This is the full version of a USENIX 2026 paper with the same title.

Shadowfax: Hybrid Security and Deniability for AKEMs

Phillip Gajland[1], Vincent Hwang[2,3], and Jonas Janneck[4]

[1] IBM Research Europe – Zurich
[2] Max Planck Institute for Security and Privacy
[3] Radboud University
[4] Ruhr University Bochum

12th January 2026

Abstract As cryptographic protocols transition to post-quantum security, most adopt hybrid solutions combining classical and post-quantum assumptions. This shift often sacrifices efficiency, compactness, or even security. One such property is *deniability*, which enables users to plausibly deny authorship of potentially incriminating messages. While classical protocols like X3DH key agreement (used in Signal and WhatsApp) provide deniability, post-quantum protocols like PQXDH and Apple's iMessage with PQ3 do not.

This work addresses this gap by investigating how to efficiently preserve deniability in post-quantum protocols. Specifically, we propose two hybrid schemes for authenticated key encapsulation mechanisms (AKEMs). The first is a black-box construction that preserves deniability when both constituent AKEMs are deniable. The second is Shadowfax, a non-black-box AKEM that achieves hybrid security, integrating a classical non-interactive key exchange, a post-quantum key encapsulation mechanism, and a post-quantum ring signature. Shadowfax satisfies deniability in both dishonest and honest receiver settings, relying on statistical security in the former and on a single pre- or post-quantum assumption in the latter.

Finally, we provide several portable implementations of Shadowfax. When instantiated with standardised components (ML–KEM and Falcon), Shadowfax yields ciphertexts of 1 728 bytes and public keys of 2 036 bytes, with encapsulation and decapsulation costs of 1.8M and 0.7M cycles on an Apple M1 Pro.

Contents

1 Introduction

The global roll out of post-quantum cryptography (PQC) is a monumental challenge. While the multi-year National Institute of Standards and Technology (NIST) standardisation [NIS16] has been a critical milestone, it marks only the beginning of a much larger effort. With several algorithms selected for standardisation (three standards have already been released [MLK24, MLD24, SLH24]), the next phase of implementation and adaptation is now underway. Migrating countless systems to PQC will likely take decades[5]. Nevertheless, significant progress has been made towards adapting widely deployed protocols for post-quantum security. For instance, X3DH [MP16], which supports billions of WhatsApp and Messenger users, has been upgraded to the post-quantum variant, PQXDH [KS24]. PQXDH is already deployed in Signal [KS24], and Apple's iMessage now uses PQ3 [App24]. However, these upgrades involve trade-offs: ciphertexts and keys grow larger, and deniability is often lost. PQXDH, for example, sacrifices the deniability of its predecessor X3DH due to a signature on the ephemeral key [FJ24]. Similarly, the analysis of Apple's iMessage with PQ3 [Ste24, LSB24], explicitly states that deniability is not a design goal. TLS [DA99, Res18] has also been updated for post-quantum security by using key encapsulation mechanisms (KEMs) in multiple papers [BCNS15, PST20, BBCT22] and real-world deployments [Lan16, Lan18, KV19, WR19].

A crucial aspect of all these adaptations is the *hybrid* approach, combining post-quantum algorithms with classical cryptographic methods. Post-quantum solutions, despite their potential, lack the decades of cryptographic analysis that traditional schemes like RSA and (EC)DH have undergone. Therefore, it is prudent to adopt hybrid solutions. This strategy is widely endorsed by national security agencies. The French National Agency for the Security of Information Systems (ANSSI) recommends a hybrid adoption of PQC [ANS23], and the German Federal Office for Information Security (BSI) explicitly states that *"post-quantum cryptography should not be used in isolation if possible, but only in hybrid mode"* for both key agreement and authentication [BSI22]. The BSI has reiterated this need in their recent updated technical guidelines, which *"only recommends the hybrid use of quantum-safe methods in combination with classical methods"* [BSI24].

1.1 Combiners

A combiner, or more generally hybrid scheme, ensures security as long as at least one of its components remains secure. For instance, if cryptographically relevant quantum computers (CRQCs) become available rendering classical schemes insecure [Sho94], the hybrid scheme would still be secure as the post-quantum component remains intact. Conversely, if advances in cryptanalysis or implementation issues break the post-quantum scheme, the classical security of the hybrid scheme would still hold due to the hardness of the classical problem. In fact, recent work demonstrated several classical attacks on post-quantum schemes [Beu22, CD23, MMP+23, Rob23] underscoring the importance of hybrids. Additionally, since only basic post-quantum primitives like KEMs and signatures have been standardised, many applications necessarily need to rely on non-standard primitives, further supporting hybrid configurations. Finally, to achieve *"cryptographic agility"* [OP19] in the long run, the permanent use of hybrid solutions may become a common practice.

Generic Combiners. Combiners are typically defined relative to a *primitive* and security *notion*, where security is *binary* – either the scheme is secure (Π) or insecure ($\neg\Pi$). Generic combiners allow statements like $\Pi_1 \vee \Pi_2 \implies \Pi$, meaning that as long as one of the schemes satisfies the security notion, the combined scheme also satisfies it. For instance, consider the *pseudorandomness* of a PRG, or *confidentiality* of a KEM. A generic combiner for the former is $\mathsf{PRG}(s_1 \| s_2) := \mathsf{PRG}_1(s_1) \oplus \mathsf{PRG}_2(s_2)$, and for the latter, KEM with $k = \mathsf{H}(k_1, k_2, c_1, c_2)$, when H is modelled as a random oracle. Such combiners are the most powerful as they are the most general.

[5] Although it has been known for over twenty years that `MD5` [Riv92] fails to provide collision resistance [WFLY04], recent research continues to exploit this insecurity in new vulnerabilities [GHH+24] within prevalent protocols.

PQC Migration. While generic combiners are sufficient for PQC migration, constructing them can be challenging in some cases, and we argue they may be unnecessarily restrictive. To understand why, one must consider the primary motivation for using combiners in PQC migration which is risk mitigation against the failure of *assumptions*. Therefore, instead of reasoning about combiners for security *notions*, one can relax this requirement to focus specifically on the underlying *assumptions* for PQC migration. Specifically, we argue that it suffices to construct combiners of the form $\text{Assumption}_{\text{pre-Q}} \vee \text{Assumption}_{\text{post-Q}} \implies \Pi$, where the security of the scheme holds as long as one of the classical or post-quantum *assumptions* is not broken. We call such a scheme a *hybrid*. This relaxation simplifies combiner design, enables constructions where generic approaches remain elusive [FH25, GRSV25, HR25, LL25, Jan25] and better aligns with the primary motivation for adopting hybrid solutions in PQC migration.

Additional related work on combiners and hybrids is further discussed in Appendix A.1.

1.2 AKEM

The integration of combiners for KEMs and signatures – ensuring confidentiality and authenticity – has already begun in the context of PQC migration. An Authenticated Key Encapsulation Mechanism (AKEM) shares the same interfaces as a standard KEM, with two key differences: encapsulation proves the sender's authenticity requiring their secret key, while decapsulation verifies the sender's authenticity using their public key. Therefore, the primitive includes both notions of confidentially and authenticity. Introduced in the HPKE standard [BBLW22], an AKEM draws inspiration from the signcryption literature [DZ10] and generalises the split-KEM primitive [BFG+20].[6] An additional feature of several AKEM constructions [ABH+21, AJKL23, CHN+24, GJK24b, JMOR25], and later formalised in [CHN+24, GJK24b] is *deniability*. This property allows a sender to deny having sent a particular ciphertext.[7] Beyond its application to HPKE, which is specified to be used in the Message Layer Security (MLS) [BBR+23] protocol, AKEMs find applications in authenticated key exchange and secure messaging [BS20, BFG+20], for instance in K-Waay [CHN+24].

Another related work discusses PQ deniable authenticated key exchange [HKKP22]. Unlike an AKEM which is a one-shot primitive, the protocol from [HKKP22] is interactive. However, the core part of the protocol could be viewed as an AKEM (with ephemeral AKEM keys). Similar to our scheme which we will present later, the construction from [HKKP22] is based on a KEM and a ring signature which can be efficiently instantiated from PQ primitives. A key difference to our approach is that we consider hybrid security guarantees along with the associated challenges.

More generally, despite the existence of both classical [ABH+21, AJKL23] and post-quantum [AJKL23, CHN+24, GJK24b, JMOR25, Nio25, HKKP22] constructions, there are no known hybrid AKEMs. This leads to the natural question:

"Can hybrid AKEMs efficiently preserve deniability?"

1.3 Technical Overview

In this work, we answer the aforementioned question in the affirmative. The following section presents a high-level summary of our technical contributions.

AKEM. The two main security properties are confidentiality and authenticity. Another desirable property is (sender) deniability, allowing the sender to plausibly deny sending a ciphertext even though the receiver can verify the sender's identity. This is formalised by showing the existence of a simulator Sim, whose output ciphertext c and key k are indistinguishable from those produced by the encapsulation algorithm Enc, to any adversary $\mathcal{A}$. The model of deniability varies depending on the scenario [GJK24b]. For a *dishonest receiver*,

[6] In fact, a symmetric split-KEM [BFG+20, Def. 4] is equivalent to an AKEM [ABH+21, Def. 9].

[7] To the best of our knowledge, combiners for deniability have not yet been studied. For background on deniability we refer to Appendix A.

Sim is given the receiver's secret key, modelling a scenario where the receiver could forge a ciphertext c to falsely attribute it to the sender. For *honest receivers*, the receiver is assumed to not simulate any values, so Sim is not given the receiver's secret key.

Black-Box Construction. A natural way to construct a hybrid AKEM would be to use the *"parallel KEM combiner"*, where the ciphertext $c := (c_1, c_2)$ and key $k := \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$. Here $(c_i, k_i) \xleftarrow{\$} \mathsf{Enc}_i(sk_i, pk_i)$ for $i \in \{1, 2\}$ and $(sk_s = (sk_1, sk_2), pk_r = (pk_1, pk_2))$. Confidentiality is guaranteed as long as *one of* AKEM_1 *or* AKEM_2 provides confidentiality, akin to the KEM combiner from [GHP18]. However, we additionally include the sender and receiver's public keys in H to satisfy the strong notion of insider CCA security (see Theorem 9). Similarly, for *authenticity*, we show that the resulting scheme inherits authenticity as long as *one of* AKEM_1 *or* AKEM_2 satisfies authenticity (see Theorem 10).

For *deniability*, we were only able to prove that the resulting scheme satisfies deniability if *both* AKEM_1 *and* AKEM_2 provide deniability (see Theorems 11 and 12). This result is unsatisfactory, as it undermines the purpose of the hybrid scheme. The challenge lies in constructing a simulator for the combined scheme without having a simulator for either AKEM_1 or AKEM_2. Specifically, such a simulator needs to output a ciphertext and key that are indistinguishable from those generated by the AKEM encapsulation. If one of the underlying AKEMs is not deniable (whether for and honest or dishonest receiver), then a distinguisher exists for any potential simulator. While the key can be simulated using known KEM combiner techniques, the problem arises when trying to simulate both ciphertext components, as no simulator exists for one of the components. That makes it hard to satisfy hybrid-like deniability that only relies on the security properties of one of the AKEMs. Interestingly, an AKEM can be constructed satisfying statistical *dishonest receiver deniability* from a ring signature scheme with information-theoretic anonymity. Although this is not true of schemes like SMILE [LNS21] and Erebor [BLL24], Gandalf [GJK24b] does satisfy the necessary anonymity. However, it appears unlikely that *honest receiver deniability* can be achieved information theoretically, as prior techniques still require computational assumptions. For instance, in [GJK24b], the authors *"boost"* an AKEM to satisfy honest receiver deniability by symmetrically encrypting the ring signature with a KEM key, which, in turn, depends on a computational assumption. Therefore, instead of analysing security notions at a black-box level, we consider the underlying computational hardness assumptions needed to achieve these security properties. This requires examining AKEM constructions at a lower abstraction level.

Shadowfax. The Shadowfax construction consists of two parts: the post-quantum component and the classical component. A high level overview of the construction is depicted in Figure 1.

The post-quantum component. The post-quantum part relies on a post-quantum KEM, a post-quantum ring signature scheme, a symmetric encryption scheme, and a PRF used as a key derivation function (KDF). For confidentiality, the receiver's KEM public key kpk_r is input to the encapsulation procedure, returning a KEM ciphertext kct and KEM key kk. For authenticity, the KEM ciphertext kct is signed using a ring signature scheme with the sender's signing key ssk_s, where the signing ring consists of the sender's public key spk_s and the receiver's public key spk_r. The anonymity property of the ring signature implies dishonest receiver deniability [GJK24b]. However, if the anonymity relies on computational assumptions as mentioned above, deniability would be lost if those assumptions turn out to be flawed. To avoid this, the ring signature scheme must provide statistical anonymity. However, this alone does not suffice for honest receiver deniability, since the signature σ is publicly verifiable. If the signature is unforgeable, the ciphertext must have originated from the sender or from the receiver, since only they can sign for the respective ring. However, since the receiver is honest (captured by the simulator not given the receiver's secret key), the adversary can deduce that the signature was issued by the sender. Therefore, the signature is symmetrically encrypted using the KEM key kk, ensuring that only the receiver can verify the signature. The symmetric ciphertext sct and KEM ciphertext kct become the AKEM ciphertext c in the Shadowfax construction, while the key k is derived from the KDF applied to sct and kk. This step is similar to the KEM combiner from [GHP18]. To ensure strong confidentiality, specifically insider CCA security, the KDF also includes the public keys of both the sender and the receiver. Up to this point, this construction mirrors the PQ-AKEM from [GJK24b].

The classical component. To obtain a hybrid scheme we add the classical part of Shadowfax, that integrates two non-interactive key exchanges (NIKEs), similar to the DH-AKEM construction of [ABH+21], and intertwines them with the post-quantum component. During encapsulation, an ephemeral NIKE key pair (nsk_e, npk_e) is generated to provide confidentiality, with the NIKE public key npk_e appended to the ciphertext. The ephemeral NIKE secret key nsk_e is then used to compute a shared NIKE key nk_e with the receiver's NIKE public key npk_e and this key is fed into the KDF. That is, Shadowfax provides confidentiality as long as *one of* the post-quantum KEM *or* the classical NIKE is secure (see Theorem 14). For (implicit) authentication, the sender's long-term NIKE secret key nsk_s is used with the receiver's NIKE public key npk_r to derive a NIKE shared key nk_ℓ, which is also used as input to the KDF. This ensures authenticity of Shadowfax as long as *one of* the post-quantum ring signature is unforgeable, *or* the classical NIKE remains secure (see Theorem 15).

