A preliminary version of this paper appears in the proceedings of the 44th Annual International Cryptology Conference (CRYPTO 2024). This is the full version.

Ring Signatures for Deniable AKEM: Gandalf's Fellowship

Phillip Gajland[1,2], Jonas Janneck[2], and Eike Kiltz[2]

[1] Max Planck Institute for Security and Privacy
[2] Ruhr-Universität Bochum

July 9, 2024

Abstract Ring signatures, a cryptographic primitive introduced by Rivest, Shamir and Tauman (ASIACRYPT 2001), offer signer anonymity within dynamically formed user groups. Recent advancements have focused on lattice-based constructions to improve efficiency, particularly for large signing rings. However, current state-of-the-art solutions suffer from significant overhead, especially for smaller rings.

In this work, we present a novel NTRU-based ring signature scheme, Gandalf, tailored towards small rings. Our post-quantum scheme achieves a 50% reduction in signature sizes compared to the linear ring signature scheme Raptor (ACNS 2019). For rings of size two, our signatures are approximately a quarter the size of DualRing (CRYPTO 2021), another linear scheme, and remain more compact for rings up to size seven. Compared to the sublinear scheme Smile (CRYPTO 2021), our signatures are more compact for rings of up to 26. In particular, for rings of size two, our ring signatures are only 1236 bytes.

Additionally, we explore the use of ring signatures to obtain deniability in Authenticated Key Encapsulation Mechanisms (AKEMs), the primitive behind the recent HPKE standard used in MLS and TLS. We take a fine-grained approach at formalising sender deniability within AKEM and seek to define the strongest possible notions. Our contributions extend to a black-box construction of a deniable AKEM from a KEM and a ring signature scheme for rings of size two. Our approach attains the highest level of confidentiality and authenticity, while simultaneously preserving the strongest forms of deniability in two orthogonal settings. Finally, we present parameter sets for our schemes, and show that our deniable AKEM, when instantiated with our ring signature scheme, yields ciphertexts of 2004 bytes.

Contents

1 Introduction

Ring Signatures. The seminal work of Rivest, Shamir and Tauman [RST01] introduced ring signatures as an extension of group signatures [Cv91], allowing users to sign messages on behalf of dynamically formed user groups. This cryptographic primitive facilitates public verification while preserving the signer's anonymity within the group, referred to as the signing ring ρ. Ring signatures have witnessed widespread adoption across various domains, including blockchains, digital currencies such as Monero and Bytecoin, as well as electronic voting systems. A plethora of constructions based on number-theoretic assumptions exist [BSS02, Nao02, AOS02, ZK02, BGLS03, DKNS04], with recent focus shifting towards post-quantum ring signatures. Here, lattice-based constructions [BK10, ABB+13, LLNW16, BLO18, ESS+19, BKP20, LNS21] represent a significant body of research. Recent advancements have leveraged proof systems [ESS+19, BKP20, LNS21], leading to better efficiency for large signing rings. The current state of the art is Smile by Lyubashevsky, Nguyen, and Seiler [LNS21], achieving asymptotic signature sizes $\mathcal{O}(\log(|\rho|))$. While asymptotically sublinear, these proof systems involve significant overhead and concrete instantiations range from 16 KB for $|\rho| \leq 32$ users to 22 KB for up to $|\rho| = 2^{25}$ users. In applications involving small rings (Monero uses rings of size 11)[3], linearly scaling schemes are often preferable. For ring of size two, the Raptor ring signature by Lu, Au, and Zhang [LAZ19] emerges as the best option, yielding signatures of approximately 2.5 KB. When the ring size is between 4 and 439, the DualRing scheme [YEL+21] is the most compact.

Deniable AKEM. The authenticated key encapsulation mechanism (AKEM) primitive studied in [ABH+21, AJKL23], can be thought of as the KEM analogue of signcryption [Zhe97, DZ10], and plays a crucial role in authenticating the sender to the receiver in two modes of the HPKE standard [BBLW22]. Despite HPKE's integration into protocols like Messaging Layer Security (MLS) [BBR+23] and the Encrypted Client Hello privacy extension for Transport Layer Security (TLS) 1.3 [ROSW23], it's deniability aspects remain unexplored in the literature. This is somewhat surprising considering HPKE constructs an AKEM using a non-interactive key exchange (NIKE) for authentication, suggesting some form of deniability. However, the specifics remain unclear.

Ring Signatures for Deniability. There exists a folklore belief regarding the potential applicability of ring signatures in constructing deniable authentication. For instance, in two-party scenarios where a sender seeks to deniably authenticate itself to a receiver, linear size ring signatures would be advantageous. However, the precise notions of anonymity within ring signatures and the resulting level of deniability remain subjects of subtlety. For example, a recent work from PKC'22 [BFG+22] suggested employing anonymous ring signatures to establish deniable key exchange within the context of the Signal Handshake (X3DH) [MP16b].

1.1 Contributions and Technical Overview

This section outlines our main contributions: a novel ring signature scheme, formalisation of deniability for Authenticated Key Encapsulation Mechanisms (AKEMs), and a generic construction of deniable AKEMs inspired by our ring signature scheme.

Ring Signatures. Our primary contribution is Gandalf, a lattice-based ring signature scheme, specifically designed for small rings. Compared to existing schemes, Gandalf offers a remarkable improvement of over 50%. It provides compact signatures, as small as 1236 bytes for two-member rings. The signature size scales linearly with the ring size $|\rho| = k$, occupying $606 \cdot k + 24$ bytes. This renders Gandalf the most compact option for rings up to size 7.

At a technical level, Gandalf, is based on the NTRU preimage sampleable trapdoor function $f_{\boldsymbol{h}}$ [GPV08] over the NTRU ring $\mathcal{R}$ [HPS98, DLP14, PFH+22]. Concretely, $f_{\boldsymbol{h}}$ inputs two ring elements of small norm and is defined as $f_{\boldsymbol{h}}(\boldsymbol{u}, \boldsymbol{v}) := \boldsymbol{h} * \boldsymbol{u} + \boldsymbol{v}$. A valid ring signature on message m for the ring $\rho = \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k\}$

[3] https://www.getmonero.org/resources/moneropedia/ring-size.html

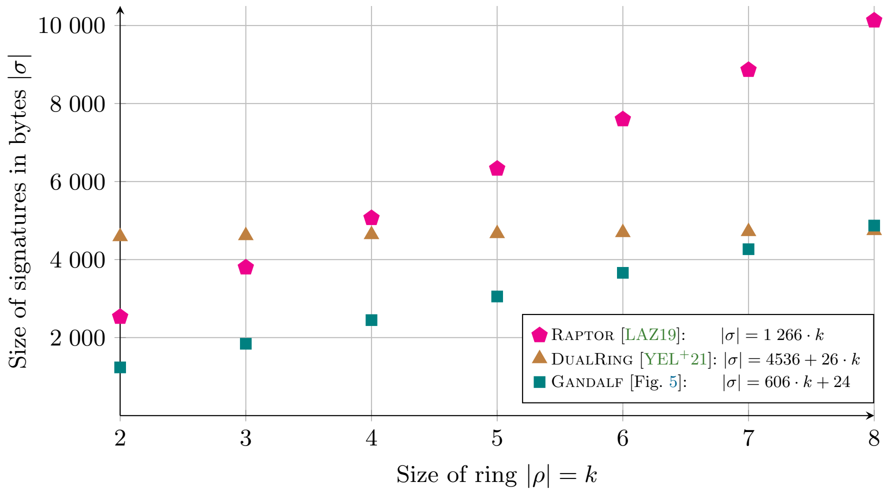

Figure 1. Signature sizes against ring sizes for state of the art lattice-based schemes.

simply consists of a vector $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k) \in \mathcal{R}^k$ such that

$$\left\| \left(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v} \coloneqq \mathsf{H}(m, \rho) - \sum_{i=1}^{k} \boldsymbol{h}_i * \boldsymbol{u}_i \right) \right\|_2 \leq \beta. \tag{1}$$

Note that the ring signature essentially consists of k "unseeded ANTRAG signatures" [ENS+23] and the ring element $\boldsymbol{v}$ is implicitly reconstructed in the verification equation. Our construction can also be seen as a *ring trapdoor function* [BK10] leveraging concrete algebraic properties of NTRU rings for compactness. The ring signature can be computed by the holder of the j-th secret key (i.e., the trapdoor for $f_{\boldsymbol{h}_j}$) by first sampling small $\boldsymbol{u}_i$ for $i \neq j$ and finally computing $(\boldsymbol{u}_j, \boldsymbol{v}) \leftarrow f_{\boldsymbol{h}_j}^{-1}(\mathsf{H}(m, \rho) - \sum_{i \neq j}^{k} \boldsymbol{u}_i * \boldsymbol{h}_i)$ using preimage sampling. Unfortunately, the norm on the verification equation Equation (1) increases with the maximal size of the ring κ, and hence security decreases with larger ring sizes. However, in these cases other schemes would be preferable.

For GANDALF we prove one per message unforgeability [FKP17] under chosen ring attacks [BKM06, BKM09], which is sufficient for our main application. Similar to FALCON or ANTRAG, we achieve full unforgeability by adding a small random seed to the hash function. Furthermore, we consider a stronger notion of anonymity than typically examined in the literature, namely that of an adaptive multi-challenge anonymity under full key exposure, as this will become useful for our applications. For all our proofs we give concrete security bounds.

DENIABILITY FOR AKEMS. Our subsequent contribution is to formally investigate deniability in the context of AKEM. Deniability aims to prevent a third party — modelled as the adversary — from conclusively attributing a, potentially incriminating, message to a particular sender. We consider eight distinct settings to characterise deniability in a fine grained approach, focusing on two main settings; honest and dishonest receivers. For scenarios involving honest receivers, we can assume that they do not simulate any values. Thus, an adversary is given only the sender's secret key sk_s. Note that a notion where the adversary is additionally given the receiver's secret key sk_r is known to be impossible. Concurrently, we investigate a different setting where the receiver is considered to be dishonest. In this setting the strongest notion one could hope to achieve gives the adversary both sk_s and sk_r and further assumes the existence of a simulator, which, given access to sk_r, is able to produce a ciphertext and key that are indistinguishable from those generated by the AKEM encapsulation process Enc. Hence, the ciphertext could be constructed by the receiver itself by executing the simulator and the sender can plausibly deny their involvement. As for all our proofs and notions we consider the multi-user setting where the adversary can query oracles adaptively.

Deniable AKEM Construction. Our third contribution is a black-box construction of deniable AKEM from key encapsulation mechanisms and ring signatures for rings of size two in the standard model. Notably, AKEM has two existing notions of authenticity. Insider authenticity models a setting with an insider adversary (having access to receivers' secret keys) where outsider authenticity only allows outsider adversaries. While insider authenticity implies outsider authenticity, the latter is the strongest notion compatible with any form of deniability. The reason being that a simulator that is given the secret key from the deniability notion can be used to break the insider authentication notion of the AKEM. Our approach achieves the highest level of CCA security, known as insider CCA security, and the most robust form of authentication, outsider authentication, while preserving deniability in both the honest and dishonest receiver setting.

Evaluation. Our final contribution is to select appropriate parameters for Gandalf and instantiating our AKEM construction from Gandalf and the best NTRU KEM from [DHK+23]. Leveraging the latest developments in Gaussian sampling we instantiate our schemes with help of [ENS+23], thus avoiding issues related to floating point arithmetic, ensuring robustness for potential future implementations. For a comprehensive comparison of our ring signature schemes against other alternatives, refer to Figure 1. Our resulting AKEM has ciphertexts of 2004 bytes and public keys of 1664 bytes.

1.2 Related Work

Ring Signatures. Ring signatures [RST01] have been extensively studied in the cryptographic literature. Bender et al. [BKM09] provide a thorough examination of ring signatures, covering various unforgeability and anonymity notions, along with several constructions. For large rings, the most efficient constructions rely on proof systems [ESS+19, BKP20, LNS21]. These could be made efficient using lattice-based succinct non-interactive arguments of knowledge (SNARKs) [ACL+22], although this would likely involve significant overhead for provers. Other schemes are more suitable for small rings. One closely related work to ours is the Raptor scheme proposed by Lu et al. [LAZ19], which also presents a linkable ring signature scheme. Their scheme, approximately 1.3 KB per user, relies on chameleon hash functions with slightly stronger properties which they call Chameleon Hash+. Moreover, their construction is also instantiated over NTRU lattices, where signatures consist of $k = |\rho|$ many $(\boldsymbol{u}, \boldsymbol{v})$ pairs of polynomials along with a 32 byte salt. Another approach was taken in [YEL+21] where they introduce a new ring signature construction they call DualRing which can be built from identification schemes. They provide an instantiation based on M-LWE and M-SIS. While the signature size grows linearly, increasing by only 24 bytes per user, each signature includes a large constant of 4536 bytes. Another approach is that of MPC-in-the-Head, where the state of the art yields signatures of at least 4.41 KB [FR23].

Authenticated KEMs. Related to AKEM is another primitive, called split-KEM, which was introduced by Brendel, Fischlin, Günther, Janson, and Stebila [BFG+20]. Split-KEMs feature two distinct key generation algorithms – one for sender keys and one for receiver keys. This comes with separate secret and public key spaces for encapsulation and decapsulation, resulting in each party having two distinct key pairs for sending and receiving. In contrast, AKEM can be regarded as a more general primitive as it provides a more unified approach, enabling constructions where a single key serves both encapsulation and decapsulation functions. This stands in contrast to split-KEMs, where using the same key for both would necessitate duplication, as exemplified by the AKEM construction employed in HPKE [BBLW22], formally analysed in [ABH+21]. While the authors of [BFG+20] present post-quantum secure instantiations of split-KEMs, none of them meets their strongest security notion, full IND-CCA security (with multiple encapsulations and decapsulations). A recent work, due to appear at USENIX [CHDN+24], constructs a lattice-based split-KEM that achieves a somewhat weaker notion of confidentiality, IND-1BatchCCA security, as well as unforgeability against one known-ciphertext attacks.

Note that the straightforward combination of KEM and signature does not fulfil the strongest security guarantees for the insider setting [DZ10, Chapter 2.3]. The AKEM from [AJKL23] achieves the strongest confidentiality notion using a black-box construction. We build upon this construction, resulting in a scheme with the same security guarantees but relying on weaker assumptions, extending seamlessly to the split

KEM context and providing robust confidentiality and authenticity. As such, AKEM could potentially be applied in new approaches to X3DH [MP16b, BFG+20], one of the main components behind Signal [MP16a], WhatsApp [Wha20, BCG23] and Facebook Messenger. Moreover, our approach is compatible with any CCA post-quantum KEM, such as NTRU [CDH+20] or Kyber [SAB+22].

Deniable Authentication. Deniable authentication, a concept introduced by Dwork, Naor and Sahai [DNS98, DNS04], combines sender authentication with the ability to deny involvement to a third party. This concept has been extensively studied in the realm of authenticated key exchange (AKE) [DG05, DGK06, UG15, UG18, BFG+22], where deniability extends from exchanging keys to the denial of entire communications under a shared key.

Typically, AKE involves multiple rounds of interaction to satisfy both authentication and deniability requirements. KEM-TLS [SSW20], which relies on a KEM, falls into the same line of work by requiring interactions. Another line of research explores deniable ring authentication [Nao02], where users within a designated ring can deny sending a message while maintaining authentication within the ring. As for AKE, there is no limit on the number of rounds of interactions. In contrast, Susilo and Mu [SM04] investigated non-interactive ring authentication which uses a single message for sender-to-receiver authentication. However, their construction is based on ring signatures and chameleon hash functions resulting in rather weak deniability properties. The work of [FM15] gives a comprehensive overview of notions of deniable message authentication; part of it is similar to our settings of deniability for AKEMs (see Section 4.2).

In [UG15, UG18] various black-box constructions are presented using ring signatures, of which Spawn+ is the most similar to our deniable AKEM (a one-shot primitive). Our black-box construction of deniable AKEM requires the following primitives: a ring signature scheme, a KEM, and a symmetric encryption scheme (which does not incur any overhead in ciphertext size); whereas the construction of Spawn+ requires: a ring signature scheme and a Dual-Receiver Encryption with Associated Data, a primitive that is objectively more costly than a KEM. Dual-Receiver Encryption with Associated Data, requires two encryptions (one to each participant), a non-interactive zero knowledge proof of knowledge (NIZKPK) proving that ciphertexts contain the same plaintext. Furthermore, instantiating Spawn+ with post-quantum NIZKs would incur an additional cost. On the other hand, the ZDH/XZDH exchange implicitly involves a Diffie-Hellman key exchange in order to derive the MAC key. Translating this to the post-quantum setting would require significantly larger public keys, if using a post-quantum NIKE.

Another work [HKKP22] constructed *Signal-conforming AKE* from ring signatures and NIZKs. However, the primitive is not one-move, as it requires ephemeral KEM keys. Additionally, they consider a weaker anonymity notion for their ring signatures (no secret keys are exposed) which translates to weaker deniability for the AKE.

Another folklore method to achieve deniability is non-interactive key exchange (NIKE) due to its symmetric nature, enabling implicit authentication. This differs from most other approaches that employ explicit methods, such as sending a signature from the sender to the receiver, which can then be explicitly verified to confirm the sender's authenticity. A main drawback of the NIKE approach is that it only provides sender deniability guarantees in a scenario where the receiver is potentially dishonest but no guarantee for the sender in an honest receiver setting (for detailed information, we refer to Section 4.2). The same setting is considered in the work of [CHDN+24] constructing a lattice-based deniable split-KEM focusing solely on the dishonest receiver setting and achieving a slightly weaker notion of deniability, where the simulator is only given the receiver's secret key, and the adversary is likewise only given the receiver's secret key (not the sender's secret key).

2 Preliminaries

In this section, we present important preliminaries. Further standard preliminaries are defined in Appendix A.

2.1 Notation

Sets and Algorithms. We write $s \xleftarrow{\$} \mathcal{S}$ to denote the uniform sampling of s from the finite set $\mathcal{S}$. For an integer n, we define $[n] := \{1, \ldots, n\}$. The notation $[\![b]\!]$, where b is a boolean statement, evaluates to 1 if the statement is true and 0 otherwise. We use uppercase letters $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ to denote algorithms. Unless otherwise stated, algorithms are probabilistic, and we write $(y_1, \ldots) \xleftarrow{\$} \mathcal{A}(x_1, \ldots)$ to denote that $\mathcal{A}$ returns $(y_1, \ldots)$ when run on input $(x_1, \ldots)$. We write $\mathcal{A}^{\mathcal{B}}$ to denote that $\mathcal{A}$ has oracle access to $\mathcal{B}$ during its execution. For a randomised algorithm $\mathcal{A}$, we use the notation $y \in \mathcal{A}(x)$ to denote that y is a possible output of $\mathcal{A}$ on input x. The support of a discrete random variable X is defined as $\sup(X) := \{x \in \mathbb{R} \mid \Pr[X = x] > 0\}$. For two polynomials $\boldsymbol{f}, \boldsymbol{g}$, we denote the polynomial multiplication of $\boldsymbol{f}$ and $\boldsymbol{g}$ by $\boldsymbol{f} * \boldsymbol{g}$.

Security Games. We use standard code-based security games [BR04]. A *game* G is a probability experiment in which an adversary $\mathcal{A}$ interacts with an implicit challenger that answers oracle queries issued by $\mathcal{A}$. The game G has one *main procedure* and an arbitrary amount of additional *oracle procedures* which describe how these oracle queries are answered. We denote the (binary) output b of game G between a challenger and an adversary $\mathcal{A}$ as $\mathsf{G}(\mathcal{A}) \Rightarrow b$. $\mathcal{A}$ is said to *win* G if $\mathsf{G}(\mathcal{A}) \Rightarrow 1$, or shortly $\mathsf{G} \Rightarrow 1$. Unless otherwise stated, the randomness in the probability term $\Pr[\mathsf{G}(\mathcal{A}) \Rightarrow 1]$ is over all the random coins in game G. If a game is aborted the output is either 0 or a random bit b in case of an indistinguishability game, i.e. a game for which the advantage of an adversary is defined as the absolute difference of winning the game to $\frac{1}{2}$. To provide a cleaner description and avoid repetitions, we sometimes refer to procedures of different games. To call the oracle procedure $\texttt{Oracle}$ of game G on input x, we shortly write $\mathsf{G}.\texttt{Oracle}(x)$.

2.2 Lattice Preliminaries

NTRU Lattices. We use the GPV [GPV08] framework instantiated over NTRU lattices as done in [DLP14].

Definition 1 (Lattice). *For a finite basis $\boldsymbol{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$, let the* lattice $\boldsymbol{\Lambda}(\boldsymbol{B})$*, or simply $\boldsymbol{\Lambda}$, be the set of vectors*

$$\boldsymbol{\Lambda}(\boldsymbol{B}) := \left\{ \sum_{i=1}^{n} c_i \boldsymbol{b}_i \mid c_i \in \mathbb{Z} \right\}.$$

Definition 2 (NTRU Lattice). Let $N = 2^k$ for $k \in \mathbb{Z}$, q prime, $\boldsymbol{f}, \boldsymbol{g} \in \mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, and $\boldsymbol{h} = \boldsymbol{g} * \boldsymbol{f}^{-1} \mod q$. The NTRU lattice parameterised by $\boldsymbol{h}$ and q is a lattice of volume q^N in $\mathbb{R}^{2N}$ in the coefficient embedding of the following module

$$\boldsymbol{\Lambda}_{\boldsymbol{h},q} := \{(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{R}_q^2 : \boldsymbol{u} * \boldsymbol{h} + \boldsymbol{v} = \boldsymbol{0} \mod q\}.$$

Equivalently, for $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, an NTRU lattice is a full-rank submodule lattice of $\mathcal{R}^2$ generated by the columns of a matrix of the form

$$\boldsymbol{B_h} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{h} & \boldsymbol{q} \end{bmatrix}$$

for prime q and some $\boldsymbol{h} \in \mathcal{R}$ coprime to q. A trapdoor for this lattice is a relatively short basis

$$\boldsymbol{B_{f,g}} = \begin{bmatrix} \boldsymbol{f} & \boldsymbol{F} \\ \boldsymbol{g} & \boldsymbol{G} \end{bmatrix}$$

where the basis vectors $(\boldsymbol{f}, \boldsymbol{g}) \in \mathcal{R}^2$ and $(\boldsymbol{F}, \boldsymbol{G}) \in \mathcal{R}^2$ are not much larger than $\sqrt{\det \boldsymbol{B_h}} = \sqrt{q}$ and $\boldsymbol{f} * \boldsymbol{G} - \boldsymbol{g} * \boldsymbol{F} = q$.

Norms. For a polynomial $\boldsymbol{f} \in \mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$, let $f \in \mathbb{Z}_q^N$ denote the coefficient embedding of $\boldsymbol{f}$, and $f_i \in \mathbb{Z}_q$ the i^{th} coefficient. For an element $f_i \in \mathbb{Z}_q$, we write $|f_i|$ to denote $|f_i \bmod q|$. Let the ℓ_2-norm for

$\boldsymbol{f} = f_0 + f_1 X + \ldots + f_{N-1} X^{N-1} \in \mathcal{R}_q$ be defined as $\|\boldsymbol{f}\|_2 \coloneqq \sqrt{\sum_{i=0}^{N-1} |f_i|^2}$. For polynomials $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k \in \mathcal{R}_q$ we use the notation

$$\|(\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k)\|_2 \coloneqq \sqrt{\sum_{i=0}^{N-1} \left(|f_{1_i}|^2 + \ldots + |f_{k_i}|^2\right)}.$$

Gaussians and Preimage Sampling. We recall some concepts and tools for Gaussian sampling.

Definition 3 (Discrete Gaussian Distribution over $\boldsymbol{\Lambda}$). For any standard deviation $s > 0$, the n-dimensional *Gaussian function* $\rho_{s,c} \colon \mathbb{R}^n \to (0, 1]$ on $\mathbb{R}^n$ centred at $c \in \mathbb{R}^n$ with standard deviation s is defined by

$$\rho_{s,c}(x) \coloneqq \exp\left(-\frac{\|x - c\|_2^2}{2s^2}\right).$$

For any $c \in \mathbb{R}^n$, $s \in \mathbb{R}^+$, and lattice $\boldsymbol{\Lambda}$, the *discrete Gaussian distribution over* $\boldsymbol{\Lambda}$ is defined as

$$\forall\, x \in \boldsymbol{\Lambda}, \quad \mathcal{D}_{\boldsymbol{\Lambda},s,c} \coloneqq \frac{\rho_{s,c}(x)}{\sum_{z \in \boldsymbol{\Lambda}} \rho_{s,c}(z)}.$$

We omit the subscript c when the Gaussian is centred at 0 and subscript $\boldsymbol{\Lambda}$ when the Gaussian is over $\mathbb{Z}^n$.

