathbf{A}$ is full-rank, uniformly sampling $\mathbf{X}$, and taking $\mathbf{t}$ to be the evaluation of $\mathcal{P}$ on $(\mathbf{V} + \mathbf{X}\mathbf{O}^t, \mathbf{X})$.

Formally, we will introduce a series of hybrids to prove deniability, starting from the game where the signer b is signing.

Hybrid_2**.** In this hybrid, we sample a matrix $\mathbf{V}_j$ for each user uniformly at random. When the signing oracle is called for the ring (i, j) and signer i, we replace the random sampling of $\mathbf{u}_j$ by $(\mathbf{V}_j + \mathbf{X}\mathbf{O}_j^t, \mathbf{X})$, where $\mathbf{X}$ is uniformly sampled.

As we only allow one signing query, this change does not change the distribution of the view of the adversary, and there is no advantage loss as the preimage for user j is still uniformly sampled.

$$\Pr\left[\mathsf{Game}_{\mathsf{RS},0,\mathcal{A}}^{\mathsf{Deny}}(1^\lambda, Q) = 0\right] = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}$$

Hybrid_3**.** In this hybrid, we replace the matrices $\mathbf{V}_i$ and individual keypairs and instead sample them such that each keypair and $\mathbf{V}_i$ has an associated matrix $\mathbf{A}_i$ that is full-rank.

Let us note E the event that all the matrices $\mathbf{A}_j$ are full-rank. Then, we can reexpress the adversary's advantage in Hybrid_2:

$$\begin{aligned}\Pr\left[\mathsf{Hybrid}_2(\mathcal{A}) = 0\right] = &\underbrace{\Pr\left[\mathsf{Hybrid}_2(\mathcal{A}) = 0 \mid E\right]}_{=\Pr\left[\mathsf{Hybrid}_3(\mathcal{A})=0\right]} \cdot \Pr\left[E\right] \\ &+ \Pr\left[\mathsf{Hybrid}_3(\mathcal{A}) = 0 \mid \neg E\right] \cdot \Pr\left[\neg E\right]\end{aligned}$$

We deduce an upper bound on the advantage loss between the Hybrid_2 and Hybrid_3:

$$\left|\Pr\left[\mathsf{Hybrid}_2(\mathcal{A}) = 0\right] - \Pr\left[\mathsf{Hybrid}_3(\mathcal{A}) = 0\right]\right| \leq \Pr\left[\neg E\right] \leq N \cdot B$$

where $B = \frac{q^{k-(n-o)}}{q-1} + \frac{q^{m-ko}}{q-1}$ is the probability that a given $\mathbf{A}_j$ is full-rank, and the factor N appears by union-bound.

Hybrid_4**.** In this hybrid, when user i is signing for the ring (i, j), we replace the use of the matrix $\mathbf{V}_j$ with MAYO's preimage sampler, i.e., we first sample a random target $\mathbf{t}_j$ for $\mathcal{P}(\mathbf{u}_j)$ and then sample $\mathbf{u}_j$ using MAYO's preimage sampler.

We show that this change does not affect the distribution of the view of the adversary.

First, the distribution of $(\mathsf{pk}_j, \mathsf{sk}_j, \mathbf{V}_j)$ is identical to the distribution of $(\mathsf{pk}_j, \mathsf{sk}_j, \mathbf{V})$ where $\mathbf{V}$ is the matrix used by MAYO's preimage sampler. Indeed, we can see that $\mathbf{V}$ is also sampled such that $\mathbf{A}_i$ is full-rank, conditionning on the value of $(\mathsf{pk}_j, \mathsf{sk}_j)$.

Then, as observed in the introduction of this proof, it is equivalent to first sample $\mathbf{X}$ uniformly, then compute the corresponding $\mathbf{t}$, or to instead sample $\mathbf{t}$ uniformly first, and then compute $\mathbf{X}_j$. We thus conclude that $(\mathbf{X}, \mathbf{t})$ follow the same distribution in Hybrid_3 and Hybrid_4.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}$$

Hybrid_5**.** In this hybrid, when the signing oracle is called on user i for the ring (i, j), we instead sign with user j.

It is easy to see at this point that when $\mathbf{t}, \mathbf{t}_j$ are uniformly distributed, then $(\mathbf{t} - H(H(\mathbf{t}) + \mathbf{t}_j), \mathbf{t}_j)$, is identically distributed to $(\mathbf{t} - H(\mathbf{t}_j), \mathbf{t}_j - H(\mathbf{t}))$, which is itself identically distributed to $(\mathbf{t}, \mathbf{t}_j - H(H(\mathbf{t}_j) + \mathbf{t}))$, i.e. we can inverse the sampling of the contributions of i and j.

Thus, this hybrid has no advantage loss:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_5}$$

Conclusion. Observe that Hybrid_5 is the exact counterpart of Hybrid_4 by changing the user who signed. We can then reapply the hybrids Hybrid_3, Hybrid_2, Hybrid_1 in reverse order to obtain that:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_5} - \Pr\left[\mathsf{Game}_{\mathsf{RS},1,\mathcal{A}}^{\mathsf{Deny}}(1^\lambda, 1) = 0\right]\right| \leq N \cdot B$$

We finally conclude the proof by collecting all the bounds.

□

I.2.2. Alternative Parameter Sets for MayoRS

As observed in Tab. 2, MAYO parameter sets proposed for standardization offer a rather low deniability, and we provide in this section alternative parameter sets that achieve higher deniability guarantees at the cost of larger public key and signature sizes.

Interestingly, we can rapidly increase the deniability of MAYO by increasing its parameter k. After adapting the parameters to balance the security loss and still achieve NIST security level I, we propose three new parameter sets MAYO*, MAYO** and MAYO***. We provide concrete values for the parameters (n, m, o, k, q) in Tab. 9.

Table 9: Alternative parameter sets for MAYO with higher deniability guarantees, aiming for NIST security level I. Recall that the parameter B translates into $(\mu = 1, \delta = 2 \cdot N \cdot B)$-deniability. Sizes are given in Tab. 2.

Scheme	$(\mathbf{n}, \mathbf{m}, \mathbf{o}, \mathbf{k}, \mathbf{q})$	**Parameter B**
MAYO*	$(69, 69, 9, 9, 16)$	2^{-52}
MAYO**	$(70, 70, 9, 10, 16)$	2^{-83}
MAYO***	$(78, 78, 9, 12, 16)$	2^{-124}