The dishonest receiver deniability of Shadowfax holds as long as the ring signature scheme is information theoretically anonymous and the NIKE is correct (see Theorem 16). Finally, combining a KEM and ring signature scheme with two NIKEs would not satisfy honest receiver deniability, as the signature is publicly verifiable. Recall that we symmetrically encrypted the signature using the post-quantum KEM key. However, if the CCA security is compromised, then honest receiver deniability would be lost. To mitigate this, both the KEM key kk and the ephemeral NIKE shared key nk_e are run through another KDF before symmetrically encrypting the signature σ. This ensures that also the honest receiver deniability of Shadowfax is satisfied in a hybrid sense. Specifically, honest receiver deniability is preserved as long as *one of* the post-quantum KEM is CPA secure *or* the classical NIKE remains secure (see Theorem 17).

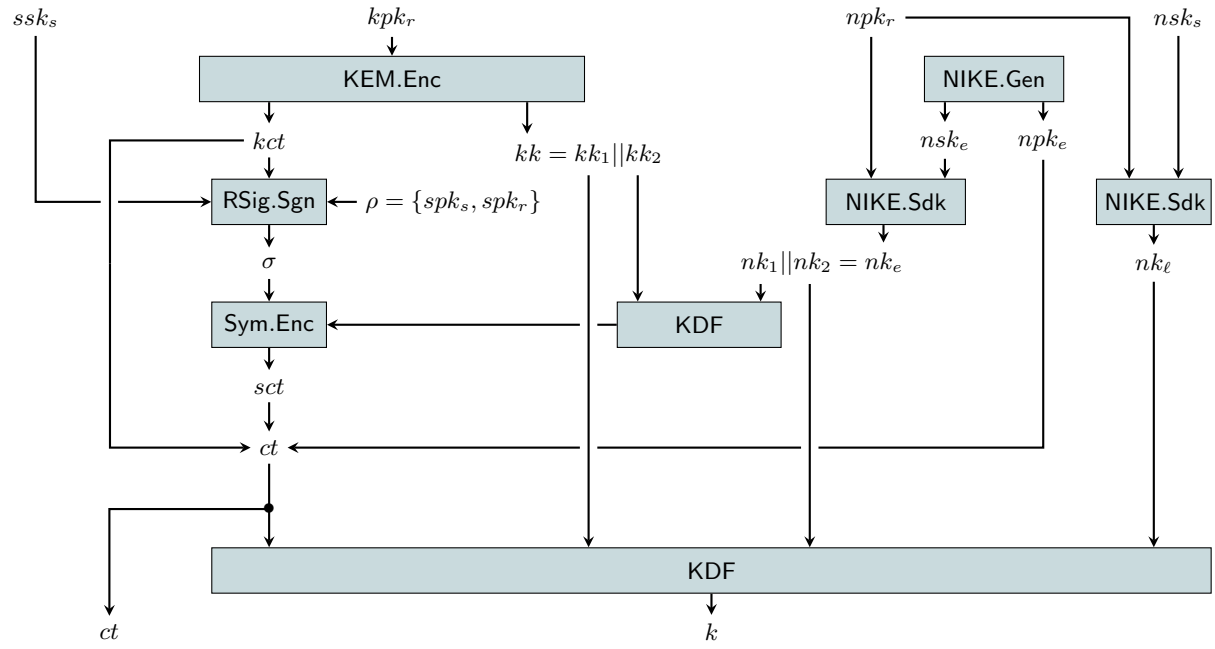

Figure 1. High level overview of the Shadowfax construction.

1.4 Contributions

This work considers hybrid schemes that preserve deniability, an area previously unexplored. Specifically, we focus on authenticated key encapsulation mechanisms (AKEMs) and present the following contributions.

Black-Box Construction. At the highest level of abstraction, we present a black-box construction that combines two AKEMs. We prove that deniability is preserved if both underlying schemes are deniable. Moreover, confidentiality and authenticity are guaranteed as long as one of the AKEMs provides the desired notion, aligning with the expected behaviour of a combiner.

Non-Black-Box Construction. At a lower level of abstraction, we introduce Shadowfax, a non-black-box AKEM that achieves hybrid security, built from a classical NIKE, a post-quantum KEM, and a post-quantum ring signature scheme. We show that Shadowfax achieves deniability in two distinct settings. In the dishonest receiver setting, deniability relies on the correctness of the NIKE and the (possibly statistical) anonymity of the ring signature. In the honest receiver setting, deniability holds under *one of* two computational assumptions: the security of the ephemeral NIKE *or* the KEM.

Implementation. Our final contribution is a set of portable implementations designed for compactness, reproducibility, and easy integration into existing cryptographic libraries. These include the first implementations of the Gandalf ring signature scheme and post-quantum AKEM from [GJK24b], as well as the hybrid AKEM Shadowfax.

Our Gandalf implementation with Falcon achieves 1.29× faster key generation and 1.63× faster signing compared to concurrent work [KNTW25, Tab. 3]. When instantiated with standardised components, such as Falcon and ML-KEM, Shadowfax achieves ciphertexts of 2 036 bytes and public keys of 1 728 bytes. With non-standard components, ciphertexts can be reduced to 1 781 bytes and public keys to 1 449 bytes (see Section 5).

For our instantiation with standardised components, encapsulation takes 1.8 million cycles, and decapsulation takes about 675 000 cycles on a Firestorm core running at 3 GHz on an Apple M1 Pro. Detailed parameters and benchmarks appear in Table 3, Table 4, and the project's GitHub repository at Shadowfax. [8]

2 Preliminaries

We introduce some relevant definitions used throughout the paper. Further notions can be found in Appendix B.

2.1 Notations

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 $[n] \coloneqq \{1, \ldots, n\}$. The notation $[\![b]\!]$, where b is a boolean statement, evaluates to 1 if the statement is true and 0 otherwise. We use uppercase letters $\mathcal{A}, \mathcal{B}, \ldots$ 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 $(x_1, \ldots)$ and $t_{\mathcal{A}}$ to denote the time of $\mathcal{A}$. 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 $\sup(X) \coloneqq \{x \in \mathbb{R} \mid \Pr[X = x] > 0\}$.

Security Games. We use standard code-based security games [BR06]. 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 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

[8] https://github.com/vincentvbh/shadowfax

oracle procedure $\texttt{Oracle}$ of game G on input x, we shortly write $\mathsf{G}.\texttt{Oracle}(x)$. Throughout the proofs we rely on game hopping and the main difference lemma [BR06].

2.2 AKEM

Definition 1 (Authenticated Key Encapsulation Mechanism [ABH+21, Def. 9]). An *authenticated key encapsulation mechanism* AKEM is defined as a tuple $\mathsf{AKEM} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of the following algorithms.

$(sk, pk) \xleftarrow{\$} \mathsf{Gen}$**:** The probabilistic generation algorithm Gen returns a secret key sk and a corresponding public key pk. We implicitly assume that pk defines 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_{AKEM} is defined as

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

where the probability is over the randomness of Gen and Enc.

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 notion of CCA security for an AKEM, in particular that of insider security [ABH+21]. As a building block we will also need a weaker notion of CCA security, namely outsider security [ABH+21]. We formalise the notion of ciphertext indistinguishability for an authenticated key encapsulation mechanism AKEM via the games depicted in Figure 2 and Figure 3, respectively. The advantage of adversary $\mathcal{A}$ is defined as

$$\mathrm{Adv}^{Q_{\mathrm{I}}\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM},\mathcal{A}} := \left|\Pr\left[Q_I\text{-}\mathbf{Ins\text{-}CCA}_{\mathsf{AKEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2}\right|,$$

$$\mathrm{Adv}^{Q_{\mathrm{O}}\text{-}\mathbf{Out\text{-}CCA}}_{\mathsf{AKEM},\mathcal{A}} := \left|\Pr\left[Q_O\text{-}\mathbf{Out\text{-}CCA}_{\mathsf{AKEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2}\right|,$$

for $Q_I = (n, Q_{\texttt{Enc}}, Q_{\texttt{Dec}}, Q_{\texttt{Chl}}), Q_O = (n, Q_{\texttt{Enc}}, Q_{\texttt{Dec}})$.

Authenticity. We consider outsider authenticity from [ABH+21], the strongest notion that is achievable when also seeking deniability [GJK24b]. We formalise the notion via the game depicted in Figure 4 and define the advantage of an adversary $\mathcal{A}$ as

$$\mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM},\mathcal{A}} := \left|\Pr\left[Q\text{-}\mathbf{Out\text{-}Aut}_{\mathsf{AKEM}}(\mathcal{A}) \Rightarrow 1\right] - \tfrac{1}{2}\right|,$$

for $Q = (n, Q_{\texttt{Enc}}, Q_{\texttt{Chl}})$.

Deniability. As in [GJK24b], we consider deniability in two independent settings. For *dishonest receiver* deniability, the receiver is potentially dishonest and capable of simulating ciphertexts. Therefore, the simulator is also given the receiver's secret key. In contrast, in the *honest receiver* setting, the receiver is assumed to behave honestly, and the simulator only has access to public key material. For an authenticated key encapsulation mechanism AKEM and a simulator Sim, we define deniability in the *dishonest receiver* setting and *honest receiver* setting via the games depicted in Figure 5. The advantage of an adversary $\mathcal{A}$ is then defined as

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

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

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

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

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

$(c, k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk)$
return (c, k)

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

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
$k \leftarrow \mathsf{Dec}(pk, sk_r, c)$
return k

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

$(c, k) \xleftarrow{\$} \mathsf{Enc}(sk, pk_r)$
if $b = 1$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), pk_r, c, k)\}$
return (c, k)

Figure 2. Game defining **Ins-CCA** for an authenticated key encapsulation mechanism $\mathsf{AKEM} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with adversary $\mathcal{A}$ making at most; $Q_{\texttt{Enc}}$ queries to `Encps`, $Q_{\texttt{Dec}}$ queries to `Decps`, $Q_{\texttt{CSK}}$ queries to `CorSK`, and $Q_{\texttt{Chl}}$ queries to `Chall`.

Game $(n, Q_{\texttt{Enc}}, Q_{\texttt{Dec}})$-**Out-CCA**$_{\mathsf{AKEM}}(\mathcal{A})$

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

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

$(c, k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk)$
if $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\}$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

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

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
$k \leftarrow \mathsf{Dec}(pk, sk_r, c)$
return k

Figure 3. Game defining **Out-CCA** for an authenticated key encapsulation mechanism $\mathsf{AKEM} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with adversary $\mathcal{A}$ making at most; $Q_{\texttt{Enc}}$ queries to `Encps` and $Q_{\texttt{Dec}}$ queries to `Decps`.

2.3 Pseudorandom Function

Definition 2 (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 6 and define the advantage of adversary $\mathcal{A}$ as

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

Based on a PRF one can also define a dual-PRF [Bel06, Bel15] which means that the function can be keyed on either the actual key or the (fixed-length) input. This was even further generalised as a split-key PRF [GHP18]. To obtain a uniformly random key from an unpredictable input, a random oracle would be needed. However, if the secrets are uniformly random, then a split-key PRF is sufficient. In particular, the output of the sk-PRF will be pseudorandom. The idea is that adversary $\mathcal{A}$ can first choose the key that is attacked (position j) and is then playing the normal PRF game where the remaining keys (for positions $\ell \in [m] \setminus \{j\}$) that were not chosen as the attacked key act as the input to the function. Note that a sk-PRF can be generically instantiated by calling a dual-PRF multiple times sequentially.

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

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

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

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

Figure 4. Game defining **Out-Aut** for an authenticated key encapsulation mechanism $\mathsf{AKEM} \coloneqq (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with adversary $\mathcal{A}$ making at most $Q_{\texttt{Enc}}$ queries to `Encps` and $Q_{\texttt{Chl}}$ queries to `Chall`.

Games $(n, Q_{\texttt{Chl}})$-**DR-Den**$_{\mathsf{AKEM},\mathsf{Sim}}(\mathcal{A})$ and $(n, Q_{\texttt{Chl}})$-**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**
  abort // **HR-Den**
return $[\![b = b']\!]$

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

if $s = r$ **return** $\bot$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{r\}$
$(c, k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk_r)$
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)

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

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

Figure 5. 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 `Chall`.

Game $(n, Q_{\texttt{Eval}})$-**PRF**$_F(\mathcal{A})$

for $i \in [n]$
  $k_i \xleftarrow{\$} \mathcal{K}$
  $f_i \xleftarrow{\$} \{f \mid f : \{0,1\}^* \to \mathcal{R}\}$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\texttt{Eval}}$
return $[\![b = b']\!]$

Oracle $\texttt{Eval}(i \in [n], x)$

if $b = 0$
  return $F(k_i, x)$
if $b = 1$
  return $f_i(x)$

Figure 6. Game defining **PRF** for a keyed function F with adversary $\mathcal{A}$ making at most $Q_{\texttt{Eval}}$ queries to `Eval`.

Definition 3 (Split-Key Pseudorandom Function). A multi-keyed function with $m \in \mathbb{N}$ inputs, input space $\mathcal{K}_1 \times \ldots \times \mathcal{K}_m$, and output space $\mathcal{R}$ is a function $F_m : \mathcal{K}_1 \times \ldots \times \mathcal{K}_m \to \mathcal{R}$. We formalise the notion of *split-key pseudorandomess* for a multi-keyed function F_m via the game $(n, Q_{\texttt{Eval}})$-**PRF** depicted in Figure 7

and define the advantage of adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as

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

Game $(n, Q_{\mathtt{Eval}})$-$\mathbf{PRF}_{F_m}(\mathcal{A})$

$j \xleftarrow{\$} \mathcal{A}_1$
$\mathcal{K}' := \bigtimes_{\ell \in [m] \setminus \{j\}} \mathcal{K}_\ell$
for $i \in [n]$
  $k_i \xleftarrow{\$} \mathcal{K}_j$
  $f_i \xleftarrow{\$} \{f \mid f : \mathcal{K}' \to \mathcal{R}\}$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}_2^{\mathtt{Eval}}$
return $[\![b = b']\!]$

Oracle $\mathtt{Eval}(i \in [n], x)$

if $b = 0$
  parse $x \to (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_m)$
  return $F(x_1, \ldots, x_{j-1}, k_i, x_{j+1}, \ldots, x_m)$
if $b = 1$
  return $f_i(x)$

Figure 7. Game defining **PRF** for a multi-keyed function F_m with adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ making at most $Q_{\mathtt{Eval}}$ queries to `Eval`.