For bounding the probability that a random variable deviates a long way from the mean, we will use the following tail bounds from [Ban93, Lyu12].

Lemma 1. Let $n > 1$ and $s > 0$.

1. For any $\tau > 0$, $\Pr_{z \leftarrow \mathcal{D}_{\mathbb{Z},s}}[|z| > \tau s] \leq 2e^{\frac{-\tau^2}{2}}$.
2. For any $\tau > 1$, $\Pr_{z \leftarrow \mathcal{D}_s}[\|z\|_2 > \tau s \sqrt{n}] < \tau^n e^{\frac{n}{2}(1-\tau^2)}$.

Definition 4 (Gram-Schmidt Norm [GPV08, DLP14]). For a finite basis $\boldsymbol{B} = (\boldsymbol{b}_i)_{i \in I}$, let $\tilde{\boldsymbol{B}} = (\tilde{\boldsymbol{b}}_i)_{i \in I}$ be its Gram-Schmidt orthogonalization. Then the Gram-Schmidt norm of $\boldsymbol{B}$ is the value $\|\boldsymbol{B}\|_{GS} \coloneqq \max_{i \in I} \left\|\tilde{\boldsymbol{b}}_i\right\|$.

Lemma 2 (NTRU Trapdoor Generation [HPS98, Pre15]). Let $\mathcal{R} \coloneqq \mathbb{Z}[X]/(X^N + 1)$. There exists a PPT algorithm, $\mathsf{TpdGen}(q, \alpha)$, that given a modulus q, and a target quality α, returns a public key $\boldsymbol{h} \in \mathcal{R}$, and the trapdoor $(\boldsymbol{f}, \boldsymbol{g}) \in \mathcal{R} \times \mathcal{R}$, such that $\boldsymbol{B}_{\boldsymbol{h}}$ and $\boldsymbol{B}_{\boldsymbol{f},\boldsymbol{g}}$ form a basis of the same lattice. Furthermore, the Gram-Schmidt norm $\|\boldsymbol{B}_{\boldsymbol{f},\boldsymbol{g}}\|_{GS} \leq \alpha\sqrt{q}$.

Let $\boldsymbol{\Lambda}$ be an n-dimensional lattice and $\epsilon > 0$, the (scaled) smoothing parameter $\eta_\epsilon(\boldsymbol{\Lambda})$ is the smallest $s > 0$ such that $\rho_{1/s}(\boldsymbol{\Lambda}^* \setminus 0) \leq \epsilon$, where $\boldsymbol{\Lambda}^*$ denotes the dual lattice (the exact definition of the dual is not required for this work). We will use the following upper bound on the smoothing parameter.

Lemma 3 (Special case of [MR07, Lem. 4.4]). For any $\epsilon \in (0, 1)$ it holds

$$\eta_\epsilon\left(\mathbb{Z}^{2N}\right) \leq \frac{1}{\pi} \cdot \sqrt{\frac{\ln(4N(1 + 1/\epsilon))}{2}}.$$

Definition 5 (Rényi Divergence [BLL+15, Pre17]). Let $\mathcal{P}, \mathcal{Q}$ be two distributions such that $\sup(\mathcal{P}) \subseteq \sup(\mathcal{Q})$. For $a \in (1, \infty)$, we define the Rényi divergence of order a as

$$R_a(\mathcal{P} \,||\, \mathcal{Q}) \coloneqq \left(\sum_{x \in \sup(\mathcal{P})} \frac{\mathcal{P}(x)^a}{\mathcal{Q}(x)^{a-1}}\right)^{\frac{1}{a-1}}.$$

Definition 6 (Kullback-Leibler Divergence). Let $\mathcal{P}$ and $\mathcal{Q}$ be two discrete probability distributions over the same countable set $\mathcal{X}$. The *KL divergence* of $\mathcal{P}$ from $\mathcal{Q}$ is defined as

$$\delta_{KL}(\mathcal{P} \mid\mid \mathcal{Q}) \coloneqq \sum_{x \in \mathcal{X}} \ln\left(\frac{\mathcal{P}(x)}{\mathcal{Q}(x)}\right) \cdot \mathcal{P}(x).$$

Definition 7 (Relative Error [MW17]). Let $\mathcal{P}$ and $\mathcal{Q}$ be two discrete probability distributions over the same countable set $\mathcal{X}$. The relative error of $\mathcal{P}$ and $\mathcal{Q}$ is defined as

$$\delta_{RE}(\mathcal{P}, \mathcal{Q}) \coloneqq \max_{x \in \sup(\mathcal{P})} \frac{|\mathcal{P}(x) - \mathcal{Q}(x)|}{\mathcal{P}(x)}.$$

The relative error can be used to bound the KL-divergence. We make use of the following result from [MW17].

Lemma 4 ([MW17, Lem. 2.1]). For any two distributions $\mathcal{P}$, and $\mathcal{Q}$ with the same support and $\delta_{RE}(\mathcal{P}, \mathcal{Q}) < 1$,

$$\delta_{KL}(\mathcal{P} \mid\mid \mathcal{Q}) \leq \frac{\delta_{RE}(\mathcal{P}, \mathcal{Q})^2}{2(1 - \delta_{RE}(\mathcal{P}, \mathcal{Q}))^2}.$$

In particular, using the Taylor series expansion the function can be approximated to $\delta_{KL}(\mathcal{P}, \mathcal{Q}) \lesssim \frac{1}{2}\delta_{RE}(\mathcal{P}, \mathcal{Q})^2$ at $\delta_{RE} = 0$.

Similarly, the relative error can be used to bound the Rényi.

Lemma 5 (Relative error [Pre17, Lem. 3]). Let $\mathcal{P}, \mathcal{Q}$ be two distributions such that $\sup(\mathcal{P}) = \sup(\mathcal{Q})$ and $\delta_{RE} > 0$. Then for $a \in (1, +\infty]$,

$$R_a(\mathcal{P} \mid\mid \mathcal{Q}) \lesssim 1 + \frac{a\delta_{RE}^2}{2}.$$

Lemma 6 (Relative Error of Preimage Sampler [Pre17, Lem. 6]). Let N be a positive integer and $\epsilon \in (0, 1/4)$. Then there exists a preimage sampling algorithm $\mathsf{PreSmp}(\boldsymbol{B}, s, \boldsymbol{c})$, such that for any basis $\boldsymbol{B}$, standard deviation $s \geq \eta_\epsilon(\mathbb{Z}^{2N}) \cdot \|\boldsymbol{B}\|_{GS}$ and arbitrary syndrome $\boldsymbol{c}$, the *relative error* is bounded by

$$\delta_{RE}(\mathsf{PreSmp}(\boldsymbol{B}, s, \boldsymbol{c}), \mathcal{D}_{\boldsymbol{\Lambda}(\boldsymbol{B}), s, \boldsymbol{c}}) \leq \left(\frac{1 + \epsilon/N}{1 - \epsilon/N}\right)^N - 1 \approx 2\epsilon.$$

Combining Lemmas 4 to 6 yields the following useful corollary.

Corollary 1. Let N be a positive integer, $a > 1$, and $\epsilon \in (0, 1/4)$. Then there exists a preimage sampling algorithm $\mathsf{PreSmp}(\boldsymbol{B}, s, \boldsymbol{c})$, such that for any basis $\boldsymbol{B}$, standard deviation $s \geq \eta_\epsilon(\mathbb{Z}^{2N}) \cdot \|\boldsymbol{B}\|_{GS}$ and arbitrary syndrome $\boldsymbol{c}$, the *KL divergence* and *Renyi divergence* is bounded by

$$\delta_{KL} \coloneqq \delta_{KL}(\mathsf{PreSmp}(\boldsymbol{B}, s, \boldsymbol{c}) \mid\mid \mathcal{D}_{\boldsymbol{\Lambda}(\boldsymbol{B}), s, \boldsymbol{c}}) \lesssim 2\epsilon^2$$

and

$$R_a(\mathsf{PreSmp}(\boldsymbol{B}, s, \boldsymbol{c}) \mid\mid \mathcal{D}_{\boldsymbol{\Lambda}(\boldsymbol{B}), s, \boldsymbol{c}}) \lesssim 1 + 2a\epsilon^2.$$

We use the shorthand $\mathcal{R}_a(\mathsf{PreSmp} \mid\mid \mathcal{D})$ if the parameters are clear from the context.

Hardness Assumption. We define the $\mathcal{R}$-**LWE** problem over NTRU lattices, with respect to a Gaussian distribution with standard deviation s.

Definition 8. Let $\mathcal{R} \coloneqq \mathbb{Z}[X]/(X^N + 1)$. The *Ring Learning With Errors* problem relative to the NTRU trapdoor algorithm TpdGen with parameters $m, q, \alpha > 0$ and $s \geq 0$ is defined via the game $\mathcal{R}$-**LWE**, depicted in Figure 2. We define the advantage of $\mathcal{A}$ in $\mathcal{R}$-**LWE** as

$$\mathrm{Adv}^{\mathcal{R}\text{-}\mathbf{LWE}}_{m,q,\alpha,s,\mathcal{A}} \coloneqq \Pr[\mathcal{R}\text{-}\mathbf{LWE}_{m,q,\alpha,s}(\mathcal{A}) \Rightarrow 1].$$

Game $\mathcal{R}$-$\mathbf{LWE}_{m,q,\alpha,s}(\mathcal{A})$

$b \xleftarrow{\$} \{0,1\}$
$\boldsymbol{u} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^N,s}$
for $i \in [m]$
  $(\boldsymbol{h}_i, \cdot, \cdot) \leftarrow \mathsf{TpdGen}(q, \alpha)$
  $\boldsymbol{v} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^N,s}$
  if $b = 0$
    $\boldsymbol{z}_i \coloneqq \boldsymbol{u} * \boldsymbol{h}_i + \boldsymbol{v}$
  else
    $\boldsymbol{z}_i \xleftarrow{\$} \mathcal{R}_q$
$b' \xleftarrow{\$} \mathcal{A}((\boldsymbol{h}_1, \boldsymbol{z}_1), \ldots, (\boldsymbol{h}_m, \boldsymbol{z}_m))$
return $[\![b = b']\!]$

Game $\mathcal{R}$-$\mathbf{ISIS}_{m,q,\alpha,\beta}(\mathcal{A})$

for $i \in [m]$
  $(\boldsymbol{h}_i, \cdot, \cdot) \leftarrow \mathsf{TpdGen}(q, \alpha)$
$\boldsymbol{c} \xleftarrow{\$} \mathcal{R}_q$
$(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m, \boldsymbol{v}) \xleftarrow{\$} \mathcal{A}(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_m, \boldsymbol{c})$
return $\left[\!\!\left[\sum_{i\in[m]} \boldsymbol{h}_i * \boldsymbol{u}_i + \boldsymbol{v} = \boldsymbol{c} \wedge \|(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m, \boldsymbol{v})\|_2 \leq \beta \right]\!\!\right]$

Figure 2. Games defining $\mathcal{R}$-$\mathbf{LWE}_{m,q,\alpha,s}$ and $\mathcal{R}$-$\mathbf{ISIS}_{m,q,\alpha,\beta}$

We will use the following variant of the $\mathcal{R}$-**ISIS** problem over NTRU lattices.

Definition 9 ($\mathcal{R}$-ISIS). Let $\mathcal{R} \coloneqq \mathbb{Z}[X]/(X^N+1)$. The *Inhomogenious Ring Short Integer Solution* problem relative to the NTRU trapdoor algorithm TpdGen with parameters $m, q > 0$ and $\alpha, \beta > 0$ is defined via the game $\mathcal{R}$-**ISIS**, depicted in Figure 2. We define the advantage of $\mathcal{A}$ in $\mathcal{R}$-**ISIS** as

$$\mathrm{Adv}^{\mathcal{R}\text{-}\mathbf{ISIS}}_{m,q,\alpha,\beta,\mathcal{A}} \coloneqq \Pr[\mathcal{R}\text{-}\mathbf{ISIS}_{m,q,\alpha,\beta}(\mathcal{A}) \Rightarrow 1].$$

According to [LM06], $\mathcal{R}$-$\mathbf{ISIS}_{m,q,\alpha,\beta}$ is as hard as $\mathbf{SVP}_\gamma$ for $\gamma = \tilde{O}(N\beta)$. In particular, its hardness is independent of m.[4]

3 Ring Signatures

3.1 Definitions

We recall syntax and standard security notions of ring signatures [RST01].

Definition 10 (Ring Signature). Formally, a *ring signature* scheme RSig is given by three algorithms $(\mathsf{Gen}, \mathsf{Sgn}, \mathsf{Ver})$.

$par \xleftarrow{\$} \mathsf{Stp}(\kappa)$**:** Given an upper bound on the ring size ρ, the probabilistic setup algorithm Stp returns system parameters par, where par defines a message space $\mathcal{M}$. We assume that all algorithms are implicitly given access to the system parameters par.

$(sk, pk) \xleftarrow{\$} \mathsf{Gen}$**:** The probabilistic key generation algorithm returns a secret key sk and a corresponding public key pk.

$\sigma \xleftarrow{\$} \mathsf{Sgn}(sk, \rho, m)$**:** Given a secret key sk, a ring $\rho = \{pk_1, \ldots, pk_k\}$ such that the public key pk corresponding to sk satisfies $pk \in \rho$ and $k \leq \kappa$, and a message $m \in \mathcal{M}$, the probabilistic signing algorithm Sgn returns a signature σ from a signature space $\mathcal{S}$.

$b \leftarrow \mathsf{Ver}(\sigma, \rho, m)$**:** Given a signature σ, a ring ρ, and a message m, the deterministic verification algorithm Ver returns a bit b, such that $b = 1$ if and only if σ is a valid signature on m and $b = 0$ otherwise.

RSig is $\delta(\kappa)$-*correct* or has *correctness error* $\delta(\kappa)$ if for all $\kappa \in \mathbb{N}$, $par \xleftarrow{\$} \mathsf{Stp}(\kappa)$, and $\{(sk_i, pk_i)\}_{i\in[k]} \in \sup(\mathsf{Gen})$, and for any $i \in [k]$ with $k \leq \kappa$,

$$\Pr[\mathsf{Ver}(\mathsf{Sgn}(sk_i, \rho, m), \rho, m) \neq 1] \leq \delta(\kappa),$$

[4] Standard $\mathcal{R}$-**ISIS** is usually defined with respect to uniform ring elements $\boldsymbol{h}_i$. But under the NTRU assumption, $\boldsymbol{h}_i$ generated using TpdGen are computationally indistinguishable from uniform ones.

where $\rho \coloneqq \{pk_1, \ldots, pk_k\}$, and the probability is taken over the random choices of Stp, Gen and Sgn.

We assume (w.l.o.g.) that there is a mapping μ from the space of secret keys to the space of public keys such that for all $(sk, pk) \in \sup(\mathsf{Gen})$ it holds $\mu(sk) = pk$.

UNFORGEABILITY. Unforgeability for ring signatures states that, given a target set of public-keys $\{pk_1, \ldots, pk_n\}$, an adversary cannot forge a signature σ^* on a message m^* and a ring $\rho^* \subseteq \{pk_1, \ldots, pk_n\}$. The adversary is furthermore allowed to make adaptive signing queries on a message m_i and ring ρ_i, as long as the ring contains at least one of the supplied key from the set $\{pk_1, \ldots, pk_n\}$ (and hence the experiment knows the corresponding secret key). This is also referred to as "insider security" in [BKM09] since it models an adversary who is part of a ring for which an honest signature is created. This is the strongest unforgeability notion for ring signatures considered in [BKM09]. We will further consider the weaker notion of *one-per-message* unforgeability, where the adversary is only allowed to make a single signing query for each message/ring pair (m_i, ρ_i). The two notions *unforgeability under chosen ring attacks* and *one-per-message unforgeability under chosen ring attacks* are formalised through the games $(n, \kappa, Q_{\mathsf{Sgn}})$-$\mathbf{UF\text{-}CRA}_{\mathsf{RSig}}(\mathcal{A})$ and $(n, \kappa, Q_{\mathsf{Sgn}})$-$\mathbf{UF\text{-}CRA1}_{\mathsf{RSig}}(\mathcal{A})$ depicted in Figure 3, where n is the number of users, κ the maximal ring size, and Q_{Sgn} is an upper bound on the signing queries. We define the advantage functions of adversary $\mathcal{A}$ as

$$\mathrm{Adv}^{(n,\kappa,Q_{\mathsf{Sgn}})\text{-}\mathbf{UF\text{-}CRA}}_{\mathsf{RSig},\mathcal{A}} \coloneqq \Pr[(n, \kappa, Q_{\mathsf{Sgn}})\text{-}\mathbf{UF\text{-}CRA}_{\mathsf{RSig}}(\mathcal{A}) \Rightarrow 1],$$
$$\mathrm{Adv}^{(n,\kappa,Q_{\mathsf{Sgn}})\text{-}\mathbf{UF\text{-}CRA1}}_{\mathsf{RSig},\mathcal{A}} \coloneqq \Pr[(n, \kappa, Q_{\mathsf{Sgn}})\text{-}\mathbf{UF\text{-}CRA1}_{\mathsf{RSig}}(\mathcal{A}) \Rightarrow 1].$$

Games $(n, \kappa, Q_{\mathsf{Sgn}})$-$\mathbf{UF\text{-}CRA}_{\mathsf{RSig}}(\mathcal{A})$ and $(n, \kappa, Q_{\mathsf{Sgn}})$-$\mathbf{UF\text{-}CRA1}_{\mathsf{RSig}}(\mathcal{A})$	**Oracle** $\mathsf{Sgn}(i \in [n], \rho, m)$
$\mathcal{Q} \leftarrow \emptyset$	**if** $pk_i \notin \rho$
$par \xleftarrow{\$} \mathsf{Stp}(\kappa)$	**return** $\perp$
for $i \in [n]$	**if** $(\rho, m) \in \mathcal{Q}$ // **UF-CRA1**
$(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$	**return** $\perp$ // **UF-CRA1**
$(\sigma^\star, \rho^\star, m^\star) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sgn}}(par, pk_1, \ldots, pk_n)$	$\sigma \xleftarrow{\$} \mathsf{Sgn}(sk_i, \rho, m)$
return $[\![\rho^\star \subseteq \{pk_1, \ldots, pk_n\} \wedge \mathsf{Ver}(\sigma^\star, \rho^\star, m^\star) = 1 \wedge (\rho^\star, m^\star) \notin \mathcal{Q}]\!]$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\rho, m)\}$
	return σ

Figure 3. Games defining **UF-CRA** and **UF-CRA1** for a ring signature scheme RSig and adversary $\mathcal{A}$.

ANONYMITY. In our study of ring signatures, we focus on the strongest possible notion of anonymity, namely that of *anonymity under full key exposures* from [BKM09]. This means, that it is indistinguishable which participant of a ring signed a message even if all the secret keys are exposed. The notion from [BKM09] is a single challenge notion with a two-staged adversary but implies a multi challenge notion by a hybrid argument as discussed in [BFG+22]. An earlier version of [BKM09] in [BKM06] defines a slightly different notion where the adversary is given the public keys and a signing oracle and first chooses a message and two indices. After that, they get the challenge signature together with the secret keys and have to guess who signed the message. Note that the adversary must choose the message and attacked users before knowing the secret keys. This can be strengthened by providing the secret keys before the challenge [BKM09]. We provide a counterexample in Appendix B.1 to illustrate the discrepancy and the need for the notion from [BKM09].

Furthermore, this approach gives a natural extension to a multi-challenge notion implied via a hybrid argument which is not directly possible for the notion from [BKM09]. The multi-challenge notion is particularly important for the use in larger protocols, for example in Section 4. The same notion was already used in the case of rings of sizes two in [BFG+22]. We formalise *multi-challenge anonymity under full key exposures* of a ring signature RSig via the game $(n, \kappa, Q_{\mathtt{Chl}})$-$\mathbf{MC\text{-}Ano}_{\mathsf{RSig}}(\mathcal{A})$ for and adversary $\mathcal{A}$, depicted in Figure 4. We define the advantage as

$$\mathrm{Adv}_{\mathsf{RSig},\mathcal{A}}^{(n,\kappa,Q_{\mathtt{Chl}})\text{-}\mathbf{MC\text{-}Ano}} := \left|\Pr[(n,\kappa,Q_{\mathtt{Chl}})\text{-}\mathbf{MC\text{-}Ano}_{\mathsf{RSig}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2}\right|.$$

Game $(n,\kappa,Q_{\mathtt{Chl}})$-**MC-Ano**$_{\mathsf{RSig}}(\mathcal{A})$

$par \xleftarrow{\$} \mathsf{Stp}(\kappa)$
for $i \in [n]$
  $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\mathtt{Chl}}(par, (sk_1, pk_1), \ldots, (sk_n, pk_n))$
return $[\![b = b']\!]$

Oracle $\mathtt{Chl}(i_0 \in [n], i_1 \in [n], \rho, m)$

if $(\rho \subseteq \{pk_1, \ldots, pk_n\}) \wedge (pk_{i_0} \in \rho) \wedge (pk_{i_1} \in \rho)$
  $\sigma \xleftarrow{\$} \mathsf{Sgn}(sk_{i_b}, \rho, m)$
  return σ
else
  return $\perp$

Figure 4. Game defining **MC-Ano** for a ring signature scheme RSig with adversary $\mathcal{A}$ making at most $Q_{\mathtt{Chl}}$ queries to $\mathtt{Chl}$.

3.2 A New Ring Signature Construction from Lattices

Construction. Our ring signature construction Gandalf is defined over $\mathcal{R}_q$ and instantiated with a trapdoor generation algorithm TpdGen and a preimage sampler PreSmp, detailed in Figure 5. The setup algorithm Stp takes as input the maximum ring size κ and outputs the system parameters. The function ψ sets an appropriate tailcut rate τ based on κ. An explicit ψ is presented in the instantiation section in Table 3. Note that we do not explicitly mention other general parameters in the construction such as the modulus q, the standard deviation s, or the quality of the trapdoor α. We refer to Table 2 for an overview of all relevant parameters and to Section 5 for a concrete parameter selection. Line 12 verifies whether the signer is actually part of the ring they intend to sign for.

$\mathsf{Stp}(\kappa)$

01 $\tau := \psi(\kappa)$
02 $\beta := \tau \cdot s \cdot \sqrt{(\kappa+1)N}$
03 $par := (\kappa, \tau, \beta) \in \mathbb{N} \times \mathbb{R} \times \mathbb{R}$
04 **return** par

Gen

05 $(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}) \xleftarrow{\$} \mathsf{TpdGen}(q, \alpha)$
06 $sk := (\boldsymbol{f}, \boldsymbol{g}) \in \mathcal{R}_q \times \mathcal{R}_q$
07 $pk := \boldsymbol{h} \in \mathcal{R}_q$
08 **return** (sk, pk)

$\mathsf{Sgn}(sk, \rho, m)$

09 **parse** $sk \to (\boldsymbol{f}, \boldsymbol{g})$
10 **parse** $\rho \to (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k)$
11 **require** $k \leq \kappa$
12 **require** $\exists\, j \in [k] : \mu(sk) = \boldsymbol{h}_j$
13 **for** $i \in [k] \setminus \{j\}$
14   $\boldsymbol{u}_i \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^N, s, \boldsymbol{0}}$
15   $\boldsymbol{c}_i := \boldsymbol{u}_i * \boldsymbol{h}_i \in \mathcal{R}_q$
16 $\boldsymbol{h} := \mathsf{H}(m, \rho) \in \mathcal{R}_q$
17 $\boldsymbol{c}_j := \boldsymbol{h} - \sum_{i \in [k] \setminus \{j\}} \boldsymbol{c}_i$
18 $(\boldsymbol{u}_j, \boldsymbol{v}) \xleftarrow{\$} \mathsf{PreSmp}(\boldsymbol{B}_{\boldsymbol{f},\boldsymbol{g}}, s, \boldsymbol{c}_j)$
19 $\sigma := (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k) \in \mathcal{R}_q^k$
20 **return** σ

$\mathsf{Ver}(\sigma, \rho, m)$

21 **parse** $\sigma \to (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)$
22 **parse** $\rho \to \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k\}$
23 $\boldsymbol{v} := \mathsf{H}(m, \rho) - \sum_{i \in [k]} \boldsymbol{u}_i * \boldsymbol{h}_i$
24 **if** $\|(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})\|_2 \leq \beta$
25   **return** 1
26 **else**
27   **return** 0

Figure 5. Construction of ring signature scheme Gandalf[$\mathsf{TpdGen}, \mathsf{PreSmp}$] $:=$ ($\mathsf{Stp}, \mathsf{Gen}, \mathsf{Sgn}, \mathsf{Ver}$) with hash function $\mathsf{H} : \{0,1\}^* \to \mathcal{R}_q$.

Lemma 7 (Correctness). The ring signature scheme GANDALF depicted in Figure 5 is $\delta(\kappa)$-correct where

$$\delta(\kappa) = \tau^{(\kappa+1)N} \cdot e^{\frac{(\kappa+1)N}{2}(1-\tau^2)},$$

with $\tau > 1$.