2.4 Non-Interactive Key Exchange (NIKE)

Definition 4 ((Simplified) Non-Interactive Key Exchange [FHKP12, App. G]). A *simplified non-interactive key exchange* NIKE is defined as a tuple $\mathsf{NIKE} := (\mathsf{Stp}, \mathsf{Gen}, \mathsf{Sdk})$ of the following algorithms.

$par \xleftarrow{\$} \mathsf{Stp}$**:** The probabilistic setup algorithm returns a set of system parameters par. We assume that par implicitly defines a shared key space $\mathcal{K}_{\mathsf{NIKE}}$ and is implicitly accessed by all other algorithms.

$(sk, pk) \xleftarrow{\$} \mathsf{Gen}$**:** Given system parameters par, the probabilistic key generation algorithm Gen returns a secret/public key pair (sk, pk).

$k \leftarrow \mathsf{Sdk}(sk, pk)$**:** Given a secret key sk and a public key pk, the deterministic shared key establishment algorithm Sdk returns a shared key $k \in \mathcal{K}_{\mathsf{NIKE}}$, or a failure symbol $\perp$. We assume that Sdk always returns $\perp$ if sk is the secret key corresponding to pk.

A NIKE is δ_{NIKE} correct if for all $par \in \mathsf{Stp}$

$$\Pr\left[\mathsf{Sdk}(sk_1, pk_2) \neq \mathsf{Sdk}(sk_2, pk_1) \,\middle|\, \begin{matrix}(sk_1, pk_1) \xleftarrow{\$} \mathsf{Gen} \\ (sk_2, pk_2) \xleftarrow{\$} \mathsf{Gen}\end{matrix}\right] \leq \delta_{\mathsf{NIKE}}.$$

Security notions can be found in Appendix B.1.

2.5 Key Encapsulation Mechanism

Definition 5 (Key Encapsulation Mechanism). A *key encapsulation mechanism* KEM is defined as a tuple $\mathsf{KEM} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of the following algorithms.

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

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

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

The correctness error δ_{KEM} is defined as

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

Security notions can be found in Appendix B.2.

2.6 Ring Signatures

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

Definition 6 (Ring Signature). A *ring signature* scheme RSig is defined as a tuple $(\mathsf{Stp}, \mathsf{Gen}, \mathsf{Sgn}, \mathsf{Ver})$ of the following algorithms.

$par \xleftarrow{\$} \mathsf{Stp}(\kappa)$**:** Given an upper bound, κ, on the ring size 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 $k \leq \kappa$,

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

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 $\mu(sk) = pk$.

Security notions can be found in Appendix B.3.

2.7 Symmetric Encryption

Definition 7 (Symmetric Encryption). A *symmetric encryption* Sym is defined as a tuple $\mathsf{Sym} \coloneqq (\mathsf{Enc}, \mathsf{Dec})$ of the following algorithms and a key space $\mathcal{K}_{\mathsf{Sym}}$.

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

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

Sym is (perfectly) correct if for all $k \in \mathcal{K}_{\mathsf{Sym}}$ and all messages m it holds $m = \mathsf{Dec}(k, \mathsf{Enc}(k, m))$.

The security notion can be found in Appendix B.4

3 Generic Construction

In this section, we present a generic construction for a deniable AKEM combiner derived from two deniable AKEMs, AKEM_1 and AKEM_2, and a multi-keyed function H with five inputs which is illustrated in Figure 8. This construction builds upon the natural approach proposed in [GHP18]. Regarding security, our results are as follows: For confidentiality (see Theorem 9) and authenticity (see Theorem 10) the combiner requires only one of the underlying AKEMs to ensure confidentiality or authenticity, aligning with the expected behaviour of a combiner. However, for deniability, we prove that our generic black-box construction requires that both schemes be deniable. Specifically, Theorem 11 shows that if both schemes are *dishonest receiver* deniable, then the combiner inherits this property. Similarly, Theorem 12 establishes that the combiner maintains deniability in the *honest receiver* setting if both underlying schemes are honest receiver deniable.

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s, pk_r)}$	$\underline{\mathsf{Dec}(pk_s, sk_r, c)}$
01 $(sk_1, pk_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$	06 **parse** $sk_s \to (sk_1, sk_2)$	13 **parse** $pk_s \to (pk_1, pk_2)$
02 $(sk_2, pk_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$	07 **parse** $pk_r \to (pk_1, pk_2)$	14 **parse** $sk_r \to (sk_1, sk_2)$
03 $sk \coloneqq (sk_1, sk_2)$	08 $(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk_1, pk_1)$	15 **parse** $c \to (c_1, c_2)$
04 $pk \coloneqq (pk_1, pk_2)$	09 $(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk_2, pk_2)$	16 $k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk_1, sk_1, c_1)$
05 **return** (sk, pk)	10 $c \coloneqq (c_1, c_2)$	17 $k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk_2, sk_2, c_2)$
	11 $k \coloneqq \mathsf{H}(k_1, k_2, (\mu(sk_1), \mu(sk_2)), (pk_1, pk_2), c)$	18 $k \coloneqq \mathsf{H}(k_1, k_2, (pk_1, pk_2), (\mu(sk_1), \mu(sk_2)), c)$
	12 **return** (c, k)	19 **return** k

Figure 8. Generic Construction of a deniable authenticated key encapsulation mechanism $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$.

Lemma 8 (Correctness). *If* AKEM_1 *has correctness error* δ_1 *and* AKEM_2 *correctness error* δ_2*, then* $\delta_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}]} \leq \delta_1 + \delta_2$.

Theorem 9 (Confidentiality). *For any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* $\Pi \coloneqq \mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, depicted in Figure 8, there exists an* **Ins-CCA** *adversary* $\mathcal{B}_1$ *against* AKEM_1*, an* **Ins-CCA** *adversary* $\mathcal{B}_2$ *against* AKEM_2*, and a* **PRF** *adversary* $\mathcal{C}$ *against* H *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2} \approx t_{\mathcal{C}}$ *such that*

$$\mathrm{Adv}^{\text{Q-}\mathbf{Ins\text{-}CCA}}_{\Pi,\mathcal{A}} \leq 2 \cdot \min\left\{\mathrm{Adv}^{Q\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}_1,\mathcal{B}_1},\ \mathrm{Adv}^{Q\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}_2,\mathcal{B}_2}\right\} + 2 \cdot \mathrm{Adv}^{Q_{\mathsf{H}}\text{-}\mathbf{PRF}}_{\mathsf{H},\mathcal{C}} + Q_{\mathtt{Chl}} \cdot \delta_{\Pi}$$

for $Q = (n, Q_{\mathtt{Enc}}, Q_{\mathtt{Dec}}, Q_{\mathtt{Chl}}), Q_{\mathsf{H}} = (Q_{\mathtt{Chl}}, Q_{\mathtt{Dec}} + Q_{\mathtt{Chl}})$.

Proof (Sketch). If AKEM_1 or AKEM_2 is **Ins-CCA** secure, one of the input keys to H is uniformly random. With the **PRF** security of H, the key of the combiner is uniformly random too. The full proof can be found in Appendix C. ■

Theorem 10 (Authenticity). *For any* **Out-Aut** *adversary* $\mathcal{A}$ *against* $\Pi \coloneqq \mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, as depicted in Figure 8, there exists an* **Out-Aut** *adversary* $\mathcal{B}_1$ *against* AKEM_1*, an* **Out-Aut** *adversary* $\mathcal{B}_2$ *against* AKEM_2*, an* **Out-CCA** *adversary* $\mathcal{C}_1$ *against* AKEM_1*, an* **Out-CCA** *adversary* $\mathcal{C}_2$ *against* AKEM_2*, and a* **PRF** *adversary* $\mathcal{D}$ *against* H *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2} \approx t_{\mathcal{C}_1} \approx t_{\mathcal{C}_2} \approx t_{\mathcal{D}}$ *such that*

$$\begin{aligned}\mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}Aut}}_{\Pi,\mathcal{A}} \leq & 2 \cdot \min\left\{\mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}_1,\mathcal{B}_1} + \mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}CCA}}_{\mathsf{AKEM}_1,\mathcal{C}_1},\ \mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}_2,\mathcal{B}_2} + \mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}CCA}}_{\mathsf{AKEM}_2,\mathcal{C}_2}\right\} \\ & + 2 \cdot \mathrm{Adv}^{Q_{\mathsf{H}}\text{-}\mathbf{PRF}}_{\mathsf{H},\mathcal{D}} + Q_{\mathtt{Chl}} \cdot \delta_{\Pi}\end{aligned}$$

for $Q = (n, Q_{\mathtt{Enc}}, Q_{\mathtt{Chl}}), Q_{\mathsf{H}} = (Q_{\mathtt{Enc}} + Q_{\mathtt{Chl}}, Q_{\mathtt{Enc}} + Q_{\mathtt{Chl}})$.

Proof (Sketch). If AKEM_1 or AKEM_2 is **Out-Aut** secure, an adversary cannot construct a ciphertext for which they can distinguish a real decapsulation from a uniformly random key. With the **PRF** security of H, the key of the combiner should be uniformly random too. However, in contrast to the confidentiality case these properties are not enough because an adversary could use the encapsulation oracle to produce a ciphertext for which they know the key and then recycle one part of the ciphertext. To avoid this, we also require confidentiality (**Out-CCA**) of AKEM_1 or AKEM_2. The full proof can be found in Appendix C. ■

Theorem 11 (Dishonest Deniability). *For any* **DR-Den** *adversary* $\mathcal{A}$ *against* $\Pi \coloneqq \mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, as depicted in Figure 8, any simulators* $\mathsf{Sim}_1, \mathsf{Sim}_2$*, and simulator* $\mathsf{Sim}[\mathsf{Sim}_1, \mathsf{Sim}_2]$ *as defined in Figure 24 there exists a* **DR-Den** *adversary* $\mathcal{B}_1$ *against* AKEM_1 *and a* **DR-Den** *adversary* $\mathcal{B}_2$ *against* AKEM_2 *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2}$ *such that*

$$\mathrm{Adv}^{Q\text{-}\mathbf{DR\text{-}Den}}_{\Pi,\mathsf{Sim},\mathcal{A}} \leq 2 \cdot \mathrm{Adv}^{Q\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}_1,\mathsf{Sim}_1,\mathcal{B}_1} + 2 \cdot \mathrm{Adv}^{Q\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}_2,\mathsf{Sim}_2,\mathcal{B}_2}$$

for $Q = (n, Q_{\mathtt{Chl}})$.

Proof (Sketch). The simulator of the combiner can be constructed by using the simulators of the underlying schemes. For this reason the security relies on both the AKEM's. The full proof can be found in Appendix C. ■

Theorem 12 (Honest Deniability). *For any* **HR-Den** *adversary* $\mathcal{A}$ *against* $\Pi \coloneqq \mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, as depicted in Figure 8, any simulators* $\mathsf{Sim}_1, \mathsf{Sim}_2$*, and simulator* $\mathsf{Sim}[\mathsf{Sim}_1, \mathsf{Sim}_2]$ *as defined in Figure 24 there exists a* **HR-Den** *adversary* $\mathcal{B}_1$ *against* AKEM_1 *and a* **HR-Den** *adversary* $\mathcal{B}_2$ *against* AKEM_2 *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2}$ *such that*

$$\mathrm{Adv}^{Q\text{-}\mathbf{HR\text{-}Den}}_{\Pi,\mathsf{Sim},\mathcal{A}} \leq 2 \cdot \mathrm{Adv}^{Q\text{-}\mathbf{HR\text{-}Den}}_{\mathsf{AKEM}_1,\mathsf{Sim}_1,\mathcal{B}_1} + 2 \cdot \mathrm{Adv}^{Q\text{-}\mathbf{HR\text{-}Den}}_{\mathsf{AKEM}_2,\mathsf{Sim}_2,\mathcal{B}_2}$$

for $Q = (n, Q_{\mathit{Chl}})$.

Proof. The theorem can be proved analogously to Theorem 11. ■

4 Hybrid Construction: Shadowfax

In this section, we present a hybrid construction for a deniable AKEM based on a non-interactive key exchange NIKE, a key encapsulation mechanism KEM, a ring signature scheme RSig, a symmetric encryption scheme Sym, and two split-key PRFs H_1 and H_2 (H_1 having two inputs and H_2 having six inputs), as shown in Figure 9. This approach leverages well-known cryptographic primitives that can be instantiated from concrete schemes, providing a practical construction. Our security results are as follows: For both confidentiality (see Theorem 14) and authenticity (see Theorem 15), the combiner requires only one of the underlying AKEMs to ensure the respective property, consistent with the generic combiner. Confidentiality is provided by the security of either the ephemeral NIKE or the KEM. Authenticity comes from the static NIKE (providing implicit authentication) or the ring signature. For dishonest receiver deniability (see Theorem 16) we only rely on security advantages that can be instantiated with statistical security arguments, specifically the correctness property of the NIKE and the anonymity property of the ring signature. Finally, we achieve honest receiver deniability (see Theorem 17) by relying on just one of the underlying computational assumptions – specifically, the security of either the ephemeral NIKE or the KEM – to ensure deniability for the combiner. The main challenge arises from the public verifiability of the ring signature. [GJK24b] addresses this issue by symmetrically encrypting the ring signature using the KEM key. We implement a similar solution but derive the key material from *both* the NIKE and the KEM. This design mirrors our approach for confidentiality, ensuring that an adversary would need to compromise both the NIKE and KEM in order to verify the signature. Additionally H_1 is used twice in the construction to simplify the instantiation and used with a tag `"auth"` for domain separation in the proof. The setup of NIKE and RSig are implicitly done; for RSig by inputting maximum ring size 2.

Lemma 13 (Correctness). *If* NIKE *has correctness error* δ_{NIKE}*,* KEM *correctness error* δ_{KEM}*, and* RSig *correctness error* δ_{RSig} *and* Sym *is (perfectly) correct, then*

$$\delta_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{Sym},\mathsf{H}_1,\mathsf{H}_2]} \leq \delta_{\mathsf{NIKE}} + \delta_{\mathsf{KEM}} + \delta_{\mathsf{RSig}}.$$

Theorem 14 (Confidentiality). *For any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* $\Pi \coloneqq \mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 9, there exists a* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, a* **PRF** *adversary* $\mathcal{C}$ *against* H_1*, an* **PRF** *adversary* $\mathcal{D}$ *against* H_2*, and an* **IND-CCA** *adversary* $\mathcal{E}$ *against* KEM *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}} \approx t_{\mathcal{C}} \approx t_{\mathcal{D}} \approx t_{\mathcal{E}}$ *such that*

$$\begin{aligned}\mathrm{Adv}^{Q\text{-}\mathbf{Ins\text{-}CCA}}_{\Pi,\mathcal{A}} \leq 2nQ_{\mathit{Chl}} \cdot \Big(&\min\Big\{\mathrm{Adv}^{Q_{\mathsf{NIKE}}\text{-}\mathbf{CKS}}_{\mathsf{NIKE},\mathcal{B}} + \mathrm{Adv}^{(1,1)\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{C}}, \mathrm{Adv}^{(1,Q_{\mathit{Dec}},1)\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{E}}\Big\} \\ &+ 2 \cdot \mathrm{Adv}^{(1,Q_{\mathit{Dec}}+1)\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{D}} + (Q_{\mathit{Enc}} + Q_{\mathit{Dec}}) \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}} + Q_{\mathit{Chl}} \cdot \delta_{\Pi}\Big)\end{aligned}$$

for $Q = (n, Q_{\mathit{Enc}}Q_{\mathit{Dec}}, Q_{\mathit{Chl}}), Q_{\mathsf{NIKE}} = (Q_{\mathit{Enc}} + 2, 2Q_{\mathit{Enc}} + 2Q_{\mathit{Dec}}, 2Q_{\mathit{Enc}} + 2Q_{\mathit{Enc}} + 1)$.

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s, pk_r)}$	$\underline{\mathsf{Dec}(pk_s, sk_r, c)}$
01 $(nsk, npk) \xleftarrow{\$} \mathsf{NIKE.Gen}$	07 **parse** $sk_s \to (nsk_s, ksk_s, ssk_s)$	21 **parse** $pk_s \to (npk_s, kpk_s, spk_s)$
02 $(ksk, kpk) \xleftarrow{\$} \mathsf{KEM.Gen}$	08 **parse** $pk_r \to (npk_r, kpk_r, spk_r)$	22 **parse** $sk_r \to (nsk_r, ksk_r, ssk_r)$
03 $(ssk, spk) \xleftarrow{\$} \mathsf{RSig.Gen}$	09 $(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$	23 **parse** $c \to (npk_e, kct, sct)$
04 $sk \coloneqq (nsk, ksk, ssk)$	10 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_s, npk_r)$	24 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_s)$
05 $pk \coloneqq (npk, kpk, spk)$	11 $nk \coloneqq \mathsf{H}_1(nk', \texttt{"auth"})$	25 $nk \coloneqq \mathsf{H}_1(nk', \texttt{"auth"})$
06 **return** (sk, pk)	12 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk_r)$	26 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_e)$
	13 $(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$	27 $kk_1 \| kk_2 \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
	14 $m \leftarrow (kct, kpk_r)$	28 $k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
	15 $\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{\mu(ssk_s), spk_r\}, m)$	29 $\sigma \coloneqq \mathsf{Sym.Dec}(k', sct)$
	16 $k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$	30 $m \leftarrow (kct, \mu(ksk_r))$
	17 $sct \coloneqq \mathsf{Sym.Enc}(k', \sigma)$	31 **if** $\mathsf{RSig.Ver}(\sigma, \rho = \{spk_s, \mu(ssk_r)\}, m) \neq 1$
	18 $c \coloneqq (npk_e, kct, sct)$	32 **return** $\perp$
	19 $k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, \mu(sk_s), pk_r)$	33 $k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, \mu(sk_r))$
	20 **return** (c, k)	34 **return** k

Figure 9. Concrete construction of a deniable AKEM $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$. By "$\|$" we denote that an output is split into two equal parts.

Proof (Sketch). If the NIKE is **CKS** secure and H_1 a **PRF**, one of the keys for the split-key PRF H_2 is uniformly random and therefore the output of the hybrid AKEM construction. Analogously, the security can be based on the **IND-CCA** security of the KEM. In this case, another input key of H_2 is uniformly random and thus the hybrid's key. The full proof can be found in Appendix D. ■

Theorem 15 (Authenticity). *For any* **Out-Aut** *adversary* $\mathcal{A}$ *against* $\Pi \coloneqq \mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 9, there exists an* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, a* **PRF** *adversary* $\mathcal{C}$ *against* H_1*, an* **PRF** *adversary* $\mathcal{D}$ *against* H_2*, a* **UF-CRA1** *adversary* $\mathcal{E}$ *against* RSig*, and an* **IND-CCA** *adversary* $\mathcal{F}$ *against* KEM *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}} \approx t_{\mathcal{C}} \approx t_{\mathcal{D}} \approx t_{\mathcal{E}} \approx t_{\mathcal{F}}$ *such that*

$$\mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}Aut}}_{\Pi,\mathcal{A}} \leq \min\left\{2 \cdot \mathrm{Adv}^{Q_{\mathsf{NIKE}}\text{-}\mathbf{CKS}}_{\mathsf{NIKE},\mathcal{B}} + 2 \cdot \mathrm{Adv}^{(n^2,n^2)\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{C}},\ \mathrm{Adv}^{(n,2,Q_{Enc})\text{-}\mathbf{UF\text{-}CRA1}}_{\mathsf{RSig},\mathcal{E}} + 2 \cdot \mathrm{Adv}^{Q\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{F}} + Q^2_{Enc} \cdot \gamma_{\mathsf{KEM}}\right\}$$
$$+ 2 \cdot \mathrm{Adv}^{(Q',Q')\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{D}} + Q_{Chl} \cdot \delta_{\Pi} + Q_{Enc} \cdot Q' \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}}$$

with $Q = (n, Q_{Enc}, Q_{Chl}), Q_{\mathsf{NIKE}} = (Q_{Enc} + 2Q_{Chl}, Q_{Enc} + 2Q_{Chl}), Q' = Q_{Enc} + Q_{Chl}$.

Proof (Sketch). If the RSig is **UF-CRA1** it is hard for an adversary to come up with a valid ciphertext unless it was produced by the encapsulation oracle. This can be a valid attack since the adversary can recycle one part of the encapsulation output and thus produce a fresh and valid ciperhetxt. Additionally requiring the KEM to be **IND-CCA** prevents this attack. An analogous argument can be made if the NIKE is **CKS** secure. Note that a NIKE is used for authenticity (analogue to the **UF-CRA1** argument) and for confidentiality (analogue to the **IND-CCA** argument). Applying the **PRF** security of H_2 makes the hybrid key uniformly random. The full proof can be found in Appendix D. ■

Theorem 16 (Dishonest Deniability). *For any* **DR-Den** *adversary* $\mathcal{A}$ *against* $\Pi \coloneqq \mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 9, and simulator* Sim *as defined in Figure 33 there exists a* **MC-Ano** *adversary* $\mathcal{B}$ *against* RSig *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}}$ *such that*

$$\mathrm{Adv}^{(n,\mathrm{Q}_{Chl})\text{-}\mathbf{DR\text{-}Den}}_{\Pi,\mathsf{Sim},\mathcal{A}} \leq \mathrm{Adv}^{(n,2,\mathrm{Q}_{Chl})\text{-}\mathbf{MC\text{-}Ano}}_{\mathsf{RSig},\mathcal{B}} + Q_{Chl} \cdot \delta_{\mathsf{NIKE}}.$$

Proof (Sketch). To achieve dishonest receiver deniability, the simulator must create an indistinguishable NIKE and RSig part. For the NIKE, this reduces to the NIKE's correctness since the simulator has access to the receiver's secret key. For the RSig, this reduces to **MC-Ano**. Note that both properties must be fulfilled because distinguishing one part is sufficient for the adversary to win their game. The full proof can be found in Appendix D. ■

Theorem 17 (Honest Deniability). *For any* **HR-Den** *adversary* $\mathcal{A}$ *against* $\Pi := \mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 9,and simulator* Sim *as defined in Figure 35 there exists a* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, an* **IND-CPA** *adversary* $\mathcal{C}$ *against* KEM*,* **PRF** *adversaries* $\mathcal{D}$ *and* $\mathcal{E}$ *against* H_1 *and* H_2*, and a* **IND-CPA** *adversary* $\mathcal{F}$ *against* Sym *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}} \approx t_{\mathcal{C}} \approx t_{\mathcal{D}} \approx t_{\mathcal{E}} \approx t_{\mathcal{F}}$ *such that*

$$\mathrm{Adv}^{(n,\mathrm{Q}_{\mathcal{CH}})\text{-}\mathbf{HR\text{-}Den}}_{\Pi,\mathsf{Sim},\mathcal{A}} \leq 4n^2 \cdot Q_{\mathcal{CH}} \cdot \left(\min\left\{\mathrm{Adv}^{(2,0,1)\text{-}\mathbf{CKS}}_{\mathsf{NIKE},\mathcal{B}}, \mathrm{Adv}^{(1,1)\text{-}\mathbf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{C}}\right\} + \mathrm{Adv}^{(1,1)\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{D}} + \mathrm{Adv}^{(1,1)\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{E}} + \mathrm{Adv}^{\mathbf{IND\text{-}CPA}}_{\mathsf{Sym},\mathcal{F}}\right).$$

Proof (Sketch). Authenticity via a NIKE is not a problem for honest receiver deniability because authentication is made implicit and there is no information about it in the AKEM ciphertext. This is different from the ring signature which can be publicly verified which is why it is encrypted in the construction. Hence, security can be reduced to said encryption. Since the symmetric encryption key is derived from a NIKE key and a KEM key via a split-key PRF, we can reduce to **CKS** security of NIKE or **IND-CPA** of the KEM. Applying the **PRF** property of H_2 yields a uniformly random encryption key which is used to apply Sym's **IND-CPA** security. The full proof can be found in Appendix D. ■

5 Implementation

Table 1. Sizes (in bytes) of Gandalf ring signature instantiations, key-encapsulation mechanisms, the resulting deniable AKEMs, and Shadowfax hybrid instantiations derived from the initial AKEM. Sizes are from our implementation. Unless stated otherwise, "✓" denotes implementations provided in this work. The most compact instantiation and the one based on NIST standards are highlighted.

RSig				KEM				(PQ-)AKEM			NIKE			Shadowfax		
Scheme	Size		Impl.	Scheme	Size		Impl.	Size		Impl.	Scheme	Size	Impl.	Size		Impl.
	pk	σ			pk	c		pk	c			pk		pk	c	
Gandalf [Antrag, Mitaka]	896	1 276	✓	NTRU-A	768	768	✗	1 664	2 044	✗	Curve25519	32	✓ [Ber06]	1 696	2 076	✗
				BAT	521	473	✓ [FKPY22]	1 417	1 749	✓				1 449	1 781	✓
				ML–KEM	800	768	✓ [SAB+20]	1 696	2 044	✓				1 728	2 076	✓
Gandalf [Falcon, Falcon]	896	1 276	✓	NTRU-A	768	768	✗	1 664	2 044	✗				1 696	2 076	✗
				BAT	521	473	✓ [FKPY22]	1 417	1 749	✓				1 449	1 781	✓
				ML–KEM	800	768	✓ [SAB+20]	1 696	2 044	✓				1 728	2 076	✓

In this paper, we instantiate the Gandalf ring signature scheme, the corresponding post-quantum AKEM by [GJK24b], and the hybrid AKEM Shadowfax of this paper with several choices for the underlying post-quantum KEM and trapdoor sampler.

Table 2. Parameter sets for BAT, ML–KEM, and Falcon in this paper.

Scheme	BAT	ML–KEM	Falcon
Parameter set	`bat-257-512`	`mlkem-512`	`falcon-512`

Optimisation Goals. We aim for portability, the compactness of public key and ciphertext sizes, and compliance with standardised components in our instantiations of post-quantum and hybrid AKEMs. Since the rapid development of Post-Quantum Cryptography Standardisation by the National Institute of Standards and Technology (NIST), there are rich C reference implementations for several post-quantum cryptosystems.[9] We follow a similar paradigm and implement the AKEMs with the C programming

[9] Additionally, standardised building blocks often provide deeper analyses [ABB+23, AOB+24, BDK+18, GJK24a] providing more trust.

language[10]. Since C is a high-level programming language, our instantiations are portable. When compactness is the main optimisation goal, we choose BAT [FKPY22], Antrag [ENS+23], and Mitaka [EFG+22] along with the latest NTRU solver by [Por23]. If standardised components are preferred, we choose ML–KEM [MLK24] and the latest implementation of the fast Fourier sampler [DP16] from Falcon [PFH+20, Por25]. Table 2 summarises the chosen parameter sets for BAT, ML–KEM, and Falcon in this paper. Our instantiation is also modular and one can replace the post-quantum KEM and the trapdoor sampler with other combinations after some wrapping for the API. Table 1 summarises the schemes used to instantiate the Gandalf ring signature and the KEM, the implemented components, and the resulting sizes of public keys, signatures, and ciphertexts. The signature size of one instantiation is 40 bytes larger than in the original proposal [GJK24b], due to different compression techniques. The authors of [GJK24b] assumed the techniques of [ETWY22, EFG+22], but no implementations of those techniques were available. We therefore use the compression from the round-3 Falcon submission, where the signature has a fixed size of 666 bytes (including a 40-byte nonce). This results in a ring signature of 1 276 bytes with a 24-byte nonce for all Gandalf instantiations in this work. There is a concurrent work instantiating Gandalf [KNTW25]. Due to a different compression and a different salt size, they achieve slightly larger signatures of 1 288 bytes. All our source code is publicly available on the GitHub repository Shadowfax. [11] A comparison in security and size with different AKEMs from the literature is shown in Table 3.