Proof. For $i \in [k]$ and $k \leq \kappa$ let $(sk_i, pk_i) \in \sup(\mathsf{Gen})$, $\rho := \{pk_1, \ldots, pk_k\}$, and $\tau > 1$. Applying Lemma 1 gives

$$\begin{aligned}\Pr[\mathsf{Ver}(\mathsf{Sgn}(sk_i, \rho, m), \rho, m) \neq 1] &= \Pr[\|(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})\|_2 > \beta] \\ &= \Pr[\|(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})\|_2 > \tau s\sqrt{(\kappa+1)\cdot N}] \\ &< \tau^{(\kappa+1)N} \cdot e^{\frac{(\kappa+1)N}{2}(1-\tau^2)}.\end{aligned}$$

■

ANONYMITY. In this section we show the **MC-Ano** of GANDALF. Note that anonymity is independent of the maximum ring size κ.

Theorem 1 (Gandalf MC-Ano). *For any adversary $\mathcal{A}$, making at most Q_{Chl} challenge queries, against the* **MC-Ano** *security of* GANDALF, *depicted in Figure 5, it holds*

$$\mathrm{Adv}^{(n,\kappa,Q_{\mathsf{Chl}})\textbf{-MC-Ano}}_{\text{GANDALF},\mathcal{A}} \leq Q_{\mathsf{Chl}} \cdot \delta_{KL}.$$

The proof of Theorem 1 can be found in Appendix B.1.

UNFORGEABILITY. We show that GANDALF fulfills **UF-CRA1** security, i.e. one-per-message unforgeability against chosen ring attacks. However, with an additional salt we can enhance the security of the signature scheme to achieve full **UF-CRA** security for the cost of increasing the signature size by the size of the salt. For a security level of 128 bits, this amounts to a salt of 24 byte (see Section 5). A generic transformation is shown in Appendix B.2.

Theorem 2 ($\mathcal{R}$-LWE + $\mathcal{R}$-ISIS ⇒ Gandalf UF-CRA1). *Let* TpdGen *be a trapdoor generation algorithm and* PreSmp *a preimage sampling algorithm. Then for any adversary $\mathcal{A}$, making at most Q_{Sgn} signing queries and Q_{H} random oracle queries, against the* **UF-CRA1** *security of* GANDALF[$\mathsf{TpdGen}, \mathsf{PreSmp}$] *(Figure 5) in the random oracle model, there exist adversaries $\mathcal{B}$ against $\mathcal{R}$-**LWE** and $\mathcal{C}$ against $\mathcal{R}$-**ISIS** such that*

$$\mathrm{Adv}^{(n,\kappa,Q_{\mathsf{Sgn}})\textbf{-UF-CRA1}}_{\text{GANDALF}[\mathsf{TpdGen},\mathsf{PreSmp}],\mathcal{A}} \leq Q_{\mathsf{H}} \cdot \mathrm{Adv}^{\mathcal{R}\text{-}\boldsymbol{LWE}}_{m=1,q,\alpha,s,\mathcal{B}} + c \cdot Q_{\mathsf{H}} \cdot \mathrm{Adv}^{\mathcal{R}\text{-}\boldsymbol{ISIS}}_{m=n,q,\alpha,\beta,\mathcal{C}} + \frac{c}{|\mathcal{R}_q|},$$

for $c = \sqrt{2} \cdot R_{2\lambda}(\mathsf{PreSmp} \,||\, \mathcal{D})^{Q_{\mathsf{Sgn}}}$ and $\beta = \tau s\sqrt{(\kappa+1)N}$.

Proof. Consider the sequence of games depicted in Figure 6.

Game G_0. This is the **UF-CRA1** game for RSig so by definition

$$\Pr[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1] = \mathrm{Adv}^{(n,\kappa,Q_{\mathsf{Sgn}})\textbf{-UF-CRA1}}_{\text{GANDALF},\mathcal{A}}.$$

Game G_1. In this game, the output of the random oracle is changed if there is at least one honest public key in ring ρ. The smallest index of such an honest user is denoted by i^* (see Line 21). Instead of drawing a uniform element from $\mathcal{R}_q$, we sample Gaussian distributed values $\boldsymbol{u}$ from $\mathcal{R}_q$ for each element in ρ and compute $\boldsymbol{c} := \boldsymbol{u} * \boldsymbol{h}'$. For user with index i^* in the ring, i.e. for public key $\boldsymbol{h}'_{i^*}$, we sample an additional element $\boldsymbol{v}$ from the same distribution and instead compute $\boldsymbol{c} := \boldsymbol{u} * \boldsymbol{h}'_{i^*} + \boldsymbol{v}$ to represent an honest signer and making the RO output uniformly random. Then we set the output of the random oracle to be $\boldsymbol{h} := \sum_{i \in [k]} \boldsymbol{c}_i$. Note that this is basically the signing procedure without using the knowledge of any signing key but programming

$\underline{\mathsf{G}_0 - \mathsf{G}_3}$

$\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset$
$par \xleftarrow{\$} \mathsf{Stp}(\kappa)$
for $i \in [n]$
  $(\boldsymbol{f}_i, \boldsymbol{g}_i, \boldsymbol{h}_i) \xleftarrow{\$} \mathsf{TpdGen}$
  $sk_i := (\boldsymbol{f}_i, \boldsymbol{g}_i)$
  $pk_i := \boldsymbol{h}_i$
$(\sigma^\star, \rho^\star, m^\star) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sgn},\mathsf{H}}(par, pk_1, \ldots, pk_n)$
parse $\sigma^\star \rightarrow (\boldsymbol{u}_1^\star, \ldots, \boldsymbol{u}_k^\star)$
parse $\rho^\star \rightarrow \{\boldsymbol{h}_1^\star, \ldots, \boldsymbol{h}_k^\star\}$
for $i \in [k]$
  $\boldsymbol{c}_i^\star := \boldsymbol{u}_i^\star * \boldsymbol{h}_i^\star$
if $(\cdot, \rho^\star, m^\star) \notin \mathcal{H}$ // G_3
  abort // G_3
$\boldsymbol{v}^\star := \mathsf{H}(m^\star, \rho^\star) - \sum_{i \in [k]} \boldsymbol{c}_i^\star$
return $[\![\rho^\star \subseteq \{pk_1, \ldots, pk_n\} \wedge \|(\boldsymbol{u}_1^\star, \ldots, \boldsymbol{u}_k^\star, \boldsymbol{v}^\star)\|_2 \leq \beta \wedge (\rho^\star, m^\star) \notin \mathcal{Q}]\!]$

$\underline{\mathsf{H}(m, \rho)}$

if $\exists\, \boldsymbol{h} : (\boldsymbol{h}, m, \rho) \in \mathcal{H}$
  return $\boldsymbol{h}$
$\boldsymbol{h} \xleftarrow{\$} \mathcal{R}_q$
parse $\rho \rightarrow \{\boldsymbol{h}_1', \ldots, \boldsymbol{h}_k'\}$ // $\mathsf{G}_1 - \mathsf{G}_3$
if $\rho \cap \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n\} \neq \emptyset$ // $\mathsf{G}_1 - \mathsf{G}_3$
  $i^* := \min\{i \mid \boldsymbol{h}_i' \in \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n\}\}$ // $\mathsf{G}_1 - \mathsf{G}_3$
  $(\boldsymbol{u}_{i^*}, \boldsymbol{v}) \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{2N}, s}$ // $\mathsf{G}_1 - \mathsf{G}_3$
  $\boldsymbol{c}_{i^*} := \boldsymbol{u}_{i^*} * \boldsymbol{h}_{i^*}' + \boldsymbol{v}$ // $\mathsf{G}_1 - \mathsf{G}_3$
  for $i \in [k] \setminus \{i^*\}$ // $\mathsf{G}_1 - \mathsf{G}_3$
    $\boldsymbol{u}_i \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^N, s}$ // $\mathsf{G}_1 - \mathsf{G}_3$
    $\boldsymbol{c}_i := \boldsymbol{u}_i * \boldsymbol{h}_i'$ // $\mathsf{G}_1 - \mathsf{G}_3$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(m, \rho, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})\}$ // $\mathsf{G}_1 - \mathsf{G}_3$
  $\boldsymbol{h} := \sum_{i \in [k]} \boldsymbol{c}_i$ // $\mathsf{G}_1 - \mathsf{G}_3$
$\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{h}, m, \rho)\}$
return $\boldsymbol{h}$

Oracle $\underline{\mathtt{Sgn}(i \in [n], \rho, m)}$

if $pk_i \notin \rho$
  return $\perp$
if $(\rho, m) \in \mathcal{Q}$
  return $\perp$
parse $\rho \rightarrow (\boldsymbol{h}_1', \ldots, \boldsymbol{h}_k')$
require $k \leq \kappa$
require $\exists\, j \in [k] : \boldsymbol{h}_i = \boldsymbol{h}_j'$
$\boldsymbol{h} := \mathsf{H}(m, \rho)$
for $\ell \in [k] \setminus \{j\}$
  $\boldsymbol{u}_\ell \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^N, s, \mathbf{0}}$
  $\boldsymbol{c}_\ell := \boldsymbol{u}_\ell * \boldsymbol{h}_\ell'$
$\boldsymbol{c}_j := \boldsymbol{h} - \sum_{\ell \in [k] \setminus \{j\}} \boldsymbol{c}_\ell$
$(\boldsymbol{u}_j, \boldsymbol{v}_j) \xleftarrow{\$} \mathsf{PreSmp}(\boldsymbol{B}_{\boldsymbol{f}_i, \boldsymbol{g}_i}, s, \boldsymbol{c}_j)$
$p \leftarrow \mathcal{P} : p = (m, \rho, \ldots)$ // $\mathsf{G}_2 - \mathsf{G}_3$
parse $p \rightarrow (m, \rho, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})$ // $\mathsf{G}_2 - \mathsf{G}_3$
$\sigma := (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\rho, m)\}$
return σ

Figure 6. Games $\mathsf{G}_0 - \mathsf{G}_3$ for the proof of Theorem 2.

the random oracle. Further, we store the preimage $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})$ together with ring and message in set $\mathcal{P}$ for later use.

Claim 1: There exists a PPT adversary $\mathcal{B}$ against $\mathcal{R}$-$\mathbf{LWE}_{1,q,\alpha,s}$ such that

$$\left|\Pr\left[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1\right]\right| \leq Q_{\mathsf{H}} \cdot \mathrm{Adv}_{1,q,\alpha,s,\mathcal{B}}^{\mathcal{R}\text{-}\mathbf{LWE}}.$$

Proof. In Game G_0, the output of the random oracle is $\boldsymbol{h} \xleftarrow{\$} \mathcal{R}_q$. In the first step, we compute $\boldsymbol{h}$ as $\boldsymbol{h} \leftarrow \sum_{i \in [k] \setminus \{i^*\}} \boldsymbol{c}_i + \boldsymbol{c}_{i^*}$ where the $\boldsymbol{c}_i$ are computed as in Game G_1 except for $\boldsymbol{c}_{i^*}$ which is chosen uniformly at random from $\mathcal{R}_q$. This change is perfectly indistinguishable. Next, we replace $\boldsymbol{c}_{i^*} \xleftarrow{\$} \mathcal{R}_q$ by $\boldsymbol{c}_{i^*} \leftarrow \boldsymbol{u} * \boldsymbol{h}_{i^*}' + \boldsymbol{v}$

with $(\boldsymbol{u}, \boldsymbol{v}) \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{2N},s,\mathbf{0}}$. This change can be reduced to $\mathcal{R}$-**LWE** with one sample ($m = 1$). Applying this change Q_{H} times results in Game G_1 using a hybrid argument. ∎

Game G_2. In this game, the signing oracle is simulated without using the signing key. To this end, the stored preimages from the random oracle query are used as a signature (Line 45).

Claim 2:

$$\Pr[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1] \leq \sqrt{2} \cdot R_{2\lambda}(\mathsf{PreSmp} \mid\mid \mathcal{D})^{Q_{\mathsf{Sgn}}} \cdot \Pr[\mathsf{G}_2^{\mathsf{A}} \Rightarrow 1].$$

Proof. The difference is that the output of the preimage sampler $\boldsymbol{u}_j, \boldsymbol{v}_j$ is now replaced by a random value drawn from distribution $\mathcal{D}_{\mathbb{Z}^{2N},s,\mathbf{0}}$ conditioned on the ring equation, i.e. $\boldsymbol{u}_j * \boldsymbol{h}_j + \boldsymbol{v}_j = \boldsymbol{h} - \sum_{\ell \in [k] \setminus \{j\}} \boldsymbol{c}_\ell$. According to [Pre17, Sec. 3.3], we get

$$\frac{\Pr[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1]}{\Pr[\mathsf{G}_2^{\mathsf{A}} \Rightarrow 1]} \leq \sqrt{2} \cdot R_{2\lambda}(\mathcal{P} \mid\mid \mathcal{Q})^{Q_{\mathsf{Sgn}}}$$

since the only difference between G_1 and G_2 is the access to the underlying distributions of $\mathsf{PreSmp}/\mathcal{D}$ and there are at most Q_{Sgn} queries to these distributions. The preimage can be used independent of signer i since the distribution of the $\boldsymbol{u}$'s as well as $\boldsymbol{v}$ is always the same. The procedure only works if there is at least one honest user in the ring which must be satisfied in the signing oracle, by definition of a ring signature. Note that the signature is now the same for any fixed pair (ρ, m) but this is not observable for a one-per-message adversary. ∎

Game G_3. Game G_3 aborts if the adversary did not query the random oracle on the forgery, i.e. did not issue a query $\mathsf{H}(\rho^\star, m^\star)$.

Claim 3:

$$\left|\Pr\left[\mathsf{G}_2^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_3^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \frac{1}{|\mathcal{R}_q|}.$$

Proof. If the adversary does not query the RO on the forgery parameters $\rho^\star$ and $m^\star$, the chances of winning the game are at most $\frac{1}{|\mathcal{R}_q|}$ since $\mathcal{R}_q$ is the output space of the RO. Moreover, the signing oracle does not reveal any information of the RO to the adversary since the same ring and message cannot be used for a valid forgery. ∎

Reduction to G_3. To upper bound the winning probability of Game G_3, we show that there exists an adversary $\mathcal{C}$ against $\mathcal{R}$-**ISIS**.

Claim 4: There exists a PPT adversary $\mathcal{C}$ against $\mathcal{R}$-$\mathbf{ISIS}_{n,q,\alpha,\beta}$ such that

$$\Pr[\mathsf{G}_3^{\mathsf{A}} \Rightarrow 1] \leq Q_{\mathsf{H}} \cdot \mathrm{Adv}^{\mathcal{R}\text{-}\mathbf{ISIS}}_{m=n,q,\alpha,\beta,\mathcal{C}}.$$

Proof. Adversary $\mathcal{C}$ is formally constructed in Figure 7. They guess a random oracle query in the beginning of the game (Line 02) and embed the ISIS challenge in this query to the random oracle in Line 43. However, we only consider explicit queries to the RO and no implicit queries from the singing oracle (see condition in Line 40). If the guess is correct, reduction $\mathcal{C}$ returns an ISIS solution; this happens with probability $\frac{1}{Q_{\mathsf{H}}}$. Then solution $(\hat{\boldsymbol{u}}_1, \ldots . \hat{\boldsymbol{u}}_n, \boldsymbol{v}^\star)$ is correct since

$$\boldsymbol{h}_1 * \hat{\boldsymbol{u}}_1 + \ldots + \boldsymbol{h}_n * \hat{\boldsymbol{u}}_n + \boldsymbol{v}^\star = \mathsf{H}(m^\star, \rho^\star) = \boldsymbol{c}$$

due to Line 13, Line 16, and the fact that all the $\boldsymbol{u}$'s are 0 for all the $\boldsymbol{h}$'s not being in $\rho^\star$ (Line 24). Taking the guessing probability into account includes the abort in Line 19 because the abort only occurs if the guess was wrong: if the guess was correct, the adversary cannot query the signing oracle on the challenge message and win the game. Further, it holds $\|(\boldsymbol{u}_1^\star, \ldots, \boldsymbol{u}_k^\star, \boldsymbol{v}^\star)\|_2 \leq \beta$ because $\mathcal{A}$ returned a valid signature (Line 17). This implies $\|(\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_n, \boldsymbol{v}^\star)\|_2 \leq \beta$ since all the $\boldsymbol{u}$'s not occurring in $(\boldsymbol{u}_1^\star, \ldots, \boldsymbol{u}_k^\star)$ were set to 0 (Line 24). ∎

Collecting the terms, we obtain the stated bound of the theorem. ∎

$\underline{\mathcal{C}(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n, \boldsymbol{c})}$

$\ell \leftarrow 0$
$\ell^* \xleftarrow{\$} [Q_{\mathsf{H}}]$ // guess RO query
$\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset$
$par \xleftarrow{\$} \mathsf{Stp}(\kappa)$
for $i \in [n]$
    $(\boldsymbol{f}_i, \boldsymbol{g}_i, \boldsymbol{h}_i) \xleftarrow{\$} \mathsf{TpdGen}$
    $sk_i := (\boldsymbol{f}_i, \boldsymbol{g}_i)$
    $pk_i := \boldsymbol{h}_i$
$(\sigma^\star, \rho^\star, m^\star) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sgn},\mathsf{H}}(par, pk_1, \ldots, pk_n)$
parse $\sigma^\star \to (\boldsymbol{u}_1^\star, \ldots, \boldsymbol{u}_k^\star)$
parse $\rho^\star \to \{\boldsymbol{h}_1^\star, \ldots, \boldsymbol{h}_k^\star\}$
for $i \in [k]$
    $\boldsymbol{c}_i^\star := \boldsymbol{u}_i^\star * \boldsymbol{h}_i^\star$
if $(\cdot, \rho^\star, m^\star) \notin \mathcal{H}$
    abort
$\boldsymbol{v}^\star := \mathsf{H}(m^\star, \rho^\star) - \sum_{i \in [k]} \boldsymbol{c}_i^\star$
if $[\![\rho^\star \subseteq \{pk_1, \ldots, pk_n\} \wedge \|(\boldsymbol{u}_1^\star, \ldots, \boldsymbol{u}_k^\star, \boldsymbol{v}^\star)\|_2 \leq \beta \wedge (\rho^\star, m^\star) \notin \mathcal{Q}]\!]$
    if $\mathsf{H}(m^\star, \rho^\star) \neq \boldsymbol{c}$
        return $\perp$ // wrong guess
    for $i \in [n]$
        if $\exists\, j : \boldsymbol{h}_i = \boldsymbol{h}_j^\star$
            $\hat{\boldsymbol{u}}_i := \boldsymbol{u}_j^\star$
        else
            $\hat{\boldsymbol{u}}_i := 0$
    return $(\hat{\boldsymbol{u}}_1, \ldots \hat{\boldsymbol{u}}_n, \boldsymbol{v}^\star)$ // return ISIS solution
return $\perp$

$\underline{\mathsf{H}(m, \rho)}$

if $\exists\, \boldsymbol{h} : (\boldsymbol{h}, m, \rho) \in \mathcal{H}$
    return $\boldsymbol{h}$
$\boldsymbol{h} \xleftarrow{\$} \mathcal{R}_q$
parse $\rho \to \{\boldsymbol{h}_1', \ldots, \boldsymbol{h}_k'\}$
if $\rho \cap \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n\} \neq \emptyset$
    $i^* := \min\{i \mid \boldsymbol{h}_i' \in \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n\}\}$
    $(\boldsymbol{u}_{i^*}, \boldsymbol{v}) \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{2N}, s}$
    $\boldsymbol{c}_{i^*} := \boldsymbol{u}_{i^*} * \boldsymbol{h}_{i^*}' + \boldsymbol{v}$
    for $i \in [k] \setminus \{i^*\}$
        $\boldsymbol{u}_i \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^N, s}$
        $\boldsymbol{c}_i := \boldsymbol{u}_i * \boldsymbol{h}_i'$
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{(m, \rho, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})\}$
    $\boldsymbol{h} := \sum_{i \in [k]} \boldsymbol{c}_i$
if Query is not from Sgn
    $\ell := \ell + 1$ // count direct queries
if $\ell = \ell^*$
    $\boldsymbol{h} := c$ // embed ISIS challenge
$\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{h}, m, \rho)\}$
return $\boldsymbol{h}$

Oracle $\underline{\mathsf{Sgn}(i \in [n], \rho, m)}$

if $pk_i \notin \rho$
    return $\perp$
if $(\rho, m) \in \mathcal{Q}$
    return $\perp$
parse $\rho \to (\boldsymbol{h}_1', \ldots, \boldsymbol{h}_k')$
require $k \leq \kappa$
require $\exists\, j \in [k] : \boldsymbol{h}_i = \boldsymbol{h}_j'$
$\boldsymbol{h} := \mathsf{H}(m, \rho)$
for $\ell \in [k] \setminus \{j\}$
    $\boldsymbol{u}_\ell \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^N, s, \boldsymbol{0}}$
    $\boldsymbol{c}_\ell := \boldsymbol{u}_\ell * \boldsymbol{h}_\ell'$
$\boldsymbol{c}_j := \boldsymbol{h} - \sum_{\ell \in [k] \setminus \{j\}} \boldsymbol{c}_\ell$
$(\boldsymbol{u}_j, \boldsymbol{v}_j) \xleftarrow{\$} \mathsf{PreSmp}(\boldsymbol{B}_{\boldsymbol{f}_i, \boldsymbol{g}_i}, s, \boldsymbol{c}_j)$
$p \leftarrow \mathcal{P} : p = (m, \rho, \ldots)$
parse $p \to (m, \rho, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})$
$\sigma := (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\rho, m)\}$
return σ

Figure 7. Adversary $\mathcal{C}$ against $\mathcal{R}$-**ISIS** simulating G_3 for adversary $\mathcal{A}$ for the proof of Theorem 2.

4 Deniable AKEM

4.1 Syntax and Security

Definition 11 (Authenticated Key Encapsulation Mechanism). An *authenticated key encapsulation mechanism* AKEM is defined as a tuple AKEM := (Gen, Enc, Dec) of the following PPT algorithms.

$(sk, pk) \xleftarrow{\$}$ **Gen:** The probabilistic generation algorithm Gen returns a secret key sk and a corresponding public key pk. We implicitly assume the existence of a shared key space $\mathcal{K}$.

$(c,k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk_r)$**:** Given a sender's secret key sk_s and a receiver's public key pk_r, the probabilistic encapsulation algorithm Enc returns a ciphertext c and a shared key $k \in \mathcal{K}$.

$k \leftarrow \mathsf{Dec}(pk_s, sk_r, c)$**:** Given a sender's public key pk_s, a receiver's secret key sk_r, and a ciphertext c, the deterministic decapsulation algorithm Dec returns a shared key $k \in \mathcal{K}$, or a failure symbol $\perp$.

The correctness error δ is defined as

$$\delta := \Pr\left[\mathsf{Dec}(pk_s, sk_r, c) \neq k \;\middle|\; \begin{array}{r} (sk_s, pk_s) \xleftarrow{\$} \mathsf{Gen} \\ (sk_r, pk_r) \xleftarrow{\$} \mathsf{Gen} \\ (c,k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk_r) \end{array}\right].$$

Without loss of generality we assume the existence of an efficiently computable function μ such that for all $(sk, pk) \in \mathsf{Gen}$ it holds $\mu(sk) = pk$.

Confidentiality. We consider the strongest possible notion of CCA security for an AKEM, in particular that of insider security [AJKL23]. The details have been deferred to Appendix C.

Authenticity. We explore two notions of *authenticity*, outsider and insider authenticity. The outsider notion for AKEMs is taken from [ABH+21], the insider notion we adapt from the outsider notion. The difference is in the challenge oracle, i.e. the oracle for which the adversary has to decide if they get the real decapsulation or a randomly sampled key. In the outsider setting, an adversary can choose an arbitrary sender's public key along with an honest receiver. In contrast, an insider adversary can choose a receiver's secret key by themselves which models a scenario in which it should be hard to distinguish between a real and a random decapsulation even for the designated receiver party. Note that the sender's key cannot be chosen by the adversary because otherwise distinguishing becomes trivial. While insider authenticity implies outsider authenticity [DZ10], we focus on the latter because it remains achievable when also considering deniability. We formalise the two notions via the games $(n, Q_{\mathtt{Enc}}, Q_{\mathtt{Chl}})$-$\mathbf{Out\text{-}Aut}_{\mathsf{AKEM}}(\mathcal{A})$ (for outsider authenticity) and $(n, Q_{\mathtt{Enc}}, Q_{\mathtt{Dec}}, Q_{\mathtt{Chl}})$-$\mathbf{Ins\text{-}Aut}_{\mathsf{AKEM}}(\mathcal{A})$ (for insider authenticity) depicted in Figure 8 and define the advantage of an adversary $\mathcal{A}$ as

$$\mathrm{Adv}_{\mathsf{AKEM},\mathcal{A}}^{(n,Q_{\mathtt{Enc}},Q_{\mathtt{Chl}})\text{-}\mathbf{Out\text{-}Aut}} := \left|\Pr\left[(n, Q_{\mathtt{Enc}}, Q_{\mathtt{Chl}})\text{-}\mathbf{Out\text{-}Aut}_{\mathsf{AKEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2}\right| \text{ and}$$

$$\mathrm{Adv}_{\mathsf{AKEM},\mathcal{A}}^{(n,Q_{\mathtt{Enc}},Q_{\mathtt{Dec}},Q_{\mathtt{Chl}})\text{-}\mathbf{Ins\text{-}Aut}} := \left|\Pr\left[(n, Q_{\mathtt{Enc}}, Q_{\mathtt{Dec}}, Q_{\mathtt{Chl}})\text{-}\mathbf{Ins\text{-}Aut}_{\mathsf{AKEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2}\right|.$$

4.2 Deniability for AKEMs

Deniability aims to model a scenario where a sender sends a potentially incriminating message to a receiver. The aim is to prevent a third party, the judge (modelled as an adversary) from conclusively attributing, potentially incriminating, messages to a particular sender. This means that authentication of the sender should be non-transferable, allowing the sender to plausibly deny their involvement.

More formally, we assume the existence of a PPT simulator Sim, which is capable of generating a ciphertext c and key k that is indistinguishable from those generated by the encapsulation Enc procedure. Such a simulator enables the sender to plausibly deny sending specific messages, as an adversary could have generated the same messages using the simulator. Depending on the scenario, the simulator may have the secret key of the receiver in addition to the public keys of the involved parties. This case represents the setting of a potentially *dishonest receiver* which means that the receiver could have potentially forged the ciphertext such that it looks like it came from the sender. If the simulator does not have access to the receiver's secret key, we consider an **honest receiver** and the judge knows about this fact. Note that this distinguishes the two settings and security in the dishonest receiver setting does not imply security in the honest receiver

Games $(n, Q_{\texttt{Enc}}, Q_{\texttt{Chl}})$-**Out-Aut**$_{\mathsf{AKEM}}(\mathcal{A})$
$(n, Q_{\texttt{Enc}}, Q_{\texttt{Dec}}, Q_{\texttt{Chl}})$-**Ins-Aut**$_{\mathsf{AKEM}}(\mathcal{A})$

for $i \in [n]$
  $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
$\mathcal{D} \leftarrow \emptyset$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\texttt{Encps},\texttt{Chall}}(pk_1, \ldots, pk_n)$ // **Out-Aut**
$b' \xleftarrow{\$} \mathcal{A}^{\texttt{Encps},\texttt{Decps},\texttt{Chall}}(pk_1, \ldots, pk_n)$ // **Ins-Aut**
return $[\![b = b']\!]$

Oracle $\texttt{Chall}(pk, r \in [n], c)$ // **Out-Aut**

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
$k \leftarrow \mathsf{Dec}(pk, sk_r, c)$
if $b = 0$
  continue
if $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \perp$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
return k

Oracle $\texttt{Encps}(s \in [n], pk)$

$(c, k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk)$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

Oracle $\texttt{Decps}(pk, r \in [n], c)$ // **Ins-Aut**

$k \leftarrow \mathsf{Dec}(pk, sk_r, c)$
return k

Oracle $\texttt{Chall}(s \in [n], sk, c)$ // **Ins-Aut**

if $\exists\, k : (pk_s, \mu(sk), c, k) \in \mathcal{D}$
  return k
$k \leftarrow \mathsf{Dec}(pk_s, sk, c)$
if $b = 0$
  continue
if $b = 1 \wedge k \neq \perp$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, \mu(sk), c, k)\}$
return k

Figure 8. Games defining **Out-Aut** and **Ins-Aut** for an authenticated key encapsulation mechanism AKEM with adversary $\mathcal{A}$ making at most $Q_{\texttt{Enc}}$ queries to **Encps**, at most $Q_{\texttt{Chl}}$ queries to **Chall**, and at most $Q_{\texttt{Dec}}$ queries to **Decps** (for **Ins-Aut**).

setting. [5] However, security in the honest setting implies security in the dishonest since the capabilities of the simulator increases and the rest stays the same.

Furthermore, different notions of deniability exist, varying in strength depending on the capabilities of the judge, modelled as the adversary $\mathcal{A}$, and which secret keys are accessible to the judge. This results in four distinct notions of deniability (for each setting of honest and dishonest receivers).

An overview of the deniability notions is presented in Table 1. For each column of the table, we observe that the deniability notion at the bottom is stronger than the one above it, since the simulator is give the same capabilities but judge $\mathcal{A}$ has more information (the secret key of the sender). For the same reason, in both the honest and dishonest receiver settings, the right column is a stronger notion than the one to the left. Further, one field in the honest setting implies the same field in the dishonest setting. Consequently, the deniability notion at the bottom right of each setting is the strongest [CHMR23]. However, in the honest receiver setting, authenticity and correctness of an AKEM imply that achieving this notion is impossible [DHM+20]. Instead, the strongest achievable notion in the honest setting is one where the sender's key is leaked while the receiver's does not. This shows that it is still relevant to consider the dishonest setting since the bottom right notion of the dishonest setting is not implied by any achievable notion of the honest setting.

While our model primarily addresses the deniability of specific messages, our definitions focus on a KEM-like primitive that returns a key/ciphertext pair, rather than a message. However, if the denial of a common secret (the KEM key) is possible, the same applies to messages encrypted using that key. For all our security notions, we consider the multi-user setting. We formally define the strongest achievable notions in the honest and dishonest receiver setting in Figure 9. For an AKEM and a PPT simulator Sim we define deniability

[5] For example, consider implicit authentication via a NIKE. In the dishonest setting the sender can always deny since the receiver could have created the shared key. In the honest setting, the judge knows that the receiver does not maliciously authenticate ciphertexts to themselves. Hence, if the judge sees a valid ciphertext the sender must have created it.

Table 1. Different deniability notions for an authenticated key encapsulation mechanism AKEM for honest and dishonest receivers. Notions where sk_s leaks imply those where sk_s does not leak, and similarly for sk_r. The strongest notions are marked in green, while those not achievable are marked in red.

		Honest Receiver		Dishonest Receiver	
		sk_r does not leak	sk_r leaks	sk_r does not leak	sk_r leaks
Honest Sender	sk_s does not leak	$\mathsf{Sim}(\emptyset), \mathcal{A}(\emptyset)$	$\mathsf{Sim}(\emptyset), \mathcal{A}(sk_r)$	$\mathsf{Sim}(sk_r), \mathcal{A}(\emptyset)$	$\mathsf{Sim}(sk_r), \mathcal{A}(sk_r)$
	sk_s leaks	$\mathsf{Sim}(\emptyset), \mathcal{A}(sk_s)$	$\mathsf{Sim}(\emptyset), \mathcal{A}(sk_s, sk_r)$	$\mathsf{Sim}(sk_r), \mathcal{A}(sk_s)$	$\mathsf{Sim}(sk_r), \mathcal{A}(sk_s, sk_r)$

in the dishonest receiver setting via game $(n, Q_{\texttt{Chl}})$-**DR-Den** (for dishonest receiver deniability) and in the honest receiver setting via game $(n, Q_{\texttt{Chl}})$-**HR-Den** (for honest receiver deniability) depicted in Figure 9 and define the advantage of adversary $\mathcal{A}$ as

$$\mathrm{Adv}^{(n,Q_{\texttt{Chl}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM},\mathcal{A},\mathsf{Sim}} := \left|\Pr[(n, Q_{\texttt{Chl}})\text{-}\mathbf{DR\text{-}Den}_{\mathsf{AKEM},\mathsf{Sim}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2}\right|,$$

$$\mathrm{Adv}^{(n,Q_{\texttt{Chl}})\text{-}\mathbf{HR\text{-}Den}}_{\mathsf{AKEM},\mathcal{A},\mathsf{Sim}} := \left|\Pr[(n, Q_{\texttt{Chl}})\text{-}\mathbf{HR\text{-}Den}_{\mathsf{AKEM},\mathsf{Sim}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2}\right|.$$

Games $(n, Q_{\texttt{Chl}})$-**DR-Den**$_{\mathsf{AKEM},\mathsf{Sim}}(\mathcal{A})$
$(n, Q_{\texttt{Chl}})$-**HR-Den**$_{\mathsf{AKEM},\mathsf{Sim}}(\mathcal{A})$

$\mathcal{R}, \mathcal{C} \leftarrow \emptyset$
for $i \in [n]$
  $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
$b \xleftarrow{\$} \{0, 1\}$
$b' \leftarrow \mathcal{A}^{\texttt{Rev},\texttt{Chall}}(pk_1, \ldots, pk_n)$
if $\mathcal{R} \cap \mathcal{C} \neq \emptyset$ // **HR-Den**
  return $b \xleftarrow{\$} \{0, 1\}$ // **HR-Den**
return $[\![b = b']\!]$

$\texttt{Rev}(i \in [n])$

$\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$
return sk_i

Oracle $\texttt{Chall}(s \in [n], r \in [n])$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{r\}$
$(c, k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk_r)$
if $b = 0$
  continue
if $b = 1$
  $(c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r, sk_r)$ // **DR-Den**
  $(c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r)$ // **HR-Den**
return (c, k)

Figure 9. Games defining **DR-Den** and **HR-Den** for an AKEM AKEM and a simulator Sim for adversary $\mathcal{A}$ where $\mathcal{A}$ makes at most $Q_{\texttt{Chl}}$ queries to $\texttt{Chall}$.

A Note on Authenticity and Deniability. The goal of deniable authentication is to achieve both authenticity and deniability at the same time. Recall, that for an AKEM there are two different settings for authenticity, the weaker outsider setting and the stronger insider setting. In the outsider setting, it is possible to achieve the strongest notions for deniability, as shown in Table 1, without losing any authenticity guarantees. However, in the insider setting, the strongest notion cannot always be achieved. For example, simultaneously achieving **Ins-Aut** security and **DR-Den** for an AKEM is not always possible. This limitation stems from the inherent conflict: if the scheme is **DR-Den** secure, there exists a simulator such that the judge having all the secret keys cannot distinguish the simulated output from the real output. However, such a simulator can be used to query the challenge in the **Ins-Aut** game and easily distinguish the output. Note that this attack works because the adversary in game **Ins-Aut** can issue challenge queries on corrupted receivers, i.e. choose the secret key of the receiver. For the honest setting it is not clear what authenticity notions can be achieved. We leave this as an interesting open question.

4.3 Generic Construction

In the following, we show a construction of an AKEM $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ from a KEM $\mathsf{KEM} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, a ring signature $\mathsf{RSig} := (\mathsf{Stp}, \mathsf{Gen}, \mathsf{Sgn}, \mathsf{Ver})$, a symmetric encryption scheme $\mathsf{SyE} := (\mathsf{Enc}, \mathsf{Dec})$, and a keyed function H. The ring signature is applied with $\mathsf{Stp}(2)$.

<u>Gen</u>

$(ksk, kpk) \xleftarrow{\$} \mathsf{KEM.Gen}$
$(ssk, spk) \xleftarrow{\$} \mathsf{RSig.Gen}$
$sk := (ksk, ssk)$
$pk := (kpk, spk)$
return (sk, pk)

<u>$\mathsf{Enc}(sk_s, pk_r)$</u>

parse $sk_s \to (ksk_s, ssk_s)$
parse $pk_r \to (kpk_r, spk_r)$
$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$m \leftarrow (kct, \mu(ksk_s), kpk_r, spk_r)$
$\sigma' \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{\mu(ssk_s), spk_r\}, m)$
$kk \to kk_1 || kk_2$
$\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
$c := (kct, \sigma)$
$k := \mathsf{H}(kk_2, \sigma, \mu(ssk_s), m)$
return (c, k)

<u>$\mathsf{Dec}(pk_s, sk_r, c)$</u>

parse $pk_s \to (kpk_s, spk_s)$
parse $sk_r \to (ksk_r, ssk_r)$
parse $c \to (kct, \sigma)$
$kk \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
$kk \to kk_1 || kk_2$
$\sigma' \leftarrow \mathsf{SyE.Dec}_{kk_1}(\sigma)$
$m \leftarrow (kct, kpk_s, \mu(ksk_r), \mu(ssk_r))$
if $\mathsf{RSig.Ver}(\sigma', \rho = \{spk_s, \mu(ssk_r)\}, m) \neq 1$
  return $\perp$
$k := \mathsf{H}(kk_2, \sigma, spk_s, m)$
return k

Figure 10. Authenticated Key Encapsulation Mechanism $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$.

Theorem 3 (KEM **IND-CCA** + H **PRF** $\Longrightarrow$ AKEM **Ins-CCA**). *Let* KEM *be an* **IND-CCA** *secure key encapsulation mechanism and* H *a* **PRF**, *then* $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ *as depicted in Figure 10 is an* **Ins-CCA** *secure authenticated key encapsulation mechanism. In particular for any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ *there exist a* **IND-CCA** *adversary* $\mathcal{B}$ *against* KEM *and a* **PRF** *adversary* $\mathcal{C}$ *against* H *such that*

$$\mathrm{Adv}^{(n, \mathrm{Q}_{Enc} \mathrm{Q}_{Dec}, \mathrm{Q}_{CSK}, \mathrm{Q}_{Chl})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}[\mathsf{KEM},\mathsf{RSig},\mathsf{SyE},\mathsf{H}],\mathcal{A}} \leq \mathrm{Adv}^{(n, \mathrm{Q}_{Dec}, \mathrm{Q}_{Chl})\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{B}} + \mathrm{Adv}^{(\mathrm{Q}_{Chl}, \mathrm{Q}_{Dec}+\mathrm{Q}_{Chl})\text{-}\mathbf{PRF}}_{\mathsf{H},\mathcal{C}}.$$

The proof of Theorem 3 can be found in Appendix C.

Theorem 4 (KEM **IND-CCA** + RSig **UF-CRA1** + H **PRF** $\Longrightarrow$ AKEM **Out-Aut**). *Let* KEM *be an* **IND-CCA** *secure key encapsulation mechanism,* RSig *an* **UF-CRA1** *secure ring signature scheme, and* H *a* **PRF**, *then* $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ *as depicted in Figure 10 is an* **Out-Aut** *secure authenticated key encapsulation mechanism. In particular, for any* **Out-Aut** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ *there exist a* **UF-CRA1** *adversary* $\mathcal{B}$ *against* RSig, *an* **IND-CCA** *adversary* $\mathcal{C}$ *against* KEM, *and a* **PRF**

adversary $\mathcal{D}$ *against* H *such that*

$$\mathrm{Adv}^{(n,Q_{Enc},Q_{Chl})\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM[KEM,RSig,SyE,H]},\mathcal{A}} \le \mathrm{Adv}^{(n,2,Q_{Enc})\text{-}\mathbf{UF\text{-}CRA1}}_{\mathsf{RSig},\mathcal{B}} + \mathrm{Adv}^{(n,Q_{Chl},Q_{Enc})\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{C}} + \mathrm{Adv}^{(Q_{Enc},Q_{Enc}+Q_{Chl})\text{-}\mathbf{PRF}}_{\mathsf{H},\mathcal{D}} + Q_{Enc}^2 \cdot \gamma_{\mathsf{KEM}}.$$

Proof. Consider the sequence of games depicted in Figure 11.

Game G_0. This is the **Out-Aut** game for $\mathsf{AKEM[KEM,RSig,SyE,H]}$ so by definition

$$\left|\Pr[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,Q_{\mathtt{Dec}},Q_{\mathtt{Chl}})\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM[KEM,RSig,SyE,H]},\mathcal{A}}.$$

$\mathsf{G}_0 - \mathsf{G}_6$

- **for** $i \in [n]$
  - $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  - $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
  - $sk_i := (ksk_i, ssk_i)$
  - $pk_i := (kpk_i, spk_i)$
- $\mathcal{D}, \mathcal{Q}, \mathcal{Q}', \mathcal{E}_{\mathsf{KEM}}, \mathcal{H} \leftarrow \emptyset$
- $b \xleftarrow{\$} \{0,1\}$
- $b' \xleftarrow{\$} \mathcal{A}^{\mathtt{Encps},\mathtt{Chall}}(pk_1, \ldots, pk_n)$
- **return** $[\![b = b']\!]$

Oracle $\mathtt{Encps}(s \in [n], pk)$

- **parse** $pk \to (kpk, spk)$
- $(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
- **if** $kpk \in \{kpk_1, \ldots, kpk_m\}$ // $\mathsf{G}_4 - \mathsf{G}_6$
  - $kk \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$ // $\mathsf{G}_4 - \mathsf{G}_6$
  - $\mathcal{E}_{\mathsf{KEM}} \leftarrow \mathcal{E}_{\mathsf{KEM}} \cup \{(kpk, kct, kk)\}$ // $\mathsf{G}_4 - \mathsf{G}_6$
- $m \leftarrow kct||kpk_s||kpk||spk$
- **if** $(\{spk_s, spk\}, m) \in \mathcal{Q}'$ // $\mathsf{G}_2 - \mathsf{G}_6$
  - **abort** // $\mathsf{G}_2 - \mathsf{G}_6$
- $\sigma' \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk\}, m)$
- $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk_s, spk\}, m, \sigma')\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
- $\mathcal{Q}' \leftarrow \mathcal{Q}' \cup \{(\{spk_s, spk\}, m)\}$ // $\mathsf{G}_2 - \mathsf{G}_6$
- $kk \to kk_1||kk_2$
- $\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
- $c := (kct, \sigma)$
- $k := \mathsf{H}(kk_2, \sigma||spk_s||m)$
- **if** $kpk \in \{kpk_1, \ldots, kpk_n\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
  - $k \xleftarrow{\$} \mathcal{K}$ // G_6
  - $\mathcal{H} \leftarrow \mathcal{H} \cup \{(k, kk_2, \sigma, spk_s, m)\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
- $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
- **return** (c, k)

Oracle $\mathtt{Chall}(pk, r \in [n], c)$

- $\mathtt{Flag} \leftarrow$ **false**
- **if** $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  - **return** k
- **parse** $pk \to (kpk, spk)$
- **parse** $c \to (kct, \sigma)$
- $m \leftarrow kct||kpk||kpk_r||spk_r$
- $kk \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
- **if** $\exists\, kk' : (kpk_r, kct, kk') \in \mathcal{E}_{\mathsf{KEM}}$ // $\mathsf{G}_4 - \mathsf{G}_6$
  - $kk \leftarrow kk'$ // $\mathsf{G}_4 - \mathsf{G}_6$
  - $\mathtt{Flag} \leftarrow$ **true** // $\mathsf{G}_4 - \mathsf{G}_6$
- $kk \to kk_1||kk_2$
- $k \leftarrow \mathsf{H}(kk_2, \sigma||spk||m)$
- $\sigma' \leftarrow \mathsf{SyE.Dec}_{kk_1}(\sigma)$
- **if** $\mathsf{RSig.Ver}(\sigma', \{spk, spk_r\}, m) \neq 1$
  - $k \leftarrow \perp$
- **elseif** $pk \in \{pk_1, \ldots, pk_n\} \wedge (\{spk, spk_r\}, m, \cdot) \notin \mathcal{Q}$ // $\mathsf{G}_1 - \mathsf{G}_6$
  - **abort** // $\mathsf{G}_3 - \mathsf{G}_6$
  - $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk, spk_r\}, m, \sigma')\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
  - $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
- **elseif** $pk \in \{pk_1, \ldots, pk_n\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
  - **if** $\exists\, k' : (k', kk_2, \sigma, spk, m) \in \mathcal{H} \cup \mathcal{H}_E$ // $\mathsf{G}_5 - \mathsf{G}_6$
    - **abort** // $\mathsf{G}_5 - \mathsf{G}_6$
  - $k \xleftarrow{\$} \mathcal{K}$ // G_6
  - $\mathcal{H} \leftarrow \mathcal{H} \cup \{(k, kk_2, \sigma, spk, m)\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
  - $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk, spk_r\}, m, \sigma')\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
  - $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
- **if** $b = 0$
  - **continue**
- **if** $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \perp$
  - $k \xleftarrow{\$} \mathcal{K}$
  - $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
- **return** k

Figure 11. Games $\mathsf{G}_0 - \mathsf{G}_6$ for the proof of Theorem 4.

Game G_1. Here, several conceptual changes are introduced.

First, the encapsulation oracle is modified to store the signing results in a set $\mathcal{Q}$ which contains elements of the form $(\{spk_s, spk_r\}, m, \sigma')$. Here, spk_s and spk_r (this is spk in the encapsulation oracle) represent the signature public keys for the sender and receiver, and m denotes the message to be signed (specifically, $m = kct||kpk_s||kpk||spk$). The challenge oracle also populates the same set $\mathcal{Q}$ when the sender key is honest and the signature is valid, as indicated on Line 47 and Line 54. In the challenge oracle, we extend the bookkeeping set $\mathcal{D}$ (regardless of the challenge bit b) whenever the sender key pk is honest and $k \neq \perp$, as shown on Line 48 and Line 55. Finally, a bookkeeping set $\mathcal{H}$ is introduced to store the inputs and the output of the hash invocation of H. This is done in the `Encps` oracle if the receiver key is honest (Line 27 and in the `Chall` on Line 53, i.e. in case of honest sender keys and a new signature on an old message $((\{spk, spk_r, m, \cdot) \in \mathcal{Q})$. As these changes are just conceptual,

$$\Pr[\mathsf{G}_0 \Rightarrow 1] = \Pr[\mathsf{G}_1 \Rightarrow 1].$$

Game G_2. In G_2, the game aborts in the encapsulation oracle if a ring/message pair $\{spk_s, spk\}/m$ is used for the signature procedure which was used before. To this end, we store the inputs to the signing procedure in a set $\mathcal{Q}'$ (Line 20) and abort the game if the same query occurs again (Line 17).

Claim 5:

$$\left|\Pr\left[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathsf{A}} \Rightarrow 1\right]\right| \leq Q_{\mathtt{Enc}}^2 \cdot \gamma_{\mathsf{KEM}}.$$

Proof. For every query to the encapsulation oracle `Encps`, a new KEM ciphertext kct is created. Since kct is part of the message m to be signed, the probability that a particular message occurs is at most γ_{KEM}. Set $\mathcal{Q}'$ contains at most $Q_{\mathtt{Enc}}$ elements and the event can happen at most $Q_{\mathtt{Enc}}$ times. This yields the upper bound of the claim. ∎

Game G_3. In G_3, the game aborts if there is a query to the challenge oracle for which the signature verifies, the sender's public keys are honest, and there was no previous signature on the same ring and same message, i.e. if the oracle reaches Line 46.

Claim 6: There exists a PPT adversary $\mathcal{B}$ against the **UF-CRA1** security of RSig, such that

$$\left|\Pr\left[\mathsf{G}_2^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_3^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{RSig},\mathcal{B}}^{(n,2,\mathrm{Q_{Enc}})\text{-}\mathbf{UF\text{-}CRA1}}.$$

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 12. The number of signing queries equals the number of queries to `Encps` and the forgery returned in Line 37 fulfils the winning condition for the unforgeability game of $\mathcal{B}$. The ring is a subring of honest users since we check for honest senders in Line 36 and we do not query the signing oracle on the same combination of ring and message twice due to the abort in Line 13 which was introduced in Game G_2. Moreover, the signature verifies due to the check in Line 34 and the message ring combination was no input of a previous signing query which is check in Line 36.

∎

Game G_4. In the encapsulation oracle `Encps`, the KEM key kk is replaced with a uniformly random value from the KEM key space $\mathcal{K}_{\mathsf{KEM}}$ if the receiver key is honest, i.e. $kpk \in \{kpk_1, \ldots, kpk_n\}$. Further, it is stored alongside the receiver's key and ciphertext in the set $\mathcal{E}_{\mathsf{KEM}}$ and the decapsulation oracle is changed to check for a corresponding element in $\mathcal{E}_{\mathsf{KEM}}$ and the actual KEM key kk is replaced by the one stored in $\mathcal{E}_{\mathsf{KEM}}$ for consistency. In this case, we also set `Flag` to **true**.

Claim 7: There exists a PPT adversary $\mathcal{C}$ against the **IND-CCA** security of KEM, such that

$$\left|\Pr\left[\mathsf{G}_3^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_4^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{KEM},\mathcal{C}}^{(n,\mathrm{Q_{Chl}},\mathrm{Q_{Enc}})\text{-}\mathbf{IND\text{-}CCA}}.$$

Proof. Adversary $\mathcal{C}$ is formally constructed in Figure 13. In the real case, $\mathcal{C}$ is simulating Game G_3, in the random case, they simulate G_4. Adversary $\mathcal{C}$ needs at most $Q_{\mathtt{Chl}}$ queries to the decapsulation oracle and at most $Q_{\mathtt{Enc}}$ queries to the challenge oracle.