5.1 Instantiation

Hash. Our instantiations use five distinct hash functions. `BLAKE2b` [SA15] which is shipped with BAT. `shake256` is used for converting the 32-byte shared key from ML–KEM to a 64-byte shared key. We use `shake128` [KjCP16] for hashing the message to a polynomial in the ring signature. Additionally, `SHA3-512` is used in the Non-Interactive Key Exchange (NIKE) construction, while `SHA3-256` is used for the hash functions H_1 and H_2 in the concrete construction (see Figure 9). Specifically, H_1 is implemented as the hmac `HMAC-SHA3-256` derived from `SHA3-256`. As for H_2, we implement it with three `HMAC-SHA3-256` calls as follows: $\mathsf{H}_2(nk, nk_2, kk_2, c, \mu(sk_s), pk_r) = \texttt{HMAC-SHA3-256}\,(nknk_2, \texttt{[rest]}_{kk_2})$ where

$$\begin{cases} nknk_2 = & \texttt{HMAC-SHA3-256}\,(nk, nk_2)\,, \\ \texttt{[rest]}_{kk_2} = & \texttt{HMAC-SHA3-256}\,(kk_2, \texttt{[rest]})\,, \end{cases}$$

and `[rest]` is the concatenation of the rest of the inputs. Note that our instantiations of H_1 and H_2 align with what we actually proved in Theorem 14. HMAC has been proven to be a dual-PRF [BBGS23] and the consecutive calls as described above instantiate a split-key PRF.

Symmetric Encryption, NIKE and KEM. We choose the CTR mode of `AES-128` for the symmetric encryption. For the NIKE, we choose the `Curve25519` Diffie-Hellman [Ber06] based on the `ref10` implementation of `crypto_scalarmult/curve25519` from `supercop-20240716` [DT24] and `SHA3-512`. After computing the raw Diffie-Hellman shared secret, we pass it through `SHA3-512` to derive the shared key for the NIKE. For the post-quantum KEM, we consider two options: ML–KEM [MLK24] and BAT [FKPY22]. For ML–KEM, we pass the 32-byte shared key to `shake256` and expand it to a 64-byte shared key. For BAT, we integrate the latest NTRU solver by [Por23], enforce the use of `BLAKE2b` in encapsulation and decapsulation, and simplify the source code with the C preprocessor.

NTRU Solver. In BAT and Antrag, we have to solve for polynomials $F, G \in \mathbb{Z}[X]/\langle X^N + 1\rangle$ satisfying the following NTRU equation: $g \cdot F - f \cdot G = q \bmod \left(X^N + 1\right)$ for a power-of-two N, a positive integer q, and polynomials $g, f \in \mathbb{Z}_q[X]/\langle X^N + 1\rangle$ with small coefficients. We integrate the latest NTRU solver by [Por23] into BAT and Antrag.

[10] The only exception is the reference implementation of Falcon: The latest source code by [Por25] comes with built-in selection for platform-specific intrinsics and falls back to C with integer emulation for the floating-point operations. On our platform, the source code automatically selects Armv8-A Neon intrinsics.

[11] https://github.com/vincentvbh/shadowfax

Table 3. Comparison of AKEMs, their security notions, and reliance on pre-quantum (pre-Q) or post-quantum (post-Q) assumptions. Deniability notions marked with "†" are not formally proven in the original works. Entries marked "⋆" indicate theoretically achievable sizes. The Swoosh [GdKQ+24] size refers to a passively secure NIKE; achieving active security requires a NIZK, which must be added to the NIKE public key. DualRing [YEL+21, Tab. 3] is included because the parameters of Gandalf would need to be slightly increased for stronger concrete anonymity (see [GJK24b] for details).

Scheme	Instantiation	Confidentiality	Authenticity	Deniability	Assumption		Size (in bytes)	
					pre-Q	post-Q	pk	c
DH-AKEM [ABH+21, Lst. 10]	X25519	**Ins-CCA**	**Out-Aut**	**DR-Den**†	✓	✗	32	32
EtSt-AKEM [AJKL23, Lst. 18]	BAT + Antrag	**Ins-CCA**	**Out-Aut**	✗	✗	✓	1 417	1 119
	ML–KEM + Falcon						1 697	1 434
NIKE-AKEM [AJKL23, Lst. 19]	Swoosh [GdKQ+24]	**Ins-CCA**	**Out-Aut**	**DR-Den**†	✗	✓	> 221 184	> 221 184
EaNtH-AKEM [JMOR25]	BAT + Swoosh [GdKQ+24]	**Ins-CCA**	**Out-Aut**	**DR-Den**†	✗	✓	> 221 705	473
FrodoKEX+ [CHN+24, Fig. 12]	N/A	**IND-1BatchCCA**	**UNF-1KCA**	**DR-Den**	✗	✓	21 300	72
Sparrow-KEM [Nio25, Fig. 7]	N/A	**IND-1BatchCCA**	**UNF-1KCA**	**DR-Den**	✗	✓	2 592	40
PQ-AKEM [GJK24b, Fig. 10]	NTRU-A + Gandalf [Antrag, Mitaka]	**Ins-CCA**	**Out-Aut**	**HR-Den** & **DR-Den**	✗	✓	1 664	2 004⋆/2 044
	BAT + Gandalf [Antrag, Mitaka]						1 417	1 709⋆/1 749
	BAT + DualRing						3 361	4 953
Shadowfax	X25519 + BAT + Gandalf [Antrag, Mitaka]	**Ins-CCA**	**Out-Aut**	**HR-Den** & **DR-Den**	✓	✓	1 449	1 741⋆/1 781
	X25519 + ML–KEM + Gandalf [Falcon,Falcon]						1 728	2 076

Ring Signature. We choose Gandalf [GJK24b] for the ring signature. According to [GJK24b], Gandalf achieves the smallest signature size for the ring of size 2, which suits well for constructing our AKEM. We consider two options: (i) components from Falcon and (ii) Antrag with Mitaka. For (i), we reorganise the construction of the signature generation and extract the trapdoor sampler. For (ii), we integrate the latest NTRU solver by [Por23] to the Antrag trapdoor generation [ENS+23] and outline below the necessary changes for achieving a compact signature size.

Modifications of Mitaka *implementation.* In the reference implementation of Mitaka released in [EFG+22], the signatures are stored as double-precision floating-point numbers with non-zero fractional parts, as opposed to integers. Therefore, existing compression techniques, which are defined over integers, cannot be straightforwardly deployed. Furthermore, there is no implementation for the latest compression technique [ETWY22] required by [EFG+22] and later used in [GJK24b]. Instead, we pull everything back to integers whenever the remaining computation can be defined entirely over $\mathbb{Z}$ and plug in the signature compression from the round 3 submission package of Falcon [PFH+20]. This results in a 40-byte increase of signature size compared to the original Gandalf by [GJK24b]. In the reference implementation of Mitaka, the program proceeds with double-precision floating-point arithmetic entirely, verifies the validity of signatures with double-precision floating-point arithmetic, and skips the signature compression. Finally, we also tweak the output of the sampler so it aligns with the definition of the trapdoor sampler. In the description of the Mitaka sampler, the output of the trapdoor sampler is negated and cannot be used directly in the ring signature scheme, as samples are supposed to be indistinguishable between parties. Therefore, we negate the output of the sampler.

5.2 Performance

Benchmarking Environment. We benchmark our portable implementations on the Firestorm core of an Apple M1 Pro with the operating system macOS Sonoma 14.6.1. Firestorm is the "big" core of the "big.LITTLE" computing architecture prevalent in Arm-based architecture in personal computing devices. It runs at the frequency of 3GHz and comes with a dedicated cryptographic extension. As we aim for portable implementations, we do not use the cryptographic extension. All programs are compiled with `GCC 13.3.0` with the optimisation flag `-O3`.

Cycle counts. Table 4 summarises the cycle counts of the C implementations of DH-AKEM, PQ-AKEM, and Shadowfax. For the key generations PQ-AKEM and Shadowfax, the cycle count is dominated by the

Table 4. Cycle counts (in thousands) of different authenticated key encapsulation mechanisms AKEM and ring signature schemes RSig run on a Firestorm core of an Apple M1 Pro running at 3GHz.

RSig	Unit	Gen	Sgn	Ver
Raptor [LAZ19, Zha20]	kcc	71 420	7 980	505
	ms	23.81	2.66	0.17
Gandalf [Falcon, Falcon]* [KNTW25]	kcc	-	-	-
	ms	5.20	0.49	0.02
Gandalf [Falcon, Falcon] (this work)	kcc	12 283	911	84
	ms	4.04	0.30	0.03
Gandalf [Antrag, Mitaka]	kcc	13 441	1 137	85
	ms	4.45	0.38	0.03
AKEM	**Unit**	**Gen**	**Enc**	**Dec**
DH-AKEM [X25519]	kcc	227	679	457
	ms	0.08	0.23	0.15
PQ-AKEM [BAT, Gandalf [Antrag, Mitaka]]	kcc	25 496	1 310	346
	ms	8.50	0.43	0.12
PQ-AKEM [ML–KEM, Gandalf [Falcon, Falcon]]	kcc	13 483	1 337	233
	ms	4.09	0.59	0.08
Shadowfax [X25519, BAT, Gandalf [Antrag, Mitaka]]	kcc	25 876	2 12	795
	ms	8.57	0.66	0.26
Shadowfax [X25519, ML–KEM, Gandalf [Falcon, Falcon]]	kcc	12 569	1 792	674
	ms	4.18	0.59	0.22

* Our benchmark. In [KNTW25], the authors accessed the timing counters through the standard C library on our platform. We benchmark their implementation on our platform. For other implementations, we access the cycle counters through the macOS API with a fallback to assembly for cycle counters on other operating systems. As mentioned before, the implementation of [KNTW25] leads to slightly larger signatures (1 288 bytes).

NTRU solver used in BAT, Mitaka, and Falcon. For the encapsulation, the cycle count is dominated by the signing of Gandalf. As for the decapsulation, the cycle count is dominated by the NIKE in Shadowfax and by ML–KEM/BAT in PQ-AKEM. We also give the cycle counts of our portable implementations of the ring signature Gandalf, and benchmark the C implementations of Raptor by [LAZ19] and the implementation by [KNTW25] on our platform. We stress that the C implementation of Raptor is based on an earlier implementation of Falcon, which had been significantly refactored after the publication of [LAZ19].

Conclusion. The dominant cost in terms of ciphertext size arises from the post-quantum ring signature, followed by the post-quantum KEM ciphertext. Public key sizes are less of a concern, and the overhead of the pre-quantum AKEM is minimal. Notably, this implies that with a post-quantum deniable AKEM, the cost of constructing a hybrid with strong security properties is virtually negligible. In conclusion, whenever a post-quantum AKEM is needed (regardless of whether it needs to be deniable), incorporating a hybrid should always be considered.

6 Acknowledgments

We thank the anonymous USENIX 2026 reviewers and Daniel Collins for their helpful feedback. Most of the work was done while Phillip Gajland was still affiliated with the Max Planck Institute for Security and Privacy, and Ruhr University Bochum. Phillip Gajland was 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).

References

[AAB+22] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions. (Cited on page 27.)

[ABB+23] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Peter Schwabe, Antoine Séré, and Pierre-Yves Strub. Formally verifying Kyber episode IV: Implementation correctness. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):164–193, 2023. doi:10.46586/tches.v2023.i3.164-193. (Cited on page 16.)

[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, Cham, Switzerland. doi:10.1007/978-3-030-77870-5_4. (Cited on pages 4, 6, 8, 18, and 44.)

[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, Singapore, Singapore. doi:10.1007/978-981-99-8736-8_11. (Cited on pages 4 and 18.)

[ANS23] ANSSI. Anssi views on the post-quantum cryptography transition (2023 follow up), 2023. URL: https://cyber.gouv.fr/sites/default/files/document/follow_up_position_paper_on_post_quantum_cryptography.pdf. (Cited on page 3.)

[AOB+24] José Bacelar Almeida, Santiago Arranz Olmos, Manuel Barbosa, Gilles Barthe, François Dupressoir, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Cameron Low, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Peter Schwabe, and Pierre-Yves Strub. Formally verifying kyber - episode V: Machine-checked IND-CCA security and correctness of ML-KEM in EasyCrypt. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part II*, volume 14921 of *Lecture Notes in Computer Science*, pages 384–421, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-68379-4_12. (Cited on page 16.)

[App24] Apple. iMessage with PQ3: The new state of the art in quantum-secure messaging at scale, February 2024. URL: https://security.apple.com/blog/imessage-pq3/. (Cited on page 3.)

[BBC+21] Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, Tung Chou, Tanja Lange, Michael Meyer, Benjamin Smith, and Jana Sotáková. CTIDH: faster constant-time CSIDH. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):351–387, 2021. URL: https://tches.iacr.org/index.php/TCHES/article/view/9069, doi:10.46586/tches.v2021.i4.351-387. (Cited on page 27.)

[BBCT22] Daniel J. Bernstein, Billy Bob Brumley, Ming-Shing Chen, and Nicola Tuveri. OpenSSLNTRU: Faster post-quantum TLS key exchange. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 845–862, Boston, MA, USA, August 10–12, 2022. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/bernstein. (Cited on page 3.)

[BBGS23] Matilda Backendal, Mihir Bellare, Felix Günther, and Matteo Scarlata. When messages are keys: Is HMAC a dual-PRF? In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 661–693, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-38548-3_22. (Cited on page 17.)

[BBLW22] Richard Barnes, Karthikeyan Bhargavan, Benjamin Lipp, and Christopher A. Wood. Hybrid Public Key Encryption. RFC 9180, February 2022. URL: https://www.rfc-editor.org/info/rfc9180, doi:10.17487/RFC9180. (Cited on pages 4, 27, and 28.)

[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. URL: https://www.rfc-editor.org/info/rfc9420, doi:10.17487/RFC9420. (Cited on page 4.)

[BCD+24] Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karoline Varner, and Bas Westerbaan. X-wing. *IACR Communications in Cryptology (CiC)*, 1(1):21, 2024. doi:10.62056/a3qj89n4e. (Cited on page 27.)

[BCNS15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society Press. doi:10.1109/SP.2015.40. (Cited on page 3.)

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy*, pages 353–367, London, United Kingdom, April 24–26, 2018. IEEE Computer Society Press. doi:10.1109/EuroSP.2018.00032. (Cited on page 16.)