∎

$\underline{\mathcal{B}^{\mathsf{Sgn}}(par, spk_1, \ldots, spk_n)}$

01 **for** $i \in [n]$
02 $\quad (ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
03 $\quad sk_i := (ksk_i, \perp)$
04 $\quad pk_i := (kpk_i, spk_i)$
05 $\mathcal{D}, \mathcal{Q}, \mathcal{Q}', \mathcal{E}_{\mathsf{KEM}}, \mathcal{H} \leftarrow \emptyset$
06 $b \xleftarrow{\$} \{0, 1\}$
07 $b' \xleftarrow{\$} \mathcal{A}^{\texttt{Encps},\texttt{Chall}}(pk_1, \ldots, pk_n)$
08 **return** $[\![b = b']\!]$

Oracle $\underline{\texttt{Encps}(s \in [n], pk)}$

09 **parse** $pk \to (kpk, spk)$
10 $(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
11 $m \leftarrow kct||kpk_s||kpk||spk$
12 **if** $(\{spk_s, spk\}, m) \in \mathcal{Q}'$
13 $\quad$ **abort**
14 $\sigma' \leftarrow \mathsf{Sgn}(s, \{spk_s, spk\}, m)$ // signing query
15 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk_s, spk\}, m, \sigma')\}$
16 $\mathcal{Q}' \leftarrow \mathcal{Q}' \cup \{(\{spk_s, spk\}, m)\}$
17 $kk \to kk_1||kk_2$
18 $\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
19 $c := (kct, \sigma)$
20 $k := \mathsf{H}(kk_2, \sigma||spk_s||m)$
21 **if** $kpk \in \{kpk_1, \ldots, kpk_n\}$
22 $\quad \mathcal{H} \leftarrow \mathcal{H} \cup \{(k, kk_2, \sigma, spk_s, m)\}$
23 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
24 **return** (c, k)

Oracle $\underline{\texttt{Chall}(pk, r \in [n], c)}$

25 **if** $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
26 $\quad$ **return** k
27 **parse** $pk \to (kpk, spk)$
28 **parse** $c \to (kct, \sigma)$
29 $m \leftarrow kct||kpk||kpk_r||spk_r$
30 $kk \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
31 $kk \to kk_1||kk_2$
32 $k \leftarrow \mathsf{H}(kk_2, \sigma||spk||m)$
33 $\sigma' \leftarrow \mathsf{SyE.Dec}_{kk_1}(\sigma)$
34 **if** $\mathsf{RSig.Ver}(\sigma', \{spk, spk_r\}, m) \neq 1$
35 $\quad k \leftarrow \perp$
36 **elseif** $pk \in \{pk_1, \ldots, pk_n\} \wedge (\{spk, spk_r\}, m, \cdot) \notin \mathcal{Q}$
37 $\quad$ **return** $(\sigma', \{spk, spk_r\}, m)$ // return forgery
38 **elseif** $pk \in \{pk_1, \ldots, pk_n\}$
39 $\quad \mathcal{H} \leftarrow \mathcal{H} \cup \{(k, kk_2, \sigma, spk, m)\}$
40 $\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk, spk_r\}, m, \sigma')\}$
41 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
42 **if** $b = 0$
43 $\quad$ **continue**
44 **if** $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \perp$
45 $\quad k \xleftarrow{\$} \mathcal{K}$
46 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
47 **return** k

Figure 12. Adversary $\mathcal{B}$ against **UF-CRA1** security of RSig having access to oracle $\texttt{Sgn}$.

Game G_5. Game G_5 aborts if there is a challenge query for which the signature verifies, the sender keys are honest, there already exists a signature on the same ring/message pair in $\mathcal{Q}$, and there already was a hash query on H on the same inputs before, i.e. the game reaches Line 51.

Claim 8:

$$\Pr[\mathsf{G}_4^{\mathsf{A}} \Rightarrow 1] = \Pr[\mathsf{G}_5^{\mathsf{A}} \Rightarrow 1].$$

Proof. We argue that the probability of winning the games is the same by showing that the **abort** in Line 51 can never be reached. Assume that **abort** is reached which means that there is an element of the form $(\cdot, kk_2, \sigma, spk, m)$ in $\mathcal{H}$ where $m = kct||kpk||kpk_r, spk_r$. For each time an element is added to $\mathcal{H}$, an element is added to $\mathcal{D}$. This element is determined by the element of $\mathcal{H}$ (except for the final AKEM key) and has the form

$$((kpk, spk), (kpk_r, spk_r), (kct, \sigma), \cdot).$$

However, if such an element exists in $\mathcal{D}$, the challenge oracle $\texttt{Chall}$ returns in Line 32 and never reaches the **abort** in Line 51. ∎

Game G_6. Game G_6 is modified such that in the $\texttt{Encps}$ oracle the AKEM key k is replaced by a uniformly random value if the receiver is honest (Line 26). It is also replaced in the $\texttt{Chall}$ oracle if the key is not $\perp$ (as in Line 44), the game did not abort, and the sender key is honest (Line 52).

$\underline{\mathcal{C}^{\texttt{Dec},\texttt{Chl}}(kpk_1,\ldots,kpk_n)}$

01 **for** $i \in [n]$
02 $\quad (ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
03 $\quad sk_i := (\perp, ssk_i)$
04 $\quad pk_i := (kpk_i, spk_i)$
05 $\mathcal{D}, \mathcal{Q}, \mathcal{Q}', \mathcal{E}_{\mathsf{KEM}}, \mathcal{H} \leftarrow \emptyset$
06 $b \xleftarrow{\$} \{0,1\}$
07 $b' \xleftarrow{\$} \mathcal{A}^{\texttt{Encps},\texttt{Chall}}(pk_1,\ldots,pk_n)$
08 **return** $[\![b = b']\!]$

Oracle $\underline{\texttt{Encps}(s \in [n], pk)}$

09 **parse** $pk \to (kpk, spk)$
10 $(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
11 **if** $\exists\, i : kpk = kpk_i$
12 $\quad kk \leftarrow \texttt{Chl}(i)$ // challenge query
13 $m \leftarrow kct||kpk_s||kpk||spk$
14 **if** $(\{spk_s, spk\}, m) \in \mathcal{Q}'$
15 $\quad$ **abort**
16 $\sigma' \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk\}, m)$
17 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk_s, spk\}, m, \sigma')\}$
18 $\mathcal{Q}' \leftarrow \mathcal{Q}' \cup \{(\{spk_s, spk\}, m)\}$
19 $kk \to kk_1||kk_2$
20 $\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
21 $c := (kct, \sigma)$
22 $k := \mathsf{H}(kk_2, \sigma||spk_s||m)$
23 **if** $kpk \in \{kpk_1,\ldots,kpk_n\}$
24 $\quad \mathcal{H} \leftarrow \mathcal{H} \cup \{(k, kk_2, \sigma, spk_s, m)\}$
25 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
26 **return** (c, k)

Oracle $\underline{\texttt{Chall}(pk, r \in [n], c)}$

27 **if** $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
28 $\quad$ **return** k
29 **parse** $pk \to (kpk, spk)$
30 **parse** $c \to (kct, \sigma)$
31 $m \leftarrow kct||kpk||kpk_r||spk_r$
32 $kk \leftarrow \texttt{Dec}(r, kct)$ // decapsulation query
33 $kk \to kk_1||kk_2$
34 $k \leftarrow \mathsf{H}(kk_2, \sigma||spk||m)$
35 $\sigma' \leftarrow \mathsf{SyE.Dec}_{kk_1}(\sigma)$
36 **if** $\mathsf{RSig.Ver}(\sigma', \{spk, spk_r\}, m) \neq 1$
37 $\quad k \leftarrow \perp$
38 **elseif** $pk \in \{pk_1,\ldots,pk_n\} \wedge (\{spk, spk_r\}, m, \cdot) \notin \mathcal{Q}$
39 $\quad$ **abort**
40 $\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk, spk_r\}, m, \sigma')\}$
41 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
42 **elseif** $pk \in \{pk_1,\ldots,pk_n\}$
43 $\quad \mathcal{H} \leftarrow \mathcal{H} \cup \{(k, kk_2, \sigma, spk, m)\}$
44 $\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk, spk_r\}, m, \sigma')\}$
45 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
46 **if** $b = 0$
47 $\quad$ **continue**
48 **if** $b = 1 \wedge pk \in \{pk_1,\ldots,pk_n\} \wedge k \neq \perp$
49 $\quad k \xleftarrow{\$} \mathcal{K}$
50 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
51 **return** k

Figure 13. Adversary $\mathcal{C}$ against **IND-CCA** security of KEM having access to oracles `Dec` and `Chl`.

Claim 9: There exists a PPT adversary $\mathcal{D}$ against the **PRF** security of H such that

$$\left|\Pr\left[\mathsf{G}_5^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_6^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{H},\mathcal{D}}^{(\mathsf{Q}_{\mathtt{Enc}}, \mathsf{Q}_{\mathtt{Enc}} + \mathsf{Q}_{\mathtt{Chl}})\text{-}\mathbf{PRF}}.$$

Proof. Adversary $\mathcal{D}$ is formally constructed in Figure 14. Note that the evaluation query in Line 54 on index $\hat{\ell}$ is well defined for the following reason. It is not possible to reach Line 54 with `Flag` $=$ **false**: if the algorithm reaches Line 54, it means that the condition in Line 47 which implies that there must exist an element of the form $(\{spk, spk_r\}, m, \cdot)$ in $\mathcal{Q}$. This means there was a query to `Encps` on the same m which equals $kct||kpk||kpk_r||spk_r$. Hence, the receiver's KEM public key in this particular encapsulation query was kpk_r which is an honest KEM public key. However, if the encapsulation oracle was queried on an honest receiver key, an element is added to set $\mathcal{E}_{\mathsf{KEM}}$ in Line 16 and thus `Flag` must be set to **true** in the current `Chall` query.

Adversary $\mathcal{D}$ simulates Game G_5 in their own real case of the **PRF** game. It remains to show that they actually simulate G_6 in the random case of the **PRF** game. In Game G_6, the AKEM key is always random but the evaluation oracle `Eval` returns the same key for the same PRF key and PRF input. However, in oracle `Encps` a new index is chosen and in oracle `Chall` there was no previous query to the same key and input due to the **abort** in Line 53.

$\underline{\mathcal{D}^{\mathtt{Eval}}}$

$\ell \leftarrow 0$
for $i \in [n]$
  $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
  $sk_i := (ksk_i, ssk_i)$
  $pk_i := (kpk_i, spk_i)$
$\mathcal{D}, \mathcal{Q}, \mathcal{Q}', \mathcal{E}_{\mathsf{KEM}}, \mathcal{H} \leftarrow \emptyset$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\mathtt{Encps},\mathtt{Chall}}(pk_1, \ldots, pk_n)$
return $[\![b = b']\!]$

$\underline{\textbf{Oracle } \mathtt{Encps}(s \in [n], pk)}$

parse $pk \rightarrow (kpk, spk)$
$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
if $kpk \in \{kpk_1, \ldots, kpk_m\}$
  $kk \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$
  $\ell \leftarrow \ell + 1$ // new index
  $\mathcal{E}_{\mathsf{KEM}} \leftarrow \mathcal{E}_{\mathsf{KEM}} \cup \{(kpk, kct, \ell)\}$ // store index
$m \leftarrow kct||kpk_s||kpk||spk$
if $(\{spk_s, spk\}, m) \in \mathcal{Q}'$
  abort
$\sigma' \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk\}, m)$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk_s, spk\}, m, \sigma')\}$
$\mathcal{Q}' \leftarrow \mathcal{Q}' \cup \{(\{spk_s, spk\}, m)\}$
$kk \rightarrow kk_1||kk_2$
$\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
$c := (kct, \sigma)$
$k := \mathsf{H}(kk_2, \sigma||spk_s||m)$
if $kpk \in \{kpk_1, \ldots, kpk_n\}$
  $k \leftarrow \mathtt{Eval}(\ell, \sigma||spk_s||m)$ // evaluation query
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(k, kk_2, \sigma, spk_s, m)\}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

$\underline{\textbf{Oracle } \mathtt{Chall}(pk, r \in [n], c)}$

$\mathtt{Flag} \leftarrow$ **false**
if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
parse $pk \rightarrow (kpk, spk)$
parse $c \rightarrow (kct, \sigma)$
$m \leftarrow kct||kpk||kpk_r||spk_r$
$kk \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
if $\exists\, \ell' : (kpk_r, kct, \ell') \in \mathcal{E}_{\mathsf{KEM}}$ // recover index
  $\hat{\ell} \leftarrow \ell'$
  $\mathtt{Flag} \leftarrow$ **true**
$kk \rightarrow kk_1||kk_2$
$k \leftarrow \mathsf{H}(kk_2, \sigma||spk||m)$
$\sigma' \leftarrow \mathsf{SyE.Dec}_{kk_1}(\sigma)$
if $\mathsf{RSig.Ver}(\sigma', \{spk, spk_r\}, m) \neq 1$
  $k \leftarrow \perp$
elseif $pk \in \{pk_1, \ldots, pk_n\} \wedge (\{spk, spk_r\}, m, \cdot) \notin \mathcal{Q}$
  abort
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk, spk_r\}, m, \sigma')\}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
elseif $pk \in \{pk_1, \ldots, pk_n\}$
  if $\exists\, k' : (k', kk_2, \sigma, spk, m) \in \mathcal{H}$
    abort
  $k \leftarrow \mathtt{Eval}(\hat{\ell}, \sigma||spk||m)$ // evaluation query
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(k, kk_2, \sigma, spk, m)\}$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\{spk, spk_r\}, m, \sigma')\}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
if $b = 0$
  continue
if $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \perp$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
return k

Figure 14. Adversary $\mathcal{D}$ against **PRF** security of H having access to oracle $\mathtt{Eval}$.

■

Eventually, Game G_6 is independent of challenge bit b since in case $b = 0$ the output of the challenge oracle is either $\perp$ or uniformly random for honest sender keys or otherwise the game aborts. However, case $b = 1$ only triggers for keys $k \neq \perp$ and honest sender keys which makes the output indepent of the challenge bit.

$$\Pr[\mathsf{G}_6 \Rightarrow 1] = \frac{1}{2}.$$

Collecting all the terms yields the stated bound. ■

Theorem 5 (RSig **MC-Ano** $\Longrightarrow$ AKEM **DR-Den**)**.** *Let* RSig *be a ring signature which is multi-challenge anonymous under full key exposure, then* $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ *as depicted in Figure 10 is an* ***DR*-Den**

secure authenticated key encapsulation mechanism. In particular, for any ***DR*-Den** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ *there exists a* PPT *simulator* Sim *and a* **MC-Ano** *adversary* $\mathcal{B}$ *against* RSig *such that*

$$\mathrm{Adv}^{(n,\mathrm{Q}_{\mathcal{Chl}})\text{-}\boldsymbol{DR}\text{-}\mathbf{Den}}_{\mathsf{AKEM}[\mathsf{KEM},\mathsf{RSig},\mathsf{SyE},\mathsf{H}],\mathcal{A},\mathsf{Sim}} \leq \mathrm{Adv}^{(n,\mathrm{Q}_{\mathcal{Chl}})\text{-}\mathbf{MC\text{-}Ano}}_{\mathsf{RSig},\mathcal{B}}.$$

The proof of Theorem 5 can be found in Appendix C.

Theorem 6 (KEM **IND-CPA** + SyE **PRP** $\Longrightarrow$ AKEM **HR-Den**). *Let* KEM *be an* **IND-CPA** *secure key encapsulation mechanism and* SyE *a symmetric encryption scheme, then* $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ *as depicted in Figure 10 is a* ***HR*-Den** *secure authenticated key encapsulation mechanism in the honest receiver setting. In particular, for any* ***HR*-Den** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ *there exists a* PPT *simulator* Sim*, a* **IND-CPA** *adversary* $\mathcal{B}$ *against* KEM*, and a* **PRP** *adversary* $\mathcal{C}$ *against* SyE *such that*

$$\mathrm{Adv}^{(n,\mathrm{Q}_{\mathcal{Chl}})\text{-}\boldsymbol{HR}\text{-}\mathbf{Den}}_{\mathsf{AKEM}[\mathsf{KEM},\mathsf{RSig},\mathsf{SyE},\mathsf{H}],\mathcal{A},\mathsf{Sim}} \leq \mathrm{Adv}^{(n,\mathrm{Q}_{\mathcal{Chl}})\text{-}\mathbf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{B}} + \mathrm{Adv}^{(\mathrm{Q}_{\mathcal{Chl}},\mathrm{Q}_{\mathcal{Chl}})\text{-}\mathbf{PRP}}_{\mathsf{SyE},\mathcal{C}}.$$

The proof of Theorem 6 can be found in Appendix C.

5 Instantiations

Table 2. Parameter selection for ring signature scheme Gandalf.

Parameter	Description	Value
λ	security parameter	128
Q_{Sgn}	maximum number of signing queries	2^{64}
N	dimension of $\mathcal{R} \coloneqq \mathbb{Z}[X]/(X^N + 1)$	512
ϵ	Smoothing parameter order	$\frac{1}{\sqrt{Q_{\mathsf{Sgn}} \cdot \lambda}}$
δ_{KL}	maximum KL-divergence of PreSmp	2ϵ
a	Rényi order	2λ
R_a	maximum Rényi divergence of PreSmp	$1 + 2a\epsilon^2$
α	quality of NTRU trapdoor	1.15
q	prime modulus	12289
s	standard deviation of Gaussian sampler	$\frac{1}{\pi} \cdot \sqrt{\frac{\ln(4N(1+1/\epsilon))}{2}} \cdot \alpha \cdot \sqrt{q}$
τ	tailcut rate of signatures	$[1.08, 1.22]$
κ	maximum size of signing ring	≥ 2
$\lvert\rho\rvert = k$	size of signing ring	$[2, \kappa]$
β	maximum norm of signatures	$\tau \cdot s \cdot \sqrt{(\kappa + 1)N}$
$\lvert pk\rvert$	verification key size (bytes)	896
$\lvert\sigma\rvert$	signature size (bytes)	$606 \cdot k + 24$

Signature Instantiation. For Gandalf we instantiate the trapdoor generation algorithm TpdGen using Antrag [ENS+23] and the preimage sampling algorithm PreSmp using the $\text{Mitaka}_{\mathbb{Z}}$ sampler [EFG+22] that avoids floating point arithmetic. This yields our choice for ϵ which we set to $1/\sqrt{Q_{\mathsf{Sgn}} \cdot \lambda}$. The Antrag signature scheme, which combines the trapdoor generation procedure from [ENS+23] and the Gaussian sampler from [EFG+22], requires a 40 byte salt in every signature, which is needed in the hash when verifying a signature. The remainder of a signature consists of a single ring element, with coefficients distributed around 0 according to a discrete Gaussian distribution of standard deviation s. A naïve implementation of the ANTRAG signature scheme would need $40 + \lceil \log_2(q) \rceil \cdot N$ bytes. However, compression techniques, as seen in FALCON and discussed in [ETWY22], offer a substantial reduction in storage requirements. ANTRAG uses such techniques, resulting in signature sizes of 646 bytes, including the non-compressible 40 byte salt. For GANDALF, only a 24 byte salt is required to amplify security (assuming a security parameter of 128 and 2^{64} signing queries). More details can be found in Appendix B.2. Therefore, GANDALF has a total signature size of $606 \cdot k + 24$ bytes. An overview of all relevant parameters can be found in Table 2. The runtime of the signing algorithm for GANDALF is (necessarily) linear in the size of the ring. As a rough estimate, a naïve implementation would be more efficient than the runtime of one FALCON signing per user in the ring, as we only require the preimage sampling to be done once for each signature. The runtime of the verification algorithm is even more efficient as this only involves linear operations and a norm check. To obtain concrete security estimates for GANDALF, consider an adversary's advantage in the unforgeability game Theorem 2. The following Lemma shows that, for our specific choice of $\epsilon = 1/\sqrt{Q_{\mathsf{Sgn}} \cdot \lambda}$, applying the Rényi divergence results in a constant loss of at most 6 bits of security. Therefore, applying Lemma 8 to Theorem 2 yields an overall loss of $c \leq 78$.

Lemma 8 (Bounding Rényi Divergence (adapted from [Pre17, Sec. 3.3])). Assume that $R_a(\mathsf{PreSmp} \mid\mid \mathcal{D}) \lesssim 1 + 2a\epsilon^2$ for all $a \in (1, +\infty]$, all $0 < \epsilon \leq \frac{1}{\sqrt{q \cdot \lambda}}$, and all $q, \lambda \in \mathbb{N}$. Then

$$R_{2\lambda}(\mathsf{PreSmp} \mid\mid \mathcal{D})^q \lesssim 55.$$

Proof. Setting $\epsilon \leq \frac{1}{\sqrt{q \cdot \lambda}}$ and $a = 2\lambda$ gives $R_a(\mathsf{PreSmp} \mid\mid \mathcal{D}) \lesssim 1 + \frac{4}{q}$, which yields

$$R_{2\lambda}(\mathsf{PreSmp} \mid\mid \mathcal{D})^q \lesssim \left(1 + \frac{4}{q}\right)^q \leq e^4.$$

In total we get

$$R_{2\lambda}(\mathsf{PreSmp} \mid\mid \mathcal{D})^q \lesssim e^4 \leq 55,$$

which is a loss of $\log(e^4) \leq 6$ bits in total. ■

In order to estimate the hardness of LWE and SIS, we use the *Lattice-Estimator* tool [APS15b, APS15a] [6]. For our parameter choices, the LWE advantage can safely be ignored, and the term $\frac{c}{|\mathcal{R}_q|}$ is approximately 2^{-6948}. Hence, the SIS advantage dominates the security bound.

The norm bound for GANDALF is $\|(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})\|_2 \leq \beta$, with $\beta = \tau \cdot s\sqrt{(k+1) \cdot N}$. This means that that the security degrades as the ring size k increases, because the SIS instance becomes easier. Conversely, Lemma 7 shows that correctness increases with larger ring sizes. We balance this trade-off by setting the tailcut rate τ based on the required maximal ring size, which may vary depending on the application. A larger τ improves correctness but only marginally reduces security. We aim for a correctness error of at most 2^{-80}. Thus, we choose τ to be the smallest value that meets the correctness goal while maximising security. Our concrete parameter proposals are detailed in Table 3 up to a maximal ring size of 26, the largest value for which the signature size remains smaller than SMILE Figure 1. The security column in Table 3 only shows the SIS advantage. Note that there is an additional 7-bit loss due to the Rényi argument (Theorem 2) and the reduction is non-tight.

DENIABLE AKEM. We instantiate the IND-CCA secure KEM with NTRU-A from [DHK+23]. For concrete parameters see Table 6 in Section 5. Our AKEM construction uses GANDALF with $\kappa = 2$. The resulting scheme has ciphertexts and public keys of size 2004 and 1664 bytes, respectively. Refer to Table 4 for an overview. The computational overhead of the AKEM is not significantly impacted by the KEM NTRU-A, as its operations are linear and its noise sampling form a centred binomial distribution is highly efficient. The efficiency of the resulting AKEM is primarily dominated by the ring signature scheme. To

[6] Commit: `f18533a`

Table 3. Definition of function $\psi(\kappa)$ for $\kappa \in [2, 26]$ and resulting parameters. The last column shows the size of a signature for a ring of maximum size $|\rho| = k = \kappa$. For smaller rings the signature size is correspondingly smaller.

max ring size κ	**tailcut rate** $\tau = \psi(\kappa)$	**correctness error** $-\log_2(\delta(\kappa))$	**norm bound** β	**security** $-\log_2\left(\mathrm{Adv}^{\mathcal{R}\text{-}\mathbf{ISIS}}_{m,q,\alpha,\beta}\right)$	**signature size (in bytes)** $\lvert\sigma\rvert$ for $k = \kappa$
2	1.2	83	6 384	142	1 244
3	1.17	81	7 372	137	1 850
4	1.16	90	8 242	133	2 456
5	1.14	83	9 029	130	3 062
6	1.13	84	9 752	128	3 668
7	1.12	82	10 426	126	4 274
8	1.12	92	11 058	124	4 880
9	1.11	86	11 656	123	5 486
10	1.11	95	12 225	121	6 092
11	1.1	86	12 769	120	6 698
12	1.1	93	13 290	119	7 304
13	1.09	81	13 792	118	7 910
14	1.09	87	14 276	117	8 516
15	1.09	93	14 744	116	9 122
16	1.09	99	15 198	115	9 728
17	1.08	83	15 639	115	10 334
18	1.08	88	16 067	114	10 940
19	1.08	92	16 485	113	11 546
20	1.08	97	16 892	112	12 152
21	1.08	101	17 289	112	12 758
22	1.07	81	17 678	111	13 364
23	1.07	85	18 058	111	13 970
24	1.07	88	18 430	111	14 576
25	1.07	92	18 795	110	15 182
26	1.07	96	19 153	109	15 788

Table 4. Schemes used for instantiating our AKEM construction. Cells marked with "—" indicate that a particular parameter is not applicable to the scheme.

Primitive	**Scheme (variant)**	**Security**	**Assumption**	**Model**	**Size (in bytes)**		
					σ	c	pk
RSig	Gandalf [Figure 5]	**UF**, **Ano**	$\mathcal{R}$-NTRU, $\mathcal{R}$-ISIS	ROM	1 236	—	896
KEM	NTRU-A [DHK+23]	**IND-CCA**	$\mathcal{R}$-NTRU, $\mathcal{R}$-LWE2	ROM/QROM	—	768	768
AKEM	AKEM [Figure 10]	**Ins-Aut**, **Ins-CCA** **HR-Den**, **DR-Den**	$\mathcal{R}$-NTRU, $\mathcal{R}$-ISIS, $\mathcal{R}$-LWE2	Standard	—	2 004	1 664

provide a comprehensive comparison of our AKEM construction with existing ones from the literature, we present an overview in Table 5. The Diffie-Hellman AKEM (DH-AKEM), formalised in [ABH+21], is instantiated with `Curve25519`. The AKEMs from [AJKL23] are black-box constructions from a KEM and a signature and a NIKE, respectively. For a fair comparison, we instantiate construction EtStH using NTRU-A [DHK+23] and Antrag [ENS+23]. The NIKE-AKEM is instantiated with Swoosh [GdKQ+23], a lattice-based NIKE. For the ciphertext size and the public key size of NIKE-AKEM we only present a lower bound since [GdKQ+23] only presents the parameters for their passive secure NIKE (without the size of a NIZK proof). For further details on the security notions of FrodoKEX+, we refer to [CHDN+24].

Acknowledgements. The authors thank the anonymous reviewers for their valuable feedback, Guilherme Rito for helpful discussions on deniability, and Thomas Prest for pointers to related work. Phillip Gajland was

Table 5. Comparison of different AKEMs along with their security notions and whether they are post-quantum secure (PQ). Deniability properties marked with a "$*$" have not been formally proven in the respective work.

Scheme (variant)	Confidentiality	Authenticity	Deniability	PQ	Size (in bytes)	
					c	pk
DH-AKEM (`Curve25519`) [ABH+21]	**Ins-CCA**	**Out-Aut**	**DR-Den***	✗	32	32
EtStH-AKEM (NTRU-A + ANTRAG) [AJKL23]	**Ins-CCA**	**Out-Aut**	—	✓	1 414	1 664
NIKE-AKEM (SWOOSH[7]) [AJKL23]	**Ins-CCA**	**Out-Aut**	**DR-Den***	✓	> 221 184	> 221 184
FrodoKEX+ [CHDN+24]	**IND-1BatchCCA**	**UNF-1KCA**	**DR-Den**	✓	72	21 300
AKEM (NTRU-A + GANDALF) [Figure 10]	**Ins-CCA**	**Out-Aut**	**HR-Den & DR-Den**	✓	2 004	1 664

Table 6. Parameter selection for key encapsulation mechanism KEM, using NTRU-A [DHK+23]. The bit security is the same as KYBER512 [SAB+20]

Parameter	Description	Value
λ	bit security (quantum)	$118 - 140$
δ	decryption error	2^{-197}
q	prime modulus	3329
N	dim of $\mathcal{R} \coloneqq \mathbb{Z}_q[X]/(X^N + 1)$	512
$\lvert pk \rvert$	public key size (bytes)	768
$\lvert c \rvert$	ciphertext size (bytes)	768

supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA - 390781972. Jonas Janneck was supported by the European Union (ERC AdG REWORC - 101054911). Eike Kiltz was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA - 390781972, and by the European Union (ERC AdG REWORC - 101054911).

References

[ABB+13] Carlos Aguilar-Melchor, Slim Bettaieb, Xavier Boyen, Laurent Fousse, and Philippe Gaborit. Adapting Lyubashevsky's signature schemes to the ring signature setting. In Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *AFRICACRYPT 13: 6th International Conference on Cryptology in Africa*, volume 7918 of *Lecture Notes in Computer Science*, pages 1–25, Cairo, Egypt, June 22–24, 2013. Springer, Heidelberg, Germany. 3

[ABH+21] Joël Alwen, Bruno Blanchet, Eduard Hauck, Eike Kiltz, Benjamin Lipp, and Doreen Riepel. Analysing the HPKE standard. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 87–116, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. 3, 5, 17, 28, 29

[ACL+22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 102–132, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. 5

[AJKL23] Joël Alwen, Jonas Janneck, Eike Kiltz, and Benjamin Lipp. The pre-shared key modes of HPKE. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VI*, volume 14443 of *Lecture Notes in Computer Science*, pages 329–360, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany. 3, 5, 17, 28, 29

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 415–432, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany. 3

[APS15a] Martin R. Albrecht, Rachel Player, and Sam Scott. Lattice estimator. https://github.com/malb/lattice-estimator, 2015. Commit: f18533a19433f6fb1d9fb396006f462adc6b8ad3. 27

[APS15b] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. 27

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, December 1993. 8

[BBLW22] Richard Barnes, Karthikeyan Bhargavan, Benjamin Lipp, and Christopher A. Wood. Hybrid Public Key Encryption. RFC 9180, February 2022. 3, 5

[BBR+23] Richard Barnes, Benjamin Beurdouche, Raphael Robert, Jon Millican, Emad Omara, and Katriel Cohn-Gordon. The Messaging Layer Security (MLS) Protocol. RFC 9420, July 2023. 3

[BCG23] David Balbás, Daniel Collins, and Phillip Gajland. WhatsUpp with sender keys? Analysis, improvements and security proofs. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part V*, volume 14442 of *Lecture Notes in Computer Science*, pages 307–341, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany. 6

[BFG+20] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. Towards post-quantum security for Signal's X3DH handshake. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *SAC 2020: 27th Annual International Workshop on Selected Areas in Cryptography*, volume 12804 of *Lecture Notes in Computer Science*, pages 404–430, Halifax, NS, Canada (Virtual Event), October 21-23, 2020. Springer, Heidelberg, Germany. 5, 6

[BFG+22] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. Post-quantum asynchronous deniable key exchange and the Signal handshake. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 3–34, Virtual Event, March 8–11, 2022. Springer, Heidelberg, Germany. 3, 6, 11

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. 3

[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. https://eprint.iacr.org/2010/086. 3, 4

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 60–79, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany. 4, 11, 37

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22(1):114–138, January 2009. 4, 5, 11, 37

[BKP20] Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 464–492, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany. 3, 5

[BLL+15] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 3–24, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany. 8

[BLO18] Carsten Baum, Huang Lin, and Sabine Oechsner. Towards practical lattice-based one-time linkable ring signatures. In David Naccache, Shouhuai Xu, Sihan Qing, Pierangela Samarati, Gregory Blanc, Rongxing Lu, Zonghua Zhang, and Ahmed Meddahi, editors, *ICICS 18: 20th International Conference on Information and Communication Security*, volume 11149 of *Lecture Notes in Computer Science*, pages 303–322, Lille, France, October 29–31, 2018. Springer, Heidelberg, Germany. 3

[BR04] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. https://eprint.iacr.org/2004/331. 7

[BSS02] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. 3

[CDH+20] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. NTRU. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. 6

[CHDN+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. Cryptology ePrint Archive, Paper 2024/120, 2024. https://eprint.iacr.org/2024/120. 5, 6, 28, 29

[CHMR23] Suvradip Chakraborty, Dennis Hofheinz, Ueli Maurer, and Guilherme Rito. Deniable authentication when signing keys leak. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 69–100, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. 18

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265, Brighton, UK, April 8–11, 1991. Springer, Heidelberg, Germany. 3

[DG05] Mario Di Raimondo and Rosario Gennaro. New approaches for deniable authentication. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 2005: 12th Conference on Computer and Communications Security*, pages 112–121, Alexandria, Virginia, USA, November 7–11, 2005. ACM Press. 6

[DGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 400–409, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. 6

[DHK+23] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, Gregor Seiler, and Dominique Unruh. A thorough treatment of highly-efficient NTRU instantiations. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 65–94, Atlanta, GA, USA, May 7–10, 2023. Springer, Heidelberg, Germany. 5, 27, 28, 29

[DHM+20] Ivan Damgård, Helene Haagh, Rebekah Mercer, Anca Nitulescu, Claudio Orlandi, and Sophia Yakoubov. Stronger security and constructions of multi-designated verifier signatures. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 229–260, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany. 18

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. 3

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 22–41, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany. 3, 7, 8

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *30th Annual ACM Symposium on Theory of Computing*, pages 409–418, Dallas, TX, USA, May 23–26, 1998. ACM Press. 6

[DNS04] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, nov 2004. 6

[DZ10] Alexander W. Dent and Yuliang Zheng, editors. *Practical Signcryption*. Springer Berlin Heidelberg, 2010. 3, 5, 17

[EFG+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 Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 222–253, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany. 26

[ENS+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 Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VII*, volume 14444 of *Lecture Notes in Computer Science*, pages 3–36, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany. 4, 5, 26, 28

[ESS+19] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 67–88, Bogota, Colombia, June 5–7, 2019. Springer, Heidelberg, Germany. 3, 5

[ETWY22] Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Shorter hash-and-sign lattice-based signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 245–275, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. 27

[FKP17] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the one-per-message unforgeability of (EC)DSA and its variants. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 519–534, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. 4

[FM15] Marc Fischlin and Sogol Mazaheri. Notions of deniable message authentication. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, WPES '15, page 55–64, New York, NY, USA, 2015. Association for Computing Machinery. 6

[FR23] Thibauld Feneuil and Matthieu Rivain. Threshold computation in the head: Improved framework for post-quantum signatures and zero-knowledge arguments. Cryptology ePrint Archive, Paper 2023/1573, 2023. https://eprint.iacr.org/2023/1573. 5

[GdKQ+23] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. Swoosh: Practical lattice-based non-interactive key exchange. Cryptology ePrint Archive, Report 2023/271, 2023. https://eprint.iacr.org/2023/271. 28

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press. 3, 7, 8

[HKKP22] 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. *Journal of Cryptology*, 35(3):17, July 2022. 6

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, Heidelberg, Germany, June 1998. 3, 8

[LAZ19] Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 110–130, Bogota, Colombia, June 5–7, 2019. Springer, Heidelberg, Germany. 3, 4, 5

[LLNW16] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 1–31, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 3

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact Knapsacks are collision resistant. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 144–155, Venice, Italy, July 10–14, 2006. Springer, Heidelberg, Germany. 10

[LNS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. SMILE: Set membership from ideal lattices with applications to ring signatures and confidential transactions. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 611–640, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 3, 5

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 8

[MP16a] Moxie Marlinspike and Trevor Perrin. The double ratchet algorithm, 2016. 6

[MP16b] Moxie Marlinspike and Trevor Perrin. The x3dh key agreement protocol, 2016. 3, 6

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007. 8

[MW17] Daniele Micciancio and Michael Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 455–485, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 9

[Nao02] Moni Naor. Deniable ring authentication. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 481–498, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. 3, 6

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`. 3

[Pre15] Thomas Prest. *Gaussian sampling in lattice-based cryptography.* PhD thesis, Ecole normale supérieure-ENS PARIS, 2015. 8

[Pre17] Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 347–374, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. 8, 9, 15, 27

[ROSW23] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-16, Internet Engineering Task Force, April 2023. Work in Progress. 3

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany. 3, 5, 10

[SAB+20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. 29

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. 6

[SM04] Willy Susilo and Yi Mu. Non-interactive deniable ring authentication. In Jong In Lim and Dong Hoon Lee, editors, *ICISC 03: 6th International Conference on Information Security and Cryptology*, volume 2971 of *Lecture Notes in Computer Science*, pages 386–401, Seoul, Korea, November 27–28, 2004. Springer, Heidelberg, Germany. 6

[SSW20] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1461–1480, Virtual Event, USA, November 9–13, 2020. ACM Press. 6

[UG15] Nik Unger and Ian Goldberg. Deniable key exchanges for secure messaging. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1211–1223, Denver, CO, USA, October 12–16, 2015. ACM Press. 6

[UG18] Nik Unger and Ian Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *Proceedings on Privacy Enhancing Technologies*, 2018(1):21–66, January 2018. 6

[Wha20] WhatsApp. WhatsApp Encryption Overview Technical white paper, v.3, oct 2020. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf. 6

[YEL+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 Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 251–281, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 3, 4, 5

[Zhe97] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany. 3

[ZK02] Fangguo Zhang and Kwangjo Kim. ID-based blind signature and ring signature from pairings. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany. 3

A Appendix for Section 2 (Preliminaries)

A.1 Pseudorandom Function

Definition 12 (Pseudorandom Function). A keyed function F with a finite key space $\mathcal{K}$, and finite output range $\mathcal{R}$ is a function $F : \mathcal{K} \times \{0,1\}^* \to \mathcal{R}$. We formalise the notion of *pseudorandomess* for a keyed function F via the game $(n, Q_{\texttt{Eval}})$-**PRF** depicted in Figure 15 and define the advantage of adversary $\mathcal{A}$ as

$$\mathrm{Adv}_{F,\mathcal{A}}^{(\mathrm{n},\mathrm{Q}_{\texttt{Eval}})\text{-}\mathbf{PRF}} := \left| \Pr\left[(n, Q_{\texttt{Eval}})\text{-}\mathbf{PRF}_F(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right|.$$

Game $(n, Q_{\texttt{Eval}})$-$\mathbf{PRF}_F(\mathcal{A})$	**Oracle** $\texttt{Eval}(i \in [n], x)$
for $i \in [n]$	**if** $b = 0$
$\quad k_i \xleftarrow{\$} \mathcal{K}$	$\quad$ **return** $F(k_i, x)$
$\quad f_i \xleftarrow{\$} \{f \mid f : \{0,1\}^* \to \mathcal{R}\}$	**if** $b = 1$
$b \xleftarrow{\$} \{0,1\}$	$\quad$ **return** $f_i(x)$
$b' \leftarrow \mathcal{A}^{\texttt{Eval}}$	
return $[\![b = b']\!]$	

Figure 15. Game defining **PRF** for a keyed function F with adversary $\mathcal{A}$ making at most $Q_{\texttt{Eval}}$ queries to $\texttt{Eval}$.

A.2 Key Encapsulation Mechanism

Definition 13 (Key Encapsulation Mechanism). A *key encapsulation mechanism* KEM is defined as a tuple $\mathsf{KEM} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of the following PPT algorithms.

$(sk, pk) \xleftarrow{\$} \mathsf{Gen}$**:** The probabilistic key generation algorithm Gen returns a key pair (sk, pk) implicitly defining a shared key space $\mathcal{K}$.

$(c, k) \xleftarrow{\$} \mathsf{Enc}(pk)$**:** The probabilistic encapsulation algorithm Enc takes as input a public key and returns a ciphertext c and a shared key $k \in \mathcal{K}$.

$k \leftarrow \mathsf{Dec}(sk, c)$**:** The deterministic decapsulation algorithm Dec takes as input a secret key sk and a ciphertext c and returns a shared key $k \in \mathcal{K}$ or a failure symbol $\perp$.

The correctness error δ is defined as

$$\delta := \Pr\left[\mathsf{Dec}(sk, c) \neq k \,\middle|\, \begin{array}{r} (sk, pk) \xleftarrow{\$} \mathsf{Gen} \\ (c, k) \xleftarrow{\$} \mathsf{Enc}(pk) \end{array}\right].$$

We also assume (without loss of generality) the existence of an efficiently computable function μ such that for all $(sk, pk) \in \mathsf{Gen}$ it holds $\mu(sk) = pk$.

The γ-spreadness of a KEM is defined as

$$\gamma_{\mathsf{KEM}} := \max_{\substack{(sk,pk) \in \mathsf{Gen} \\ c \in \mathcal{C}}} \Pr\left[\mathsf{Enc}(pk) = (c, \cdot)\right].$$

We formalise the notion of ciphertext indistinguishability for a key encapsulation mechanism KEM via the game $(n, Q_{\texttt{Dec}}, Q_{\texttt{Chl}})$-$\mathbf{IND\text{-}CCA}_{\mathsf{KEM}}(\mathcal{A})$ depicted in Figure 16 and define the advantage of adversary $\mathcal{A}$ as

$$\mathrm{Adv}_{\mathsf{KEM},\mathcal{A}}^{(\mathrm{n},\mathrm{Q}_{\texttt{Dec}},\mathrm{Q}_{\texttt{Chl}})\text{-}\mathbf{IND\text{-}CCA}} := \left| \Pr\left[(n, Q_{\texttt{Dec}}, Q_{\texttt{Chl}})\text{-}\mathbf{IND\text{-}CCA}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right|.$$

Game $(n, Q_{\texttt{Dec}}, Q_{\texttt{Chl}})$-**IND-CCA**$_{\mathsf{KEM}}(\mathcal{A})$	**Oracle** $\texttt{Dec}(r \in [n], c)$	**Oracle** $\texttt{Chl}(r \in [n])$
for $i \in [n]$	**if** $\exists\, k : (pk_r, c, k) \in \mathcal{D}$	$(c, k) \xleftarrow{\$} \mathsf{Enc}(pk_r)$
$\quad (sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$	$\quad$ **return** k	**if** $b = 0$
$b \xleftarrow{\$} \{0, 1\}$	$k \leftarrow \mathsf{Dec}(sk_r, k)$	$\quad$ **continue**
$b' \leftarrow \mathcal{A}^{\texttt{Dec},\texttt{Chall}}(pk_1, \ldots, pk_n)$	**return** k	**if** $b = 1$
return $[\![b = b']\!]$		$\quad k \xleftarrow{\$} \mathcal{K}$
		$\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_r, c, k)\}$
		return (c, k)

Figure 16. Game defining **IND-CCA** for a key encapsulation mechanism KEM with adversary $\mathcal{A}$ making at most $Q_{\texttt{Dec}}$ queries to $\texttt{Dec}$ and at most $Q_{\texttt{Chl}}$ queries to $\texttt{Chl}$.

We define **IND-CPA** security with corruptions of a KEM via the game $(n, Q_{\texttt{Chl}})$-**IND-CPA**$_{\mathsf{KEM}}(\mathcal{A})$ depicted in Figure 17 and define the advantage of adversary $\mathcal{A}$ as

$$\mathrm{Adv}_{\mathsf{KEM},\mathcal{A}}^{(\mathrm{n},\mathrm{Q}_{\texttt{Chl}})\text{-}\textbf{IND-CPA}} := \left| \Pr\left[(n, Q_{\texttt{Chl}})\text{-}\textbf{IND-CPA}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right|.$$

Game $(n, Q_{\texttt{Chl}})$-**IND-CPA**$_{\mathsf{KEM}}(\mathcal{A})$	**Oracle** $\texttt{Rev}(i \in [n])$	**Oracle** $\texttt{Chl}(r \in [n])$
for $i \in [n]$	$\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$	$\mathcal{C} \leftarrow \mathcal{C} \cup \{r\}$
$\quad (sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$	**return** sk_i	$(c, k) \xleftarrow{\$} \mathsf{Enc}(pk_r)$
$b \xleftarrow{\$} \{0, 1\}$		**if** $b = 0$
$b' \leftarrow \mathcal{A}^{\texttt{Chall},\texttt{Rev}}(pk_1, \ldots, pk_n)$		$\quad$ **continue**
return $[\![b = b' \wedge \mathcal{R} \cap \mathcal{C} = \emptyset]\!]$		**if** $b = 1$
		$\quad k \xleftarrow{\$} \mathcal{K}$
		return (c, k)

Figure 17. Game defining **IND-CPA** for a key encapsulation mechanism KEM with adversary $\mathcal{A}$ making at most $Q_{\texttt{Chl}}$ queries to $\texttt{Chl}$.

A.3 Symmetric Encryption

We recall the syntax and security of a symmetric encryption scheme.

Definition 14 (Symmetric Encryption). A *symmetric encryption* SyE is defined as a tuple $\mathsf{SyE} := (\mathsf{Enc}, \mathsf{Dec})$ of the following PPT algorithms.

$c \leftarrow \mathsf{Enc}_k(m)$**:** The determinsitic encryption algorithm Enc parametrized by a symmetric key k takes as input a message m and outputs a ciphertext c.

$m \leftarrow \mathsf{Dec}_k(c)$**:** The deterministic decryption algorithm Dec parametrized by a symmetric key k takes as input a ciphertext c and outputs a message m.

We define security in the sense of a pseudo random permutation via the advantage of adversary $\mathcal{A}$ having access to an oracle $\texttt{Eval}$. The advantage for adversary $\mathcal{A}$ issuing at most Q queries to the evaluation oracle is defined as

$$\mathrm{Adv}_{\mathsf{SyE},\mathcal{A}}^{(n,\mathrm{Q})\text{-}\textbf{PRP}} := \left| \Pr[b \leftarrow \mathcal{A}^{\texttt{Eval}_0(i \in [n], \cdot)}] - \Pr[b \leftarrow \mathcal{A}^{\texttt{Eval}_1(i \in [n], \cdot)}] \right|,$$

where $\texttt{Eval}_0(i, m)$ returns $\mathsf{Enc}_{k_i}(m)$ for randomly chosen secret keys $k_i \xleftarrow{\$} \mathcal{K}$, and $\textbf{PRP}_1(i, m)$ returns $\pi_i(m)$ for randomly chosen permutations $\pi_i, i \in [n]$.

B Appendix for Section 3 (Ring Signatures)

B.1 Counter Example

The notions from [BKM06] and [BKM09] are repeated in Figure 18. W.l.o.g. we ignore the Stp algorithm here since these notions do not use a setup.

Game (n, Q_{Sgn})-$\mathbf{Ano}_{\mathsf{RSig}}(\mathcal{A})$ [BKM06]

$\mathbf{for}\ i \in [n]$
$\quad (sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
$(m^\star, \rho^\star, i_0, i_1) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Sgn}}(pk_1, \ldots, pk_n)$
$b \xleftarrow{\$} \{0,1\}$
$\sigma^\star \xleftarrow{\$} \mathsf{Sgn}(sk_{i_b}, \rho^\star, m^\star)$
$b' \xleftarrow{\$} \mathcal{A}_2^{\mathsf{Sgn}}(\sigma^\star, sk_1, \ldots, sk_n)$
$\mathbf{return}\ [\![b = b' \wedge pk_{i_0} \in \rho^\star \wedge pk_{i_1} \in \rho^\star]\!]$

Game $(n, Q_{\mathtt{Chl}})$-$\mathbf{Ano}_{\mathsf{RSig}}(\mathcal{A})$ [BKM09]

$\mathbf{for}\ i \in [n]$
$\quad (sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\mathtt{Chl}}((sk_1, pk_1), \ldots, (sk_n, pk_n))$
$\mathbf{return}\ [\![b = b']\!]$

Oracle $\mathtt{Sgn}(i \in [n], \rho, m)$

$\mathbf{if}\ pk_i \in \rho$
$\quad \sigma \xleftarrow{\$} \mathsf{Sgn}(sk_i, \rho, m)$
$\quad \mathbf{return}\ \sigma$
$\mathbf{else}$
$\quad \mathbf{return}\ \perp$

Oracle $\mathtt{Chl}(i_0 \in [n], i_1 \in [n], \rho, m)$

$\mathbf{if}\ pk_{i_0} \in \rho \wedge pk_{i_1} \in \rho$
$\quad \sigma \xleftarrow{\$} \mathsf{Sgn}(sk_{i_b}, \rho, m)$
$\quad \mathbf{return}\ \sigma$
$\mathbf{else}$
$\quad \mathbf{return}\ \perp$

Figure 18. Games defining **Ano** in [BKM06] and [BKM09].

For both, we have advantage

$$\mathrm{Adv}_{\mathsf{RSig},\mathcal{A}}^{(\mathrm{n},\cdot)\text{-}\mathbf{Ano}} := \left| \Pr[(n,\cdot)\text{-}\mathbf{Ano}_{\mathsf{RSig}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right|.$$

Claim 10: The notion from [BKM06] does not imply the notion from [BKM09].

Proof. Let $\mathsf{RSig} := (\mathsf{Gen}, \mathsf{Sgn}, \mathsf{Ver})$ be an unforgeable ring signature scheme secure under full key exposure anonymity [BKM06]. We construct another ring signature scheme $\mathsf{RSig}' := (\mathsf{Gen}, \mathsf{Sgn}', \mathsf{Ver})$ such that $\mathsf{Sgn}'(sk, \rho, m)$ outputs $\perp$ if queried on $m = sk$ and $\mathsf{Sgn}(sk, \rho, m)$ otherwise. Sgn' is also secure under the old anonymity notion, and we show this by constructing an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against RSig using adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against RSig' depicted in Figure 19. If the underlying ring signature is unforgeable, the probability of correctly guessing a secret key should be negligible only giving the public keys and signatures. Thus, the abort conditions trigger with only negligible probability and the ring signature scheme RSig' is also anonymous under full key exposure.