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619, Santa Barbara, CA, USA, August 20–24, 2006. Springer Berlin Heidelberg, Germany. doi:10.1007/11818175_36. (Cited on page 9.)

[Bel15] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision resistance. *Journal of Cryptology*, 28(4):844–878, October 2015. doi:10.1007/s00145-014-9185-x. (Cited on page 9.)

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228, New York, NY, USA, April 24–26, 2006. Springer Berlin Heidelberg, Germany. doi:10.1007/11745853_14. (Cited on pages 16 and 17.)

[Beu22] Ward Beullens. Breaking rainbow takes a weekend on a laptop. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 464–479, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-15979-4_16. (Cited on page 3.)

[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, Cham, Switzerland. doi:10.1007/978-3-030-81652-0_16. (Cited on page 4.)

[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, Cham, Switzerland. doi:10.1007/978-3-030-97131-1_1. (Cited on page 30.)

[BHMS17] Nina Bindel, Udyani Herath, Matthew McKague, and Douglas Stebila. Transitioning to a quantum-resistant public key infrastructure. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 384–405, Utrecht, The Netherlands, June 26–28, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-59879-6_22. (Cited on page 27.)

[BLL24] Giacomo Borin, Yi-Fu Lai, and Antonin Leroux. Erebor and durian: Full anonymous ring signatures from quaternions and isogenies. Cryptology ePrint Archive, Report 2024/1185, 2024. URL: https://eprint.iacr.org/2024/1185. (Cited on page 5.)

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer Berlin Heidelberg, Germany. doi:10.1007/11761679_25. (Cited on pages 7 and 8.)

[BS20] Mihir Bellare and Igors Stepanovs. Security under message-derived keys: Signcryption in iMessage. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 507–537, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-45727-3_17. (Cited on page 4.)

[BSI22] BSI. Quantum-safe cryptography – fundamentals, current developments and recommendations, 2022. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.pdf. (Cited on page 3.)

[BSI24] BSI. Cryptographic mechanisms: Recommendations and key lengths - bsi tr-02102-1, 2024. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf. (Cited on page 3.)

[CCH23] Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. Real world deniability in messaging. Real World Crypto Symposium, 2023. https://www.youtube.com/watch?v=sthXs4zJ5XU&t=5504s. (Cited on page 27.)

[CCH25] Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. Real world deniability in messaging. *Proceedings on Privacy Enhancing Technologies*, 2025. URL: https://eprint.iacr.org/2023/403. (Cited on page 27.)

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-30589-4_15. (Cited on page 3.)

[CHH+25] Deirdre Connolly, Kathrin Hövelmanns, Andreas Hülsing, Stavros Kousidis, and Matthias Meijers. Starfighters — on the general applicability of x-wing. Cryptology ePrint Archive, Paper 2025/1397, 2025. URL: https://eprint.iacr.org/2025/1397. (Cited on page 27.)

[CHN+24] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. K-waay: Fast and deniable post-quantum X3DH without ring signatures. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024: 33rd USENIX Security Symposium*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/collins. (Cited on pages 4 and 18.)

[CKS09] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. *Journal of Cryptology*, 22(4):470–504, October 2009. doi:10.1007/s00145-009-9041-6. (Cited on page 28.)

[CPS19] Eric Crockett, Christian Paquin, and Douglas Stebila. Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. Cryptology ePrint Archive, Report 2019/858, 2019. URL: https://eprint.iacr.org/2019/858. (Cited on page 27.)

[DA99] Tim Dierks and Christopher Allen. *RFC 2246 - The TLS Protocol Version 1.0*. Internet Activities Board, January 1999. (Cited on page 3.)

[DP16] Léo Ducas and Thomas Prest. Fast Fourier Orthogonalization. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, pages 191–198, 2016. URL: https://link.springer.com/chapter/10.1007/978-3-031-15777-6_7. (Cited on page 17.)

[DT24] D.J.Bernstein and T.Lange. ebacs:ecrypt benchmarking of cryptographic systems, 2024. accessed 16 July 2024. URL: https://bench.cr.yp.to. (Cited on page 17.)

[DZ10] Alexander W. Dent and Yuliang Zheng, editors. *Practical Signcryption*. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-540-89411-7. (Cited on page 4.)

[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, Cham, Switzerland. doi:10.1007/978-3-031-07082-2_9. (Cited on pages 17 and 18.)

[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, Singapore, Singapore. doi:10.1007/978-981-99-8739-9_1. (Cited on pages 17 and 18.)

[ETS20] ETSI. CYBER; quantum-safe cryptography (QSC); quantum-safe hybrid key exchanges. Standard, European Telecommunications Standards Institute (ETSI), December 2020. URL: https://www.etsi.org/deliver/etsi_ts/103700_103799/103744/01.01.01_60/ts_103744v010101p.pdf. (Cited on page 27.)

[ETWY22] Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Shorter Hash-and-Sign Lattice-Based Signatures. In *Annual International Cryptology Conference*, pages 245–275. Springer, 2022. (Cited on pages 17 and 18.)

[FH25] Sebastian Faller and Julia Hesse. How to (not) combine oblivious pseudorandom functions. Cryptology ePrint Archive, Paper 2025/1084, 2025. URL: https://eprint.iacr.org/2025/1084. (Cited on page 4.)

[FHKP12] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. Cryptology ePrint Archive, Paper 2012/732/20130101:143205, 2012. URL: https://eprint.iacr.org/archive/2012/732/20130101:143205. (Cited on pages 11 and 28.)

[FJ24] Rune Fiedler and Christian Janson. A deniability analysis of Signal's initial handshake PQXDH. *Proceedings on Privacy Enhancing Technologies*, 2024(4):907–928, October 2024. doi:10.56553/popets-2024-0148. (Cited on pages 3 and 28.)

[FKPY22] Pierre-Alain Fouque, Paul Kirchner, Thomas Pornin, and Yang Yu. BAT: Small and fast KEM over NTRU lattices. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(2):240–265, 2022. doi:10.46586/tches.v2022.i2.240-265. (Cited on pages 16 and 17.)

[GdKQ+24] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. SWOOSH: Efficient lattice-based non-interactive key exchange. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024: 33rd USENIX Security Symposium*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/gajland. (Cited on pages 18 and 27.)

[GHH+24] Sharon Goldberg, Miro Haller, Nadia Heninger, Mike Milano, Dan Shumow, Marc Stevens, and Adam Suhl. RADIUS/UDP considered harmful. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024: 33rd USENIX Security Symposium*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/goldberg. (Cited on page 3.)

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 190–218, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-76578-5_7. (Cited on pages 5, 9, 12, and 27.)

[GJK24a] Phillip Gajland, Jonas Janneck, and Eike Kiltz. A closer look at falcon. Cryptology ePrint Archive, Report 2024/1769, 2024. URL: https://eprint.iacr.org/2024/1769. (Cited on page 16.)

[GJK24b] Phillip Gajland, Jonas Janneck, and Eike Kiltz. Ring signatures for deniable AKEM: Gandalf's fellowship. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 305–338, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-68376-3_10. (Cited on pages 4, 5, 7, 8, 14, 16, 17, 18, 27, 29, and 30.)

[GRSV25] Felix Günther, Michael Rosenberg, Douglas Stebila, and Shannon Veitch. Hybrid obfuscated key exchange and KEMs. Cryptology ePrint Archive, Report 2025/408, 2025. URL: https://eprint.iacr.org/2025/408. (Cited on pages 4 and 27.)

[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. doi:10.1007/s00145-022-09427-1. (Cited on page 4.)

[HR25] Julia Hesse and Michael Rosenberg. PAKE combiners and efficient post-quantum instantiations. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 395–420, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-031-91124-8_14. (Cited on pages 4 and 27.)

[HV21] Loïs Huguenin-Dumittan and Serge Vaudenay. FO-like combiners and hybrid post-quantum cryptography. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *CANS 21: 20th International Conference on Cryptology and Network Security*, volume 13099 of *Lecture Notes in Computer Science*, pages 225–244, Vienna, Austria, December 13–15, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-92548-2_12. (Cited on page 27.)

[Jan25] Jonas Janneck. Bird of prey: Practical signature combiners preserving strong unforgeability. Cryptology ePrint Archive, Paper 2025/1844, 2025. URL: https://eprint.iacr.org/2025/1844. (Cited on pages 4 and 27.)

[JMOR25] Jonas Janneck, Jonas Meers, Massimo Ostuzzi, and Doreen Riepel. Snake mackerel: An isogeny-based AKEM leveraging randomness reuse. Cryptology ePrint Archive, Paper 2025/1474, 2025. URL: https://eprint.iacr.org/2025/1474. (Cited on pages 4 and 18.)

[KjCP16] John Kelsey, Shu jen Change, and Ray Perlner. SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash. Technical report, National Institute of Standards and Technology, December 2016. doi:10.6028/nist.sp.800-185. (Cited on page 17.)

[KNTW25] Shuichi Katsumata, Guilhem Niot, Ida Tucker, and Thom Wiggers. Comprehensive deniability analysis of signal handshake protocols: X3DH, PQXDH to fully post-quantum with deniable ring signatures. Cryptology ePrint Archive, Paper 2025/1090, 2025. To appear in USENIX Security 2025: 34th USENIX Security Symposium. URL: https://eprint.iacr.org/2025/1090. (Cited on pages 7, 17, and 19.)

[KS24] Ehren Kret and Rolfe Schmidt. The pqxdh key agreement protocol, 2024. URL: https://signal.org/docs/specifications/pqxdh/pqxdh.pdf. (Cited on pages 3 and 28.)

[KSL+19] Krzysztof Kwiatkowski, Nick Sullivan, Adam Langley, Dave Levin, and Alan Mislove. Measuring tls key exchange with post-quantum kem, 2019. URL: https://csrc.nist.gov/CSRC/media/Events/Second-PQC-Standardization-Conference/documents/accepted-papers/kwiatkowski-measuring-tls.pdf. (Cited on page 27.)

[KV19] Kris Kwiatkowski and Luke Valenta. The TLS post-quantum experiment. Post on the Cloudflare blog, 2019. https://blog.cloudflare.com/the-tls-post-quantum-experiment/. (Cited on page 3.)

[KW16] Hugo Krawczyk and Hoeteck Wee. The OPTLS protocol and TLS 1.3. In *2016 IEEE European Symposium on Security and Privacy*, pages 81–96, Saarbrücken, Germany, March 21–24, 2016. IEEE Computer Society Press. doi:10.1109/EuroSP.2016.18. (Cited on page 27.)

[Lan16] Adam Langley. CECPQ1 results. Blog post, 2016. https://www.imperialviolet.org/2016/11/28/cecpq1.html. (Cited on page 3.)

[Lan18] Adam Langley. CECPQ2. Blog post, 2018. https://www.imperialviolet.org/2018/12/12/cecpq2.html. (Cited on page 3.)

[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 2019: 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, Cham, Switzerland. doi:10.1007/978-3-030-21568-2_6. (Cited on page 19.)

[LHT16] Adam Langley, Mike Hamburg, and Sean Turner. Elliptic Curves for Security. RFC 7748, January 2016. URL: https://www.rfc-editor.org/info/rfc7748, doi:10.17487/RFC7748. (Cited on page 27.)

[LL25] You Lyu and Shengli Liu. Hybrid password authentication key exchange in the UC framework. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 421–450, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-031-91124-8_15. (Cited on pages 4 and 27.)

[LLJ+19] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. LAC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions. (Cited on page 27.)

[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, Cham, Switzerland. doi:10.1007/978-3-030-84245-1_21. (Cited on page 5.)

[LSB24] Felix Linker, Ralf Sasse, and David Basin. A formal analysis of apple's iMessage PQ3 protocol. Cryptology ePrint Archive, Paper 2024/1395, 2024. URL: https://eprint.iacr.org/2024/1395. (Cited on pages 3 and 28.)

[MLD24] Module-lattice-based digital signature standard. National Institute of Standards and Technology NIST FIPS PUB 204, U.S. Department of Commerce, August 2024. URL: http://dx.doi.org/10.6028/NIST.FIPS.204, doi:10.6028/nist.fips.204. (Cited on page 3.)

[MLK24] Module-lattice-based key-encapsulation mechanism standard. National Institute of Standards and Technology NIST FIPS PUB 203, U.S. Department of Commerce, August 2024. URL: http://dx.doi.org/10.6028/NIST.FIPS.203, doi:10.6028/nist.fips.203. (Cited on pages 3 and 17.)

[MMP+23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 448–471, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-30589-4_16. (Cited on page 3.)

[MP16] Moxie Marlinspike and Trevor Perrin. The x3dh key agreement protocol, 2016. URL: https://signal.org/docs/specifications/x3dh/x3dh.pdf. (Cited on pages 3, 27, and 28.)

[Nio25] Guilhem Niot. Practical deniable post-quantum X3DH: A lightweight split-KEM for k-waay. In *ASIACCS 2025*. ACM Press, 2025. (Cited on pages 4 and 18.)

[NIS16] NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, 2016. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf. (Cited on page 3.)

[OGP+24] Mike Ounsworth, John Gray, Massimiliano Pala, Jan Klaußner, and Scott Fluhrer. Composite ML-DSA for use in Internet PKI. Internet-Draft draft-ietf-lamps-pq-composite-sigs-02, Internet Engineering Task Force, July 2024. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ietf-lamps-pq-composite-sigs/02/. (Cited on page 27.)

[OP19] David Ott and Christopher Peikert. Identifying research challenges in post quantum cryptography migration and cryptographic agility, 2019. URL: https://arxiv.org/abs/1909.07353, arXiv:1909.07353. (Cited on page 3.)

[PFH+20] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. Submission to the NIST Post-Quantum Cryptography Standardization Project, 2020. URL: https://falcon-sign.info/. (Cited on pages 17 and 18.)

[Por23] Thomas Pornin. Improved key pair generation for falcon, BAT and hawk. Cryptology ePrint Archive, Report 2023/290, 2023. URL: https://eprint.iacr.org/2023/290. (Cited on pages 17 and 18.)

[Por25] Thomas Pornin. Falcon on ARM cortex-m4: an update. Cryptology ePrint Archive, Paper 2025/123, 2025. URL: https://eprint.iacr.org/2025/123. (Cited on page 17.)