However, it is evident that RSig' is not secure under the new notion. This is because the adversary obtains the secret keys in advance and can query the challenge on $m = sk_{i_0}$ and check if the result is $\perp$ or not directly winning the game with probability 1. This shows that the notion of [BKM09] is not implied by the old notion of [BKM06]. ■

Theorem 1 (Gandalf MC-Ano). *For any adversary $\mathcal{A}$, making at most $Q_{\mathtt{Chl}}$ challenge queries, against the* **MC-Ano** *security of* Gandalf*, depicted in Figure 5, it holds*

$$\mathrm{Adv}_{\textsc{Gandalf},\mathcal{A}}^{(n,\kappa,Q_{\mathtt{Chl}})\text{-}\mathbf{MC\text{-}Ano}} \le Q_{\mathtt{Chl}} \cdot \delta_{KL}.$$

Proof. Consider the sequence of games depicted in Figure 20.

$\mathcal{B}_1^{\mathtt{Sgn}_\mathcal{B}}(pk_1, \ldots, pk_n)$

$(m^\star, \rho^\star, i_0, i_1) \xleftarrow{\$} \mathcal{A}_1^{\mathtt{Sgn}_1}(pk_1, \ldots, pk_n)$
if $\exists\, pk_i : \mu(m^\star) = pk_i$
  abort
return $(m^\star, \rho^\star, i_0, i_1)$

Oracle $\mathtt{Sgn}_1(i \in [n], \rho, m)$

if $\exists\, pk_i : \mu(m) = pk_i$
  abort
if $pk_i \in \rho$
  $\sigma \xleftarrow{\$} \mathsf{Sgn}(sk_i, \rho, m)$
  return σ
else
  return $\perp$

$\mathcal{B}_2^{\mathtt{Sgn}_\mathcal{B}}(\sigma^\star, sk_1, \ldots, sk_n)$

$b' \xleftarrow{\$} \mathcal{A}_2^{\mathtt{Sgn}_2}(\sigma^\star, sk_1, \ldots, sk_n)$
return b'

Oracle $\mathtt{Sgn}_2(i \in [n], \rho, m)$

if $pk_i \in \rho$
  $\sigma \xleftarrow{\$} \mathsf{Sgn}(sk_i, \rho, m)$
  return σ
else
  return $\perp$

Figure 19. Adversary $\mathcal{B}$ against RSig anonymity using an adversary $\mathcal{A}$ against RSig' anonymity.

Game G_0*.* This is the multi-challenge anonymity with full key exposure game for RSig so by definition

$$\Pr[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1] = \mathrm{Adv}_{\textsc{Gandalf},\mathcal{A}}^{(n,\kappa,Q_{\mathtt{Chl}})\text{-}\mathbf{MC}\text{-}\mathbf{Ano}}.$$

$\mathsf{G}_0 - \mathsf{G}_1$

$par \xleftarrow{\$} \mathsf{Stp}(\kappa)$
for $i \in [n]$
  $(\boldsymbol{f}_i, \boldsymbol{g}_i, \boldsymbol{h}_i) \xleftarrow{\$} \mathsf{TpdGen}$
  $sk_i := (\boldsymbol{f}_i, \boldsymbol{g}_i)$
  $pk_i := \boldsymbol{h}_i$
$b \xleftarrow{\$} \{0,1\}$
$b'' \xleftarrow{\$} \{0,1\}$ // G_1
$b' \xleftarrow{\$} \mathcal{A}^{\mathtt{Chl}}(par, (sk_1, pk_1), \ldots, (sk_n, pk_n))$
return $[\![b = b']\!]$

Oracle $\mathtt{Chl}(i_0 \in [n], i_1 \in [n], \rho, m)$

if $\rho \subseteq \{pk_1, \ldots, pk_n\} \wedge pk_{i_0} \in \rho \wedge pk_{i_1} \in \rho$
  $\sigma \xleftarrow{\$} \mathsf{Sgn}(sk_{i_b}, \rho, m)$
  $\sigma \xleftarrow{\$} \mathsf{Sgn}(sk_{i_{b''}}, \rho, m)$ // G_1
  return σ
else
  return $\perp$

Figure 20. Games $\mathsf{G}_0 - \mathsf{G}_1$ for the proof of Theorem 1.

Game G_1*.* In this game, the signatures of the challenge oracle are constructed using the secret key of user $i_{b''}$ instead of user i_b where $b'' \xleftarrow{\$} \{0,1\}$ is a random bit chosen independently of b.

Claim 11:

$$\left|\Pr\left[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1\right]\right| \leq Q_{\mathtt{Chl}} \cdot \delta_{KL}.$$

Proof. To prove the claim, we distinguish two cases. First, if $b'' = b$, the output distributions is exactly the same and the change cannot be distinguished. This occurs with probability $\frac{1}{2}$. In the other case, we compare the distribution of the output of the signing oracle in case of two different senders. Let the ring be $\rho = \{\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_k\}$ and consider the case of $\boldsymbol{h}_{i_0}, \boldsymbol{h}_{i_1} \in \rho$ (otherwise the output is $\perp$). W.l.o.g assume that $\boldsymbol{h}'_1 = \boldsymbol{h}_{i_0}$ and $\boldsymbol{h}'_2 = \boldsymbol{h}_{i_1}$.

The view of adversary $\mathcal{A}$ consists of $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ (the output of the signing oracle), as well as the output of the hash function satisfying

$$\boldsymbol{v} := \mathsf{H}(m, \rho) - \sum_{i \in [k]} \boldsymbol{h}'_i \boldsymbol{u}_i$$

as well as

$$\|(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v})\|_2 \leq \beta \qquad \text{with probability} \quad \delta,$$

for a δ-correct ring signature scheme.

Case $b'' = 0$: If user 1 is the signer, $\boldsymbol{u}_i \sim \mathcal{D}_{\mathbb{Z}^N,s,\mathbf{0}}$ for $2 \leq i \leq k$ and $(\boldsymbol{u}_1, \boldsymbol{v}) \xleftarrow{\$} \mathsf{PreSmp}(\cdot,\cdot,\mathsf{H}(m,\rho) - \sum_{i\in[k]\setminus\{1\}} \boldsymbol{h}'_i \boldsymbol{u}_i)$ by construction. Next, we use the property of the preimage sampler that the output is close to values sampled from $\mathcal{D}_{\mathbb{Z}^{2N},s,\mathbf{0}}$ conditioned on $\boldsymbol{v} = \mathsf{H}(m,\rho) - \sum_{i\in[k]} \boldsymbol{h}'_i \boldsymbol{u}_i$:

$$(\boldsymbol{u}_1, \boldsymbol{v}) \sim \mathcal{D}_{\mathbb{Z}^{2N},s,\mathbf{0}} \mid \boldsymbol{v} = \mathsf{H}(m,\rho) - \sum_{i\in[k]} \boldsymbol{h}'_i \boldsymbol{u}_i.$$

To obtain a concrete bound, we apply Corollary 1 for an upper bound on the KL divergence δ_{KL} between the output of the sampler and the conditional Gaussian. For $Q_{\mathtt{Chl}}$ queries, this yields $Q_{\mathtt{Chl}} \cdot \delta_{KL}$.

Case $b'' = 0$: If user 2 is the signer, we apply the same procedure as before and obtain $\boldsymbol{u}_i \sim \mathcal{D}_{\mathbb{Z}^N,s,\mathbf{0}}$ for $i = 1$ and $3 \leq i \leq k$ as well as

$$(\boldsymbol{u}_2, \boldsymbol{v}) \sim \mathcal{D}_{\mathbb{Z}^{2N},s,\mathbf{0}} \mid \boldsymbol{v} = \mathsf{H}(m,\rho) - \sum_{i\in[k]} \boldsymbol{h}'_i \boldsymbol{u}_i.$$

Again, we obtain the bound $Q_{\mathtt{Chl}} \cdot \delta_{KL}$.

If the hash value $\mathsf{H}(m,\rho)$ was not known to $\mathcal{A}$, the KL divergence of the joint distributions of both cases from

$$(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v}) \sim \mathcal{D}_{\mathbb{Z}^{(k+1)N},s,\mathbf{0}}$$

is close. However, the knowledge of $\mathsf{H}(m,\rho)$ does not help in distinguishing since in both cases it holds

$$\mathsf{H}(m,\rho) = \sum_{i\in[k]} \boldsymbol{h}'_i \boldsymbol{u}_i + \boldsymbol{v}.$$

Further, the norm bound is at most β with the same probability δ since the values are sampled according to a Gaussian and with the tailcut lemma we can use the same results as in Lemma 7.

We recall that the changes can only be distinguished if $b \neq b''$ yielding an overall bound of

$$\frac{1}{2} \cdot 2 \cdot Q_{\mathtt{Chl}} \cdot \delta_{KL}.$$

∎

Note that G_1 is independent of challenge bit b. hence, we obtain the stated bound.

∎

B.2 Enhancing security.

To boost one-per-message unforgeability to full unforgeability, i.e. allowing for arbitrary singing queries, we present a generic compiler which introduces only a small constant overhead. The compiler transforms a **UF-CRA1** ring signatures scheme $\mathsf{RSig} := (\mathsf{Stp}, \mathsf{Gen}, \mathsf{Sgn}, \mathsf{Ver})$ into a **UF-CRA** ring signature $\mathsf{RSig}'[\mathsf{RSig}] := (\mathsf{Stp}, \mathsf{Gen}, \mathsf{Sgn}', \mathsf{Ver}')$ and is depicted in Figure 21. The drawback of the compiler is that the size of the signature increases by ν bits. However, this constant term is quite small compared to the signature.

Theorem 7. *Let* RSig *be a* **UF-CRA1** *secure ring signature, then* $\mathsf{RSig}'[\mathsf{RSig}]$ *as depicted in Figure 21 is a* **UF-CRA** *secure ring signature. In particular, for any* **UF-CRA** *adversary* $\mathcal{A}$ *against* $\mathsf{RSig}'[\mathsf{RSig}]$ *there exists a* **UF-CRA1** *adversary* $\mathcal{B}$ *against* RSig *such that*

$$\mathrm{Adv}_{\mathsf{RSig}',\mathcal{A}}^{(n,\kappa,Q_{\mathit{Sgn}})\text{-}\mathbf{UF\text{-}CRA}} \leq \mathrm{Adv}_{\mathsf{RSig},\mathcal{B}}^{(n,\kappa,Q_{\mathit{Sgn}})\text{-}\mathbf{UF\text{-}CRA1}} + \frac{Q_{\mathit{Sgn}}(Q_{\mathit{Sgn}} - 1)}{2^{\nu+1}}.$$

Proof. We define two games in Figure 22.

$\underline{\mathsf{Sgn}'(sk, \rho, m)}$

$r \xleftarrow{\$} \{0,1\}^{\nu}$
$\sigma \xleftarrow{\$} \mathsf{RSig.Sgn}(sk, \rho, m||r)$
$\sigma' \leftarrow (\sigma, r)$
return σ'

$\underline{\mathsf{Ver}'(\sigma', \rho, m)}$

parse $\sigma' \rightarrow (\sigma, r)$
return $\mathsf{RSig.Ver}(\sigma, \rho, m||r)$

Figure 21. Generic Compiler $\mathsf{RSig}'[\mathsf{RSig}] := (\mathsf{Stp}, \mathsf{RSig}, \mathsf{Sgn}', \mathsf{Ver}')$.

$\underline{\mathsf{G}_0 - \mathsf{G}_1}$

01 $\mathcal{Q}, \mathcal{R} \leftarrow \emptyset$
02 $par \xleftarrow{\$} \mathsf{Stp}(\kappa)$
03 **for** $i \in [n]$
04 $\quad (sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
05 $(\sigma^\star, \rho^\star, m^\star) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sgn}}(par, pk_1, \ldots, pk_n)$
06 **return** $[\![\rho^\star \subseteq \{pk_1, \ldots, pk_n\} \wedge \mathsf{Ver}'(\sigma^\star, \rho^\star, m^\star) = 1 \wedge (\rho^\star, m^\star, \sigma^\star) \notin \mathcal{Q}]\!]$

Oracle $\underline{\mathtt{Sgn}(i \in [n], \rho, m)}$

07 $r \xleftarrow{\$} \{0,1\}^{\nu}$
08 **if** $r \in \mathcal{R}$ // G_1
09 $\quad$ **abort** // G_1
10 $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$ // G_1
11 $\sigma \xleftarrow{\$} \mathsf{Sgn}(sk_i, \rho, m||r)$
12 $\sigma' \leftarrow (\sigma, r)$
13 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\rho, m, \sigma')\}$
14 **return** σ'

Figure 22. Games $\mathsf{G}_0 - \mathsf{G}_1$ for the proof of Theorem 7

Game G_0. This is the **UF-CRA** game for RSig' so by definition

$$\Pr[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1] = \mathrm{Adv}_{\mathsf{RSig}',\mathcal{A}}^{(n,\kappa,\mathrm{Q}_{\mathsf{Sgn}})\textbf{-UF-CRA}}.$$

Game G_1. In Game G_1, the signing oracle is changed by storing the randomness which is chosen to sign together with the original message. Further, the game aborts if the same randomness is used twice.

Claim 12:

$$\left|\Pr\left[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \frac{Q_{\mathsf{Sgn}}(Q_{\mathsf{Sgn}} - 1)}{2^{\nu+1}}.$$

Proof. The randomness is chosen uniformly random and independent for each query to the signing oracle from a set of size $|2^{\nu}|$. Hence, the claim follows directly by applying the birthday bound. ∎

Reduction to G_1. We can now make a reduction from **UF-CRA1** security of the underlying ring signature scheme RSig to Game G_1, i.e. there exists an adversary $\mathcal{B}$ such that

Claim 13:

$$\Pr[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1] \leq \mathrm{Adv}_{\mathsf{RSig},\mathcal{B}}^{(n,\kappa,\mathrm{Q}_{\mathsf{Sgn}})\textbf{-UF-CRA1}}.$$

Proof. Adversary $\mathcal{B}$ against **UF-CRA1** security of RSig simulating the **UF-CRA** game for an adversary $\mathcal{A}$ against RSig is formally constructed in Figure 23. Due to the abort from Game G_1, the queried messages to $\mathtt{Sgn}$ in Line 12 must be unique such that adversary $\mathcal{B}$ can simulate the signing oracle $\mathtt{Sgn}'$ properly. If $\mathcal{A}$ returns a valid forgery, the forgery $\mathcal{B}$ returns must also be valid: by construction of the scheme, it verifies iff $\mathcal{A}$ forgery verifies, the ring $\rho^\star$ is the same and thus a subset of the challenge public keys, and the output triple cannot be in the bookkeeping set of $\mathcal{B}$'s game because in this case it was also in $\mathcal{A}$'s by construction of the ring signature scheme. ∎

Combining the two losses, we obtain the stated bound. ∎

$\mathcal{B}^{\mathtt{Sgn}}(pk_1,\dots,pk_n)$	**Oracle** $\mathtt{Sgn}'(i\in[n],\rho,m)$
$\mathcal{Q},\mathcal{R}\leftarrow\emptyset$	$r \xleftarrow{\$} \{0,1\}^{\nu}$
$par \xleftarrow{\$} \mathsf{Stp}(\kappa)$	**if** $r\in\mathcal{R}$
$(\sigma^\star,\rho^\star,m^\star) \xleftarrow{\$} \mathcal{A}^{\mathtt{Sgn}'}(par,pk_1,\dots,pk_n)$	**abort**
if $\mathsf{Ver}'(\sigma^\star,\rho^\star,m^\star)=1\wedge\rho^\star\subseteq\{pk_1,\dots,pk_n\}\wedge(\rho^\star,m^\star,\sigma^\star)\notin\mathcal{Q}$	$\mathcal{R}\leftarrow\mathcal{R}\cup\{r\}$
$\sigma^\star\to(\sigma,r)$	$\sigma \xleftarrow{\$} \mathtt{Sgn}(i,\rho,m\|\|r)$ // unique message
return $(\sigma,\rho^\star,m^\star\|\|r)$ // return forgery	$\sigma'\leftarrow(\sigma,r)$
return $\perp$	$\mathcal{Q}\leftarrow\mathcal{Q}\cup\{(\rho,m,\sigma')\}$
	return σ'

Figure 23. Adversary $\mathcal{B}$ against **UF-CRA1** security of RSig having access to oracle Sgn simulating G_1 for adversary $\mathcal{A}$ from the proof of Theorem 7.

C Appendix for Section 4 (Deniable AKEM)

We formalise the notion of ciphertext indistinguishability for an authenticated key encapsulation mechanism AKEM via the game $(n,Q_{\mathtt{Enc}},Q_{\mathtt{Dec}},Q_{\mathtt{CSK}},Q_{\mathtt{Chl}})$-**Ins-CCA**$_{\mathsf{AKEM}}(\mathcal{A})$ depicted in Figure 24 and define the advantage of adversary $\mathcal{A}$ as

$$\mathrm{Adv}_{\mathsf{AKEM},\mathcal{A}}^{(n,Q_{\mathtt{Enc}},Q_{\mathtt{Dec}},Q_{\mathtt{CSK}},Q_{\mathtt{Chl}})\text{-}\mathbf{Ins\text{-}CCA}} := \left|\Pr\left[(n,Q_{\mathtt{Enc}},Q_{\mathtt{Dec}},Q_{\mathtt{CSK}},Q_{\mathtt{Chl}})\text{-}\mathbf{Ins\text{-}CCA}_{\mathsf{AKEM}}(\mathcal{A})\Rightarrow 1\right]-\frac{1}{2}\right|.$$

Game $(n,Q_{\mathtt{Enc}},Q_{\mathtt{Dec}},Q_{\mathtt{CSK}},Q_{\mathtt{Chl}})$-**Ins-CCA**$_{\mathsf{AKEM}}(\mathcal{A})$	**Oracle** $\mathtt{Decps}(pk,r\in[n],c)$	**Oracle** $\mathtt{Chall}(s\in[n],r\in[n])$
for $i\in[n]$	**if** $\exists\, k:(pk,pk_r,c,k)\in\mathcal{D}$	**if** $r\in\mathcal{C}$
$(sk_i,pk_i)\xleftarrow{\$}\mathsf{Gen}$	**return** k	**return** $\perp$
$b\xleftarrow{\$}\{0,1\}$	$k\leftarrow\mathsf{Dec}(pk,sk_r,c)$	$(c,k)\xleftarrow{\$}\mathsf{Enc}(sk_s,pk_r)$
$b'\leftarrow\mathcal{A}^{\mathtt{Encps},\mathtt{Decps},\mathtt{Chall},\mathtt{CorSK}}(pk_1,\dots,pk_n)$	**return** k	**if** $b=0$
return $[\![b=b']\!]$	**Oracle** $\mathtt{CorSK}(i\in[n],sk)$	**continue**
Oracle $\mathtt{Encps}(s\in[n],pk)$	$sk_i\leftarrow sk$	**if** $b=1$
$(c,k)\xleftarrow{\$}\mathsf{Enc}(sk_s,pk)$	$pk_i\leftarrow\mu(pk)$	$k\xleftarrow{\$}\mathcal{K}$
return (c,k)	$\mathcal{C}\leftarrow\mathcal{C}\cup\{i\}$	$\mathcal{D}\leftarrow\mathcal{D}\cup\{(pk_s,pk_r,c,k)\}$
		return (c,k)

Figure 24. Game defining **Ins-CCA** for an authenticated key encapsulation mechanism AKEM with adversary $\mathcal{A}$ making at most $Q_{\mathtt{Enc}}$ queries to Encps, at most $Q_{\mathtt{Dec}}$ queries to Decps, at most $Q_{\mathtt{CSK}}$ queries to CorSK, and at most $Q_{\mathtt{Chl}}$ queries to Chall.

Theorem 3 (KEM **IND-CCA** + H **PRF** $\implies$ AKEM **Ins-CCA**). *Let* KEM *be an* **IND-CCA** *secure key encapsulation mechanism and* H *a* **PRF**, *then* AKEM[KEM, RSig, SyE, H] *as depicted in Figure 10 is an* **Ins-CCA** *secure authenticated key encapsulation mechanism. In particular for any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* AKEM[KEM, RSig, SyE, H] *there exist a* **IND-CCA** *adversary* $\mathcal{B}$ *against* KEM *and a* **PRF** *adversary* $\mathcal{C}$ *against* H *such that*

$$\mathrm{Adv}_{\mathsf{AKEM[KEM,RSig,SyE,H]},\mathcal{A}}^{(n,Q_{Enc}Q_{Dec},Q_{CSK},Q_{Chl})\text{-}\mathbf{Ins\text{-}CCA}} \le \mathrm{Adv}_{\mathsf{KEM},\mathcal{B}}^{(n,Q_{Dec},Q_{Chl})\text{-}\mathbf{IND\text{-}CCA}} + \mathrm{Adv}_{\mathsf{H},\mathcal{C}}^{(Q_{Chl},Q_{Dec}+Q_{Chl})\text{-}\mathbf{PRF}}.$$

Proof of Theorem 3. Consider the sequence of games depicted in Figure 25.

Game G_0. This is the $\textbf{Ins-CCA}_{\mathsf{AKEM}}(\mathcal{A})$ game for $\mathsf{AKEM}[\mathsf{KEM},\mathsf{RSig},\mathsf{SyE},\mathsf{H}]$ so by definition

$$\left|\Pr[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,\mathsf{Q}_{\mathsf{Enc}}\mathsf{Q}_{\mathsf{Dec}},\mathsf{Q}_{\mathsf{CSK}},\mathsf{Q}_{\mathtt{Chl}})\text{-}\textbf{Ins-CCA}}_{\mathsf{AKEM}[\mathsf{KEM},\mathsf{RSig},\mathsf{SyE},\mathsf{H}],\mathcal{A}}.$$

Games $\mathsf{G}_0 - \mathsf{G}_2$

$\textbf{for } i \in [n]$
    $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
    $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
    $sk_i := (ksk_i, ssk_i)$
    $pk_i := (kpk_i, spk_i)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathtt{Encps},\mathtt{Decps},\mathtt{Chall},\mathtt{CorSK}}(pk_1, \dots, pk_n)$
$\textbf{return } [\![b = b']\!]$

Oracle $\mathtt{Encps}(s \in [n], pk)$

$\textbf{parse } pk \to (kpk, spk)$
$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
$m := kct||kpk_s||kpk||spk$
$\sigma' \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk\}, m)$
$kk \to kk_1||kk_2$
$\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
$c := (kct, \sigma)$
$k := \mathsf{H}(kk_2, \sigma||spk_s||m)$
$\textbf{return } (c, k)$

Oracle $\mathtt{Decps}(pk, r \in [n], c)$

$\textbf{if } \exists\, k : (pk, pk_r, c, k) \in \mathcal{E}$
    $\textbf{return } k$
$\textbf{parse } pk \to (kpk, spk)$
$\textbf{parse } c \to (kct, \sigma)$
$m \leftarrow kct||kpk||kpk_r||spk_r$
$kk \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
$kk \to kk_1||kk_2$
$k := \mathsf{H}(kk_2, \sigma||spk||m)$
$\sigma' \leftarrow \mathsf{SyE.Dec}_{kk_1}(\sigma)$
$\textbf{if } \exists\, kk' : (kpk_r, kct, kk') \in \mathcal{E}_{\mathsf{KEM}}$ // $\mathsf{G}_1 - \mathsf{G}_2$
    $kk' \to kk_1||kk_2$ // $\mathsf{G}_1 - \mathsf{G}_2$
    $k := \mathsf{H}(kk_2, \sigma||spk||m)$ // $\mathsf{G}_1 - \mathsf{G}_2$
    $\sigma' \leftarrow \mathsf{SyE.Dec}_{kk_1}(\sigma)$ // $\mathsf{G}_1 - \mathsf{G}_2$
    $k \xleftarrow{\$} \mathcal{K}$ // G_2
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(pk, pk_r, kct, k)\}$ // G_2
$\textbf{if } \mathsf{RSig.Ver}(\sigma', \{spk, spk_r\}, m) \neq 1$
    $\textbf{return } \perp$
$\textbf{return } k$

Oracle $\mathtt{Chall}(s \in [n], r \in [n])$

$\textbf{if } r \in \mathcal{R}$
    $\textbf{return } \perp$
$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$kk \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$ // $\mathsf{G}_1 - \mathsf{G}_2$
$\mathcal{E}_{\mathsf{KEM}} \leftarrow \mathcal{E}_{\mathsf{KEM}} \cup \{(kpk_r, kct, kk)\}$ // $\mathsf{G}_1 - \mathsf{G}_2$
$m := kct||kpk_s||kpk_r||spk_r$
$\sigma' \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk_r\}, m)$
$kk \to kk_1||kk_2$
$\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
$c := (kct, \sigma)$
$k := \mathsf{H}(kk_2, \sigma||spk_s||m)$
$k \xleftarrow{\$} \mathcal{K}$ // G_2
$\textbf{if } b = 0$
    $k := k$
$\textbf{if } b = 1$
    $k \xleftarrow{\$} \mathcal{K}$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(pk_s, pk_r, c, k)\}$
$\mathcal{E} \leftarrow \mathcal{E} \cup \{(pk_s, pk_r, c, k)\}$ // G_2
$\textbf{return } (c, k)$

Oracle $\mathtt{CorSK}(i \in [n], sk)$

$sk_i \leftarrow sk$
$pk_i \leftarrow \mu(sk)$
$\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$

Figure 25. Games $\mathsf{G}_0 - \mathsf{G}_2$ for the proof of Theorem 3.

Game G_1. In the challenge oracle, the KEM key kk is replaced with a uniformly random value from the KEM key space $\mathcal{K}_{\mathsf{KEM}}$, and stored alongside the receiver's key and ciphertext in the set $\mathcal{E}_{\mathsf{KEM}}$. Additionally, the decapsulation oracle is changed to check for a corresponding element in $\mathcal{E}_{\mathsf{KEM}}$ and the actual KEM key kk is replaced by the one stored in $\mathcal{E}_{\mathsf{KEM}}$.

Claim 14: There exists a PPT adversary $\mathcal{B}$ against the **IND-CCA** security of KEM, such that

$$\left|\Pr\left[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{KEM},\mathcal{B}}^{(n,\mathsf{Q_{Dec}},\mathsf{Q_{Chl}})\text{-}\mathbf{IND\text{-}CCA}}.$$

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 26. ∎

```
B^{Dec_KEM, Chall_KEM}(kpk_1, ..., kpk_n)
01 for i ∈ [n]
02   (ssk_i, spk_i) <-$ RSig.Gen
03   sk_i := (⊥, ssk_i)
04   pk_i := (kpk_i, spk_i)
05 b <-$ {0, 1}
06 b' <- A^{Encps,Decps,Chall,CorSK}(pk_1, ..., pk_n)
07 return [[b = b']]

Oracle Encps(s ∈ [n], pk)
08 return G_0.Encps(s, pk)

Oracle Decps(pk, r ∈ [n], c)
09 if ∃ k : (pk, pk_r, c, k) ∈ E
10   return k
11 parse pk -> (kpk, spk)
12 parse c -> (kct, σ)
13 kk <- Dec_KEM(r, kct)                 // call decapsualtion
14 m <- kct||kpk||kpk_r||spk_r
15 kk -> kk_1||kk_2
16 σ' <- SyE.Dec_{kk_1}(σ)
17 if RSig.Ver(σ', {spk, spk_r}, m) ≠ 1
18   return ⊥
19 k := H(kk_2, σ||spk||m)
20 return k

Oracle Chall(s ∈ [n], r ∈ [n])
21 if r ∈ R
22   return ⊥
23 (kct, kk) <-$ Chall_KEM(r)            // call challenge
24 m := kct||kpk_s||kpk_r||spk_r
25 σ' <- RSig.Sgn(ssk_s, {spk_s, spk_r}, m)
26 kk -> kk_1||kk_2
27 σ <- SyE.Enc_{kk_1}(σ')
28 c := (kct, σ)
29 k := H(kk_2, σ||spk_s||m)
30 if b = 0
31   k := k
32 if b = 1
33   k <-$ K
34   E <- E ∪ {(pk_s, pk_r, c, k)}
35 return (c, k)

Oracle CorSK(i ∈ [n], sk)
36 sk_i <- sk
37 pk_i <- μ(sk)
38 R <- R ∪ {i}
```

Figure 26. Adversary $\mathcal{B}$ against **IND-CCA** security of KEM having access to oracles $\mathtt{Dec}_{\mathsf{KEM}}$ and $\mathtt{Chall}_{\mathsf{KEM}}$ simulating $\mathsf{G}_1/\mathsf{G}_2$ for adversary $\mathcal{A}$ from the proof of Theorem 3.

Game G_2. In the challenge oracle, the output of H is replaced with a random value from the key space $\mathcal{K}$. Furthermore, regardless of the challenge bit's value (0 or 1), the challenge query outcome is stored in the bookkeeping set $\mathcal{E}$. The same changes are applied to the decapsulation oracle, but only when there is a matching element in the set $\mathcal{E}_{\mathsf{KEM}}$ as indicated by Line 27.

Claim 15: There exists a PPT adversary $\mathcal{C}$ against **PRF** security of H, such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{H},\mathcal{C}}^{(\mathsf{Q_{Chl}},\mathsf{Q_{Dec}}+\mathsf{Q_{Chl}})\text{-}\mathbf{PRF}}.$$

Proof. The adversary $\mathcal{C}$ is formally constructed in Figure 27. The first observation is that adding the elements to the bookkeeping set $\mathcal{E}$ does not impact the winning probability but ensures consistent outputs when changing k. Due to the changes in the previous game, the first input to H, kk_2, is uniformly random aligning exactly with the **PRF** game. Thus, an adversary $\mathcal{C}$ can simulate G_1 or G_2 (depending on their challenge bit) by selecting a new **PRF** key for each call to the challenge oracle. To correctly simulate the decapsulation oracle, they must identify the required **PRF** key. This is done in the same way as in the original game $\mathsf{G}_1/\mathsf{G}_2$ by storing results in the set $\mathcal{E}_{\mathsf{KEM}}$ but using an index ℓ instead of the actual key, which remains unknown to the **PRF** adversary.

$\underline{\mathcal{C}^{\mathtt{Eval}}}$

$\ell \leftarrow 0$
for $i \in [n]$
  $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
  $sk_i := (ksk_i, ssk_i)$
  $pk_i := (kpk_i, spk_i)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathtt{Encps},\mathtt{Decps},\mathtt{Chall},\mathtt{CorSK}}(pk_1, \ldots, pk_n)$
return $[\![b = b']\!]$

Oracle $\underline{\mathtt{Encps}(s \in [n], pk)}$

return $\mathsf{G}_0.\mathtt{Encps}(s, pk)$

Oracle $\underline{\mathtt{Decps}(pk, r \in [n], c)}$

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{E}$
  return k
parse $pk \rightarrow (kpk, spk)$
parse $c \rightarrow (kct, \sigma)$
$m \leftarrow kct||kpk||kpk_r||spk_r$
$kk \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
$kk \rightarrow kk_1||kk_2$
$k := \mathsf{H}(kk_2, \sigma||spk||m)$
if $\exists\, \ell' : (kpk_r, kct, \ell') \in \mathcal{E}_{\mathsf{KEM}}$ // check for index
  $k \leftarrow \mathtt{Eval}(\ell', \sigma||spk||m)$ // query oracle
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{(pk, pk_r, kct, k)\}$
$\sigma' \leftarrow \mathsf{SyE.Dec}_{kk_1}(\sigma)$
if $\mathsf{RSig.Ver}(\sigma', \{spk, spk_r\}, m) \neq 1$
  return $\perp$
return k

Oracle $\underline{\mathtt{Chall}(s \in [n], r \in [n])}$

if $r \in \mathcal{R}$
  return $\perp$
$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$kk \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$
$\ell \leftarrow \ell + 1$ // new key index
$\mathcal{E}_{\mathsf{KEM}} \leftarrow \mathcal{E}_{\mathsf{KEM}} \cup \{(kpk_r, kct, \ell)\}$ // store index
$m := kct||kpk_s||kpk_r||spk_r$
$\sigma' \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk_r\}, m)$
$kk \rightarrow kk_1||kk_2$
$\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
$c := (kct, \sigma)$
$k \leftarrow \mathtt{Eval}(\ell, \sigma||spk_s||m)$ // query oracle
if $b = 0$
  $k := k$
if $b = 1$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{(pk_s, pk_r, c, k)\}$
$\mathcal{E} \leftarrow \mathcal{E} \cup \{(pk_s, pk_r, c, k)\}$
return (c, k)

Oracle $\underline{\mathtt{CorSK}(i \in [n], sk)}$

$sk_i \leftarrow sk$
$pk_i \leftarrow \mu(sk)$
$\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$

Figure 27. Adversary $\mathcal{C}$ against **PRF** security of H having access to oracle `Eval` simulating $\mathsf{G}_1/\mathsf{G}_2$ for adversary $\mathcal{A}$ from the proof of Theorem 3.

∎

In G_2, the output distribution of the challenge oracle is now independent of challenge bit b an thus

$$\Pr[\mathsf{G}_2^{\mathsf{A}} \Rightarrow 1] = \frac{1}{2}.$$

Adding up the analysed bounds yields the bound stated in the Theorem. ∎

Theorem 5 (RSig **MC-Ano** $\implies$ AKEM **DR-Den**). *Let* RSig *be a ring signature which is multi-challenge anonymous under full key exposure, then* AKEM[KEM, RSig, SyE, H] *as depicted in Figure 10 is an* ***DR*-Den** *secure authenticated key encapsulation mechanism. In particular, for any* ***DR*-Den** *adversary* $\mathcal{A}$ *against* AKEM[KEM, RSig, SyE, H] *there exists a* PPT *simulator* Sim *and a* **MC-Ano** *adversary* $\mathcal{B}$ *against* RSig *such that*

$$\mathrm{Adv}^{(n,\mathsf{Q}_{\mathtt{Chl}})\text{-}\boldsymbol{DR}\text{-}\mathbf{Den}}_{\mathsf{AKEM[KEM,RSig,SyE,H]},\mathcal{A},\mathsf{Sim}} \leq \mathrm{Adv}^{(n,\mathsf{Q}_{\mathtt{Chl}})\text{-}\mathbf{MC\text{-}Ano}}_{\mathsf{RSig},\mathcal{B}}.$$

Proof. We show the existence of a simulator Sim such that the upper bound on the advantage holds. The simulator is depicted in Figure 28.

```
Sim(pk_s, pk_r, sk_r)
parse pk_s → (kpk_s, spk_s)
parse pk_r → (kpk_r, spk_r)
parse sk_r → (ksk_s, ssk_s)
(kct, kk) ←$ KEM.Enc(kpk_r)
m ← (kct, kpk_s, kpk_r, spk_r)
σ ←$ RSig.Sgn(ssk_r, {spk_s, spk_r}, m)
c := (kct, σ)
k := H(kk, σ, spk_s, m)
return (c, k)
```

Figure 28. Simulator for the proof of Theorem 5.

Consider the sequence of games depicted in Figure 29.

Game G_0. This is the $(n, Q_{\mathtt{Chl}})$-**DR-Den** game for AKEM[KEM, RSig, SyE, H] and simulator Sim as described in Figure 28 so by definition

$$\left|\Pr[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,\mathsf{Q}_{\mathtt{Chl}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM[KEM,RSig,SyE,H]},\mathcal{A},\mathsf{Sim}}.$$

Game G_1. In this game, the signature in the challenge oracle is now created with the receiver's signing key instead of the sender's.

Claim 16: There exists a PPT adversary $\mathcal{C}$ against the **MC-Ano** security of RSig, such that

$$\left|\Pr\left[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{(n,\mathsf{Q}_{\mathtt{Chl}})\text{-}\mathbf{MC\text{-}Ano}}_{\mathsf{RSig},\mathcal{B}}.$$

Proof. The adversary is formally constructed in Figure 30. Adversary $\mathcal{B}$ perfectly simulates Game G_0 in their own case $b = 0$ and Game G_1 in case $b = 1$. Note that calls from the AKEM challenge oracle automatically fulfill all the requirements of the challenge oracle from the anonymity game by default. ∎

In Game G_1, judge $\mathcal{A}$ cannot distinguish the challenge bit b anymore since the output of the challenge is independent of b. We obtain

$$\Pr[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1] = \frac{1}{2}.$$

∎

Games $\mathsf{G}_0 - \mathsf{G}_1$

for $i \in [n]$
  $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
  $sk_i := (ksk_i, ssk_i)$
  $pk_i := (kpk_i, spk_i)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\texttt{Rev},\texttt{Chall}}(pk_1, \ldots, pk_n)$
return $[\![b = b']\!]$

$\texttt{Rev}(i \in [n])$

return sk_i

Oracle $\texttt{Chall}(s \in [n], r \in [n])$

$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
$\sigma \xleftarrow{\$} \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk_r\}, m)$
$\sigma \xleftarrow{\$} \mathsf{RSig.Sgn}(ssk_r, \{spk_s, spk_r\}, m)$ // G_1
$c := (kct, \sigma)$
$k := \mathsf{H}(kk, \sigma, spk_s, m)$
if $b = 0$
  continue
if $b = 1$
  $(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
  $m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
  $\sigma \xleftarrow{\$} \mathsf{RSig.Sgn}(ssk_r, \{spk_s, spk_r\}, m)$
  $c := (kct, \sigma)$
  $k := \mathsf{H}(kk, \sigma, spk_s, m)$
return (c, k)

Figure 29. Games $\mathsf{G}_0 - \mathsf{G}_1$ for the proof of Theorem 5.

$\mathcal{B}^{\texttt{Chl}_{\mathsf{RSig}}}((ssk_1, spk_1), \ldots, (ssk_n, spk_n))$

for $i \in [n]$
  $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  $sk_i := (ksk_i, ssk_i)$
  $pk_i := (kpk_i, spk_i)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\texttt{Rev},\texttt{Chall}}(pk_1, \ldots, pk_n)$
return $[\![b = b']\!]$

$\texttt{Rev}(i \in [n])$

return sk_i

Oracle $\texttt{Chall}(s \in [n], r \in [n])$

$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
$\sigma \xleftarrow{\$} \texttt{Chl}_{\mathsf{RSig}}(s, r, \{spk_s, spk_r\}, m)$
$c := (kct, \sigma)$
$k := \mathsf{H}(kk, \sigma, spk_s, m)$
if $b = 0$
  continue
if $b = 1$
  $(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
  $m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
  $\sigma \xleftarrow{\$} \mathsf{RSig.Sgn}(ssk_r, \{spk_s, spk_r\}, m)$
  $c := (kct, \sigma)$
  $k := \mathsf{H}(kk, \sigma, spk_s, m)$
return (c, k)

Figure 30. Adversary $\mathcal{B}$ against **MC-Ano** security of RSig having access to oracle $\texttt{Chl}_{\mathsf{RSig}}$ simulating $\mathsf{G}_0/\mathsf{G}_1$ from the proof of Theorem 5.

Theorem 6 (KEM **IND-CPA** + SyE **PRP** $\implies$ AKEM **HR-Den**). *Let* KEM *be an* **IND-CPA** *secure key encapsulation mechanism and* SyE *a symmetric encryption scheme, then* AKEM[KEM, RSig, SyE, H] *as depicted in Figure 10 is a* ***HR*-Den** *secure authenticated key encapsulation mechanism in the honest receiver setting. In particular, for any* ***HR*-Den** *adversary* $\mathcal{A}$ *against* AKEM[KEM, RSig, SyE, H] *there exists a* PPT *simulator* Sim*, a* **IND-CPA** *adversary* $\mathcal{B}$ *against* KEM*, and a* **PRP** *adversary* $\mathcal{C}$ *against* SyE *such that*

$$\mathrm{Adv}^{(n,\mathrm{Q}_{\mathit{Chl}})\text{-}\boldsymbol{HR}\textbf{-Den}}_{\mathsf{AKEM[KEM,RSig,SyE,H]},\mathcal{A},\mathsf{Sim}} \leq \mathrm{Adv}^{(n,\mathrm{Q}_{\mathit{Chl}})\text{-}\textbf{IND-CPA}}_{\mathsf{KEM},\mathcal{B}} + \mathrm{Adv}^{(\mathrm{Q}_{\mathit{Chl}},\mathrm{Q}_{\mathit{Chl}})\text{-}\textbf{PRP}}_{\mathsf{SyE},\mathcal{C}}.$$

Proof. We show that the existence of a simulator Sim such that the upper bound on the advantage holds. The simulator is depicted in Figure 31.

$\mathsf{Sim}(pk_s, pk_r)$

parse $pk_s \to (kpk_s, spk_s)$
parse $pk_r \to (kpk_r, spk_r)$
$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
$kk \to kk_1 || kk_2$
$\sigma \xleftarrow{\$} \mathcal{S}$
$c := (kct, \sigma)$
$k := \mathsf{H}(kk_2, \sigma, spk_s, m)$
return (c, k)

Figure 31. Simulator for the proof of Theorem 6.

Consider the sequence of games depicted in Figure 32.

Game G_0. This is the $(n, Q_{\texttt{Chl}})$-**HR-Den** game for $\mathsf{AKEM}[\mathsf{KEM}, \mathsf{RSig}, \mathsf{SyE}, \mathsf{H}]$ in the honest receiver setting and simulator Sim as described in Figure 31 so by definition

$$\left|\Pr[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n, Q_{\texttt{Chl}})\text{-}\mathbf{HR\text{-}Den}}_{\mathsf{AKEM}[\mathsf{KEM},\mathsf{RSig},\mathsf{SyE},\mathsf{H}],\mathcal{A},\mathsf{Sim}}.$$

Games $\mathsf{G}_0 - \mathsf{G}_2$

$\mathcal{R}, \mathcal{C} \leftarrow \emptyset$
for $i \in [n]$
  $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
  $sk_i := (ksk_i, ssk_i)$
  $pk_i := (kpk_i, spk_i)$
$b \xleftarrow{\$} \{0, 1\}$
$b' \leftarrow \mathcal{A}^{\texttt{Rev},\texttt{Chall}}(pk_1, \ldots, pk_n)$
if $\mathcal{R} \cap \mathcal{C} \neq \emptyset$
  return $r \xleftarrow{\$} \{0, 1\}$
return $[\![b = b']\!]$

$\texttt{Rev}(i \in [n])$

$\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$
return sk_i

Oracle $\texttt{Chall}(s \in [n], r \in [n])$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{r\}$
$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$kk \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$ // $\mathsf{G}_1 - \mathsf{G}_2$
$m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
$\sigma' \xleftarrow{\$} \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk_r\}, m)$
$kk \to kk_1 || kk_2$
$\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
$\sigma \xleftarrow{\$} \mathcal{S}$ // G_2
$c := (kct, \sigma)$
$k := \mathsf{H}(kk_2, \sigma, spk_s, m)$
if $b = 0$
  continue
if $b = 1$
  $(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
  $m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
  $kk \to kk_1 || kk_2$
  $\sigma \xleftarrow{\$} \mathcal{S}$
  $c := (kct, \sigma)$
  $k := \mathsf{H}(kk_2, \sigma, spk_s, m)$
return (c, k)

Figure 32. Games $\mathsf{G}_0 - \mathsf{G}_2$ for the proof of Theorem 6.

Game G_1. In Game G_1, the KEM key is replaced by a uniformly random value from the KEM key space $\mathcal{K}_{\mathsf{KEM}}$.

Claim 17: There exists a PPT adversary $\mathcal{B}$ against the **IND-CPA** security of KEM, such that

$$\left|\Pr\left[\mathsf{G}_0^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{KEM},\mathcal{B}}^{(n,Q_{\mathtt{Chl}})\textbf{-IND-CPA}}.$$

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 33. Note that adversary $\mathcal{A}$ of Game $\mathsf{G}_0/\mathsf{G}_1$ is able to reveal secret keys via the `Rev` oracle. However, if they reveal a secret key corresponding to a receiver index of a `Chall` query, the game will be lost. Thus, the output of games with such an adversary is 0 anyway and it only remains to show the difference for adversaries without the knowledge of the receiver's secret keys.

$\underline{\mathcal{B}^{\mathtt{Chl},\mathtt{Rev}_{\mathsf{KEM}}}(kpk_1,\dots,kpk_n)}$

01 $\mathcal{R},\mathcal{C} \leftarrow \emptyset$
02 **for** $i \in [n]$
03 $\quad (ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
04 $\quad sk_i := (\perp, ssk_i)$
05 $\quad pk_i := (kpk_i, spk_i)$
06 $b \xleftarrow{\$} \{0,1\}$
07 $b' \leftarrow \mathcal{A}^{\mathtt{Rev},\mathtt{Chall}}(pk_1,\dots,pk_n)$
08 **if** $\mathcal{R} \cap \mathcal{C} \neq \emptyset$
09 $\quad$ **return** $r \xleftarrow{\$} \{0,1\}$
10 **return** $[\![b = b']\!]$

$\underline{\mathtt{Rev}(i \in [n])}$

11 $\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$
12 $ksk_i \leftarrow \mathtt{Rev}_{\mathsf{KEM}}(i)$ // KEM key corruption
13 **return** (ksk_i, ssk_i)

$\underline{\textbf{Oracle } \mathtt{Chall}(s \in [n], r \in [n])}$

14 $\mathcal{C} \leftarrow \mathcal{C} \cup \{r\}$
15 $(kct, kk) \leftarrow \mathtt{Chl}(r)$ // challenge query
16 $m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
17 $\sigma' \xleftarrow{\$} \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk_r\}, m)$
18 $kk \rightarrow kk_1 || kk_2$
19 $\sigma \leftarrow \mathsf{SyE.Enc}_{kk_1}(\sigma')$
20 $c := (kct, \sigma)$
21 $k := \mathsf{H}(kk_2, \sigma, spk_s, m)$
22 **if** $b = 0$
23 $\quad$ **continue**
24 **if** $b = 1$
25 $\quad (kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
26 $\quad m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
27 $\quad kk \rightarrow kk_1 || kk_2$
28 $\quad \sigma \xleftarrow{\$} \mathcal{S}$
29 $\quad c := (kct, \sigma)$
30 $\quad k := \mathsf{H}(kk_2, \sigma, spk_s, m)$
31 **return** (c, k)

Figure 33. Adversary $\mathcal{B}$ against **IND-CPA** security of KEM having access to oracles `Chl` and $\mathtt{Rev}_{\mathsf{KEM}}$ simulating $\mathsf{G}_0/\mathsf{G}_1$ from the proof of Theorem 6.

■

Game G_2. In Game G_2, the output of the symmetric encryption in the `Chall` is replaced by a uniformly random value of the signature space $\mathcal{S}$ (Line 21).

Claim 18: There exists a PPT adversary $\mathcal{C}$ against the **PRP** security of SyE, such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathsf{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathsf{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{SyE},\mathcal{C}}^{(Q_{\mathtt{Chl}},Q_{\mathtt{Chl}})\textbf{-PRP}}.$$

Proof. Adversary $\mathcal{C}$ is formally constructed in Figure 34. For the real case ($b = 0$ in the **PRP** game), the reduction simulates G_1 for adversary $\mathcal{A}$. For the random case ($b = 1$), they simulate G_2. The total number of instances as well as oracle queries to `Eval` is $Q_{\mathtt{Chl}}$.

■

The output distribution of `Chall` in G_2 is now the same in case of $b = 0$ and $b = 1$, thus it holds

$$\Pr[\mathsf{G}_2^{\mathsf{A}} \Rightarrow 1] = \frac{1}{2}.$$

$\underline{\mathcal{C}^{\texttt{Eval}}}$

$\ell \leftarrow 0$
$\mathcal{R}, \mathcal{C} \leftarrow \emptyset$
for $i \in [n]$
  $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
  $sk_i := (ksk_i, ssk_i)$
  $pk_i := (kpk_i, spk_i)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\texttt{Rev},\texttt{Chall}}(pk_1, \dots, pk_n)$
if $\mathcal{R} \cap \mathcal{C} \neq \emptyset$
  return $r \xleftarrow{\$} \{0,1\}$
return $[\![b = b']\!]$

$\underline{\texttt{Rev}(i \in [n])}$

$\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$
return sk_i

Oracle $\underline{\texttt{Chall}(s \in [n], r \in [n])}$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{r\}$
$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$kk \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$
$m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
$\sigma' \xleftarrow{\$} \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk_r\}, m)$
$kk \rightarrow kk_1 || kk_2$
$\ell \leftarrow \ell + 1$ // new index
$\sigma \leftarrow \texttt{Eval}(\ell, \sigma')$ // query PRP oracle
$c := (kct, \sigma)$
$k := \mathsf{H}(kk_2, \sigma, spk_s, m)$
if $b = 0$
  continue
if $b = 1$
  $(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
  $m \leftarrow (kct, kpk_s, kpk_r, spk_r)$
  $kk \rightarrow kk_1 || kk_2$
  $\sigma \xleftarrow{\$} \mathcal{S}$
  $c := (kct, \sigma)$
  $k := \mathsf{H}(kk_2, \sigma, spk_s, m)$
return (c, k)

Figure 34. Adversary $\mathcal{C}$ against **PRP** security of SyE having access to oracle `Eval` simulating $\mathsf{G}_1/\mathsf{G}_2$ from the proof of Theorem 6.

■