[PST20] Christian Paquin, Douglas Stebila, and Goutam Tamvada. Benchmarking post-quantum cryptography in TLS. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 72–91, Paris, France, April 15–17, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-44223-1_5. (Cited on pages 3 and 27.)

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018. URL: https://www.rfc-editor.org/info/rfc8446, doi:10.17487/RFC8446. (Cited on page 3.)

[Riv92] Ronald L. Rivest. *RFC 1321: The MD5 Message-Digest Algorithm*. Internet Activities Board, April 1992. (Cited on page 3.)

[RMA+23] Nathan Reitinger, Nathan Malkin, Omer Akgul, Michelle L. Mazurek, and Ian Miers. Is cryptographic deniability sufficient? non-expert perceptions of deniability in secure messaging. In *2023 IEEE Symposium on Security and Privacy*, pages 274–292, San Francisco, CA, USA, May 21–25, 2023. IEEE Computer Society Press. doi:10.1109/SP46215.2023.10179361. (Cited on page 27.)

[Rob23] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 472–503, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-30589-4_17. (Cited on page 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 Berlin Heidelberg, Germany. doi:10.1007/3-540-45682-1_32. (Cited on page 12.)

[RYAJ+24] Anamika Rajendran, Tarun Kumar Yadav, Malek Al-Jbour, Francisco Manuel Mares Solano, Kent Seamons, and Joshua Reynolds. Deniable encrypted messaging: User understanding after hands-on social experience. In *Proceedings of the 2024 European Symposium on Usable Security*, EuroUSEC '24, page 155–171, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3688459.3688479. (Cited on page 27.)

[SA15] Markku-Juhani O. Saarinen and Jean-Philippe Aumasson. The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC). RFC 7693, November 2015. URL: https://www.rfc-editor.org/info/rfc7693, doi:10.17487/RFC7693. (Cited on page 17.)

[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. (Cited on page 16.)

[SFG24] Douglas Stebila, Scott Fluhrer, and Shay Gueron. Hybrid key exchange in TLS 1.3. Internet-Draft draft-ietf-tls-hybrid-design-10, Internet Engineering Task Force, April 2024. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/10/. (Cited on page 27.)

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press. doi:10.1109/SFCS.1994.365700. (Cited on page 3.)

[SLH24] Stateless hash-based digital signature standard. National Institute of Standards and Technology NIST FIPS PUB 205, U.S. Department of Commerce, August 2024. URL: http://dx.doi.org/10.6028/NIST.FIPS.205, doi:10.6028/nist.fips.205. (Cited on page 3.)

[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. doi:10.1145/3372297.3423350. (Cited on page 27.)

[Ste24] Douglas Stebila. Security analysis of the iMessage PQ3 protocol. Cryptology ePrint Archive, Report 2024/357, 2024. URL: https://eprint.iacr.org/2024/357. (Cited on pages 3 and 28.)

[TTB+23] C. Tjhai, M. Tomlinson, G. Bartlett, Scott Fluhrer, Daniel Van Geest, Oscar Garcia-Morchon, and Valery Smyslov. Multiple Key Exchanges in the Internet Key Exchange Protocol Version 2 (IKEv2). RFC 9370, May 2023. URL: https://www.rfc-editor.org/info/rfc9370, doi:10.17487/RFC9370. (Cited on page 27.)

[WFLY04] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199, 2004. URL: https://eprint.iacr.org/2004/199. (Cited on page 3.)

[WR19] Bas Westerbaan and Cefan Daniel Rubin. Defending against future threats: Cloudflare goes post-quantum. Post on the Cloudflare blog, 2019. https://blog.cloudflare.com/post-quantum-for-all/. (Cited on page 3.)

[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, Cham, Switzerland. doi:10.1007/978-3-030-84242-0_10. (Cited on page 18.)

[YGS23] Tarun Kumar Yadav, Devashish Gosain, and Kent E. Seamons. Cryptographic deniability: A multi-perspective study of user perceptions and expectations. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023: 32nd USENIX Security Symposium*, pages 3637–3654, Anaheim, CA, USA, August 9–11, 2023. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/yadav. (Cited on page 27.)

[Zha20] Zhenfei Zhang. Raptor. https://github.com/zhenfeizhang/raptor, 2020. (Cited on page 19.)

A Additional Related Work

A.1 Combiners

Confidentiality. Hybrid KEMs have been explored as a means to achieve *confidentiality* in the post-quantum era. For instance, [GHP18] demonstrated that the simple KEM combiner $\mathsf{H}(k_1, k_2)$ does not provide CCA security, whereas incorporating the ciphertexts as $\mathsf{H}(k_1, k_2, c_1, c_2)$ resolves this issue. Furthermore, [HV21] demonstrated a hybrid KEM combining the CPA-secure versions of HQC [AAB+22] and LAC [LLJ+19] achieving IND-CCA security. Industry leaders have also explored hybrid approaches. In 2019 Cloudflare and Google conducted experiments [KSL+19] to assess the performance of hybrid cryptographic solutions in real-world scenarios. This work led to the adoption of hybrid cryptography in platforms such as Amazon's s2n and various forks of OpenSSL and OpenSSH [CPS19]. Further investigations into post-quantum hybrid cryptography include benchmarks for its application in TLS [PST20], underscoring industry's intent to integrate these solutions. The European Telecommunications Standards Institute (ETSI) has also formalised quantum-safe hybrid key exchanges [ETS20], while the TLS protocol is exploring hybrid key exchange designs using concatenated key derivation functions [SFG24]. Additionally, the Internet Key Exchange (IKE) protocol is evolving to incorporate hybrid post-quantum cryptographic methods [TTB+23]. A recent concrete hybrid KEM, X-Wing [BCD+24], combines X25519 [LHT16] and ML-KEM-768, though it is a specific instantiation rather than a generic combiner as in [GHP18]. The construction was later generalized in [CHH+25].

Authenticity. Hybrid solutions for *authenticity*, such as combining digital signature schemes have also been explored [BHMS17, OGP+24, Jan25]. A natural way is to concatenate signatures, accepting the result as valid only if all signatures are valid. This achieves existential unforgeability under a chosen message attack (EUF-CMA) but not *strong* existential unforgeability. A hybrid solution for strong unforgeability was recently presented in [Jan25]. The works of [BHMS17, OGP+24, Jan25] examined hybrid digital signatures within a public key infrastructure. In particular, [BHMS17] introduced the concept of *non-separability*, ensuring that a signature in a combined scheme cannot be split into valid signatures for either of its individual components.

Other Notions. A concurrent study on PAKE combiners leverages statistical password hiding to circumvent the difficulty of constructing a generic combiner [HR25, LL25]. Another concurrent work explores obfuscated KEMs and constructs combiners, similarly relying on *statistical* ciphertext uniformity [GRSV25].

A.2 Deniability

Despite limited awareness of deniability's benefits among non-experts [RMA+23, YGS23], recent work has focused on improving the real-world deniability of protocols [RMA+23, RYAJ+24, CCH25, CCH23] particularly in messaging systems. While screenshots of message transcripts have traditionally been used as legal evidence, [CCH25, CCH23] proposed enabling message editing at the application level, enhancing real-world deniability. However, such solutions still rely on underlying cryptographic deniability to be effective. Many protocols, such as X3DH [MP16], provide deniability by design, while others, like certain versions of the Hybrid Public Key Encryption (HPKE) [BBLW22] standard, have deniability *accidentally* as a relic of using Diffie-Hellman for implicit authentication. As noted in [GJK24b], the authenticated mode of HPKE [BBLW22] exhibits some deniability properties as an unintended consequence of this design choice. Another example is OPTLS by Krawczyk and Wee [KW16], a proposal that eliminates the need for handshake signatures in TLS. Such protocols typically use X25519 [LHT16] in practice, and can be upgraded to the post-quantum setting with a post-quantum non-interactive key exchange (NIKE). However, existing post-quantum NIKEs are limited by prohibitively large public keys [GdKQ+24] or slow performance [BBC+21]. As a result, most protocols instead tend to rely on post-quantum KEMs and/or standard post-quantum signature schemes. For example, KEMTLS [SSW20] eliminates the need for handshake signatures like OPTLS, but it uses static KEM keys for authentication, which differs from the ephemeral key approach of protocols like X3DH. This presents a dilemma: while post-quantum security is achievable, many protocols lose additional security properties – such as deniability – provided by their

classical counterparts. For instance, Signal's new Post-Quantum Extended Diffie-Hellman (PQXDH) protocol [KS24] combines classical and post-quantum cryptography. However, PQXDH does not satisfy the same level of deniability as its predecessor, X3DH [MP16], due to the signature on the ephemeral key [FJ24]. Similarly, the analysis of Apple's iMessage with PQ3 [Ste24, LSB24], explicitly states that deniability is not a design goal. We hypothesise that this omission stems from the cost of providing deniability or the relative simplicity of omitting it in favour of other security priorities. Moreover, a likely approach to migrating authenticated HPKE [BBLW22] to the post-quantum setting would likely involve using a KEM and signatures for explicit authentication, which would eliminate the deniability properties present in its pre-quantum counterpart.

B Additional Preliminaries

B.1 Non-Interactive Key Exchange (NIKE)

We formalise the notion of key indistinguishability with *active security* for a simplified non-interactive key exchange NIKE, with respect to system parameters $par \in \sup(\mathsf{Stp})$ via the game $(n, Q_{\mathtt{RC}}, Q_{\mathtt{T}})$-$\mathbf{CKS}_{\mathsf{NIKE},par}(\mathcal{A})$ depicted in Figure 10 and define the advantage of adversary $\mathcal{A}$ as

$$\mathrm{Adv}^{(n,Q_{\mathtt{RC}},Q_{\mathtt{T}})\text{-}\mathbf{CKS}}_{\mathsf{NIKE},par}(\mathcal{A}) := \left|\Pr\left[(n, Q_{\mathtt{RC}}, Q_{\mathtt{T}})\text{-}\mathbf{CKS}_{\mathsf{NIKE},par}(\mathcal{A}) \Rightarrow 1\right] - \tfrac{1}{2}\right|.$$

Note that this is an abstraction of the model equivalent to the original CKS [CKS09] notion for *simplified NIKEs* from [FHKP12, App. G]. We reduce the number of oracles to a minimum. Instead of the register honest oracles, we provide the adversary with n honestly generated public keys in the beginning. Instead of registering corrupted users and querying to a corrupt reveal oracle, we directly provide the corrupt reveal oracle on an adversarially chosen (corrupted) public key. This matches the interface of notions for other primitives much better and eases the presentation of the proofs.

Lemma 18. Definition **CKS** is equivalent to the original definition (**CKS-Orig**). In particular for any adversary $\mathcal{A}$ against one of the notions there exists an adversary $\mathcal{B}$ against the other notion such that

$$\mathrm{Adv}^{(n,Q_{\mathtt{RC}},Q_{\mathtt{T}})\text{-}\mathbf{CKS}}_{\mathsf{NIKE},par,\mathcal{A}} \leq \mathrm{Adv}^{(n,Q_{\mathtt{RC}},Q_{\mathtt{RC}},Q_{\mathtt{T}})\text{-}\mathbf{CKS\text{-}Orig}}_{\mathsf{NIKE},par,\mathcal{B}},$$
$$\mathrm{Adv}^{(Q_{\mathtt{RHU}},Q_{\mathtt{RCU}},Q_{\mathtt{RC}},Q_{\mathtt{T}})\text{-}\mathbf{CKS\text{-}Orig}}_{\mathsf{NIKE},par,\mathcal{A}} \leq \mathrm{Adv}^{(Q_{\mathtt{RHU}},Q_{\mathtt{RC}},Q_{\mathtt{T}})\text{-}\mathbf{CKS}}_{\mathsf{NIKE},par,\mathcal{B}}.$$

Games $(n, Q_{\mathtt{RC}}, Q_{\mathtt{T}})$-$\mathbf{CKS}_{\mathsf{NIKE},par}(\mathcal{A})$

```
01 for i ∈ [n]
02   (sk_i, pk_i) ←$ Gen
03 b ←$ {0,1}
04 D := ∅
05 b' ← A^{RevCor,Test,Ext,RevHon}(pk_1, ..., pk_n)
06 return [[b = b']]
```

Oracle $\mathtt{RevCor}(i \in [n], pk \notin \{pk_1, \ldots, pk_n\})$

```
07 k ← Sdk(sk_i, pk)
08 return k
```

Oracle $\mathtt{Test}(i \in [n], j \in [n])$

```
09 if i = j return ⊥
10 if b = 0
11   k ← Sdk(sk_i, pk_j)
12 if b = 1
13   if ∃ k' : ({pk_i, pk_j}, k') ∈ D
14     k ← k'
15   else
16     k ←$ K
17     D ← D ∪ {({pk_i, pk_j}, k)}
18 return k
```

Figure 10. Games defining **CKS** for a simplified non-interactive key exchange NIKE with adversary $\mathcal{A}$ making at most $Q_{\mathtt{RC}}$ queries to `RevCor` and at most $Q_{\mathtt{T}}$ queries to `Test`.

Proof. The reduction for the first inequality is depicted in Figure 11, the reduction for the second in Figure 12. ■

$\mathcal{B}^{\mathtt{RegHonUsr}_{\mathcal{B}},\mathtt{RegCorUsr}_{\mathcal{B}},\mathtt{RevCor}_{\mathcal{B}},\mathtt{Test}_{\mathcal{B}}}$

01 **for** $i \in [n]$
02 $\quad pk_i \xleftarrow{\$} \mathtt{RegHonUsr}_{\mathcal{B}}()$
03 $\mathcal{C} \coloneqq \emptyset$
04 $b' \leftarrow \mathcal{A}^{\mathtt{RevCor},\mathtt{Test}}(pk_1, \ldots, pk_n)$
05 **return** b'

Oracle $\mathtt{RevCor}(i \in [n], pk \notin \{pk_1, \ldots, pk_n\})$

06 **if** $pk \notin \mathcal{C}$
07 $\quad \mathtt{RegCorUsr}_{\mathcal{B}}(pk)$
08 $\quad \mathcal{C} \coloneqq \mathcal{C} \cup \{pk\}$
09 $k \xleftarrow{\$} \mathtt{RevCor}_{\mathcal{B}}(pk_i, pk)$
10 **return** k

Oracle $\mathtt{Test}(i \in [n], j \in [n])$

11 **if** $i = j$ **return** $\bot$
12 $k \xleftarrow{\$} \mathtt{Test}_{\mathcal{B}}(pk_i, pk_j)$
13 **return** k

Figure 11. Reduction for **CKS-Orig** $\Rightarrow$ **CKS**.

$\mathcal{B}^{\mathtt{RevCor}_{\mathcal{B}},\mathtt{Test}_{\mathcal{B}}}(pk_1, \ldots, pk_n)$

01 $\mathcal{C} \coloneqq \emptyset$
02 $i \coloneqq 0$
03 $b' \xleftarrow{\$} \mathcal{A}^{\mathtt{RegHonUsr},\mathtt{RegCorUsr},\mathtt{RevCor},\mathtt{Test}}$
04 **return** b'

Oracle $\mathtt{RegHonUsr}()$

05 $i \coloneqq i + 1$
06 **return** pk_i

Oracle $\mathtt{RegCorUsr}(pk)$

07 $\mathcal{C} \coloneqq \mathcal{C} \cup \{pk\}$

Oracle $\mathtt{RevCor}(pk, pk')$

08 **if** $\exists\, j : pk = pk_j \wedge pk' \in \mathcal{C}$
09 $\quad k \xleftarrow{\$} \mathtt{RevCor}_{\mathcal{B}}(j, pk')$
10 $\quad$ **return** k
11 **return** $\bot$

Oracle $\mathtt{Test}(pk, pk')$

12 **if** $\exists\, j, j' : pk = pk_j \wedge pk' = pk_{j'}$
13 $\quad k \xleftarrow{\$} \mathtt{Test}_{\mathcal{B}}(j, j')$
14 $\quad$ **return** k
15 **return** $\bot$

Figure 12. Reduction for **CKS** $\Rightarrow$ **CKS-Orig**.

B.2 Key Encapsulation Mechanism

The γ-spreadness of a KEM is defined as

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

where $\mathcal{C}$ denotes the ciphertext space.

We formalise the notion of ciphertext indistinguishability (**IND-CCA** and **IND-CPA**) for a key encapsulation mechanism KEM via the game $(n, Q_{\mathtt{Dec}}, Q_{\mathtt{Chl}})$-$\mathbf{IND\text{-}CCA}_{\mathsf{KEM}}(\mathcal{A})$ depicted in Figure 13 and define the advantage of adversary $\mathcal{A}$ as

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

$$\mathrm{Adv}^{(\mathrm{n},\mathrm{Q}_{\mathtt{Chl}})\text{-}\mathbf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{A}} \coloneqq \mathrm{Adv}^{(\mathrm{n},0,\mathrm{Q}_{\mathtt{Chl}})\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{A}}.$$

B.3 Ring Signatures

Unforgeability. We consider the notion of *one-per-message unforgeability under chosen ring attacks*, where the adversary is only allowed to make at most one signing query per message/ring pair (m_i, ρ_i) from [GJK24b].

Game $(n, Q_{\mathtt{Dec}}, Q_{\mathtt{Chl}})$-**IND-CCA**$_{\mathsf{KEM}}(\mathcal{A})$

01 **for** $i \in [n]$
02 $\quad (sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
03 $b \xleftarrow{\$} \{0,1\}$
04 $b' \leftarrow \mathcal{A}^{\mathtt{Dec},\mathtt{Chall}}(pk_1, \ldots, pk_n)$
05 **return** $[\![b = b']\!]$

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

06 **if** $\exists\, k : (pk_r, c, k) \in \mathcal{D}$
07 $\quad$ **return** k
08 $k \leftarrow \mathsf{Dec}(sk_r, c)$
09 **return** k

Oracle $\mathtt{Chl}(r \in [n])$

10 $(c, k) \xleftarrow{\$} \mathsf{Enc}(pk_r)$
11 **if** $b = 0$
12 $\quad$ **continue**
13 **if** $b = 1$
14 $\quad k \xleftarrow{\$} \mathcal{K}$
15 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_r, c, k)\}$
16 **return** (c, k)

Figure 13. Game defining **IND-CCA** for a key encapsulation mechanism KEM with adversary $\mathcal{A}$ making at most $Q_{\mathtt{Dec}}$ queries to $\mathtt{Dec}$ and at most $Q_{\mathtt{Chl}}$ queries to $\mathtt{Chl}$.

Game $(n, \kappa, Q_{\mathtt{Sgn}})$-**UF-CRA1**$_{\mathsf{RSig}}(\mathcal{A})$

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

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

07 **if** $pk_i \notin \rho \vee (\rho, m) \in \mathcal{Q}$
08 $\quad$ **return** $\perp$
09 $\sigma \xleftarrow{\$} \mathsf{Sgn}(sk_i, \rho, m)$
10 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\rho, m)\}$
11 **return** σ

Figure 14. Game **UF-CRA1** for a ring signature scheme RSig and adversary $\mathcal{A}$.

The notion is formalised through the game $(n, \kappa, Q_{\mathtt{Sgn}})$-**UF-CRA1**$_{\mathsf{RSig}}(\mathcal{A})$ depicted in Figure 14, where n is the number of users, κ the maximal ring size, and $Q_{\mathtt{Sgn}}$ is an upper bound on the signing queries. We define the advantage functions of adversary $\mathcal{A}$ as

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

Anonymity. We consider *multi-challenge anonymity under full key exposures* of a ring signature RSig from [BFG+22, GJK24b]. It is defined via the game $(n, \kappa, Q_{\mathtt{Chl}})$-**MC-Ano**$_{\mathsf{RSig}}(\mathcal{A})$ for an adversary $\mathcal{A}$, depicted in Figure 15. We define the advantage as

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

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

01 $par \xleftarrow{\$} \mathsf{Stp}(\kappa)$
02 **for** $i \in [n]$
03 $\quad (sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
04 $b \xleftarrow{\$} \{0,1\}$
05 $b' \xleftarrow{\$} \mathcal{A}^{\mathtt{Chl}}(par, (sk_1, pk_1), \ldots, (sk_n, pk_n))$
06 **return** $[\![b = b']\!]$

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

07 **if** $(\rho \subseteq \{pk_1, \ldots, pk_n\}) \wedge (pk_{i_0} \in \rho) \wedge (pk_{i_1} \in \rho)$
08 $\quad \sigma \xleftarrow{\$} \mathsf{Sgn}(sk_{i_b}, \rho, m)$
09 $\quad$ **return** σ
10 **else**
11 $\quad$ **return** $\perp$

Figure 15. Game defining **MC-Ano** for a ring signature scheme RSig with adversary $\mathcal{A}$ making at most $Q_{\mathtt{Chl}}$ queries to $\mathtt{Chl}$.

B.4 Symmetric Encryption

We formalise the notion of ciphertext indistinguishability (**IND-CPA**) for a symmetric encryption scheme Sym via the game $\textbf{IND-CPA}_{\mathsf{Sym}}(\mathcal{A})$ depicted in Figure 16 and define the advantage of adversary $\mathcal{A}$ as

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

Game $\textbf{IND-CPA}_{\mathsf{Sym}}(\mathcal{A})$	**Oracle** $\mathtt{Chl}(m_0, m_1)$ // one query
01 $k \xleftarrow{\$} \mathcal{K}_{\mathsf{Sym}}$	05 $c := \mathsf{Enc}(k, m_b)$
02 $b \xleftarrow{\$} \{0,1\}$	06 **return** c
03 $b' \leftarrow \mathcal{A}^{\mathtt{Chall}}$	
04 **return** $[\![b = b']\!]$	

Figure 16. Game defining **IND-CPA** for a symmetric encryption scheme Sym with adversary $\mathcal{A}$ making at most one query $\mathtt{Chl}$.

C Proofs for Section 3 (Generic Construction)

Theorem 9 (Confidentiality). *For any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* $\Pi := \mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$, *depicted in Figure 8, there exists an* **Ins-CCA** *adversary* $\mathcal{B}_1$ *against* AKEM_1, *an* **Ins-CCA** *adversary* $\mathcal{B}_2$ *against* AKEM_2, *and a* **PRF** *adversary* $\mathcal{C}$ *against* H *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2} \approx t_{\mathcal{C}}$ *such that*

$$\mathrm{Adv}^{\text{Q-}\textbf{Ins-CCA}}_{\Pi,\mathcal{A}} \leq 2 \cdot \min\left\{\mathrm{Adv}^{Q\text{-}\textbf{Ins-CCA}}_{\mathsf{AKEM}_1,\mathcal{B}_1},\ \mathrm{Adv}^{Q\text{-}\textbf{Ins-CCA}}_{\mathsf{AKEM}_2,\mathcal{B}_2}\right\} + 2 \cdot \mathrm{Adv}^{Q_{\mathsf{H}}\text{-}\textbf{PRF}}_{\mathsf{H},\mathcal{C}} + Q_{\mathcal{C}hl} \cdot \delta_{\Pi}$$

for $Q = (n, Q_{Enc}, Q_{Dec}, Q_{\mathcal{C}hl}), Q_{\mathsf{H}} = (Q_{\mathcal{C}hl}, Q_{Dec} + Q_{\mathcal{C}hl})$.

Proof. Consider the sequence of games depicted in Figure 17.

Game G_0 This is the $\textbf{Ins-CCA}_{\mathsf{AKEM}}(\mathcal{A})$ game for $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$ so by definition

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,\mathrm{Q_{Enc}},\mathrm{Q_{Dec}},\mathrm{Q_{Chl}})\text{-}\textbf{Ins-CCA})\text{-}\textbf{Ins-CCA}}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathcal{A}}.$$

Game G_1 Game G_1 is the same as G_0 except that in the challenge oracle an element is added to $\mathcal{D}$ independent of challenge bit b. The changes can only be distinguished if the decapsulation is incorrect. For $Q_{\mathtt{Chl}}$ queries to the challenge oracle, we obtain

$$\left|\Pr\left[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq Q_{\mathtt{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}]}.$$

Game G_2 Game G_2 is the same as G_1 except that in the challenge oracle, the shared key of AKEM_1 is replaced by a uniformly random element of the key space $\mathcal{K}_1$ and stored together with ciphertext c_1 in set $\mathcal{E}_1$. Additionally, the decapsulation oracle is changed to check for a corresponding element in $\mathcal{E}_1$ and the actual KEM key k_1 is replaced by the one stored in $\mathcal{E}_1$.

Claim. There exists an adversary $\mathcal{B}_1$ against the **Ins-CCA** security of AKEM_1, such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(n,\mathrm{Q_{Enc}},\mathrm{Q_{Dec}},\mathrm{Q_{Chl}})\text{-}\textbf{Ins-CCA}}_{\mathsf{AKEM}_1,\mathcal{B}_1}.$$

Proof. Adversary $\mathcal{B}_1$ is formally constructed in Figure 18. If $\mathcal{B}_1$ is in the real case, i.e. challenge bit $b = 0$, they perfectly simulate G_1 for adversary $\mathcal{A}$. In case $b = 1$ they simulate Game G_2 for adversary $\mathcal{A}$. Hence, the advantage of distinguishing between G_1 and G_2 is at most the advantage of $\mathcal{B}_1$. ■

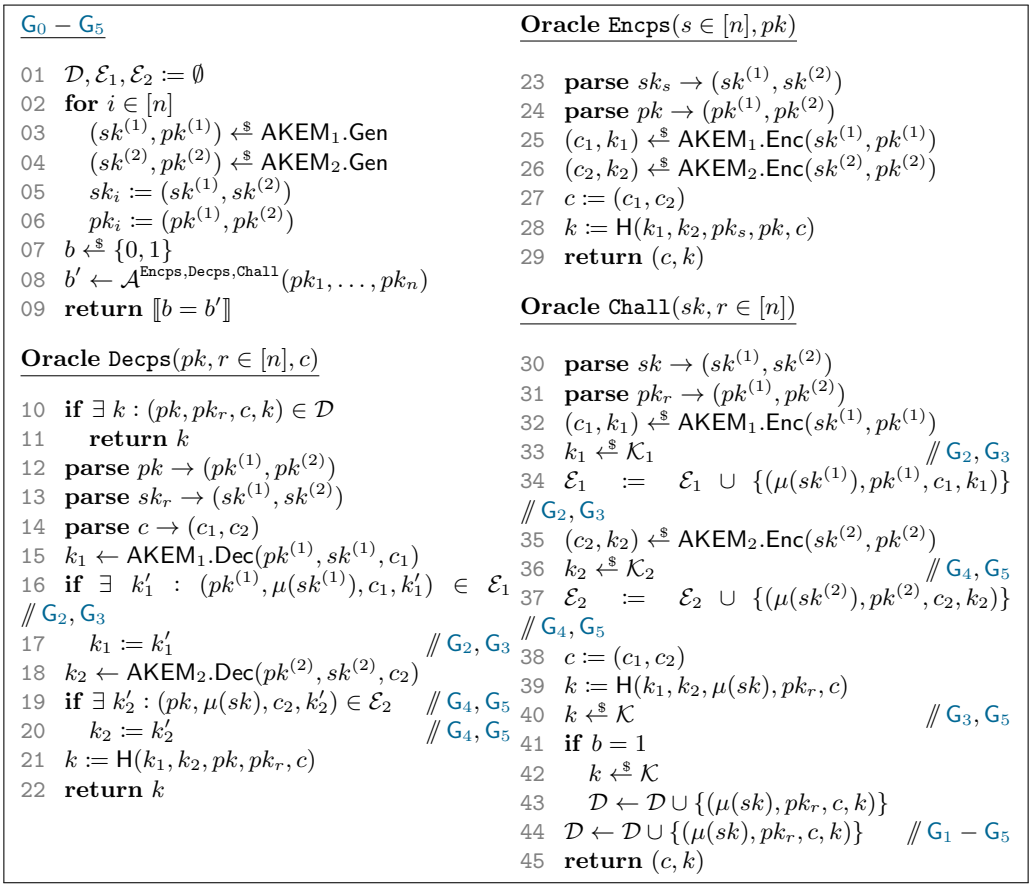

Figure 17. Games $\mathsf{G}_0 - \mathsf{G}_5$ for the proof of Theorem 9.

$\mathcal{B}_1^{\mathtt{Encps}_{\mathcal{B}},\mathtt{Decps}_{\mathcal{B}},\mathtt{Chall}_{\mathcal{B}}}(\hat{pk}_1,\ldots,\hat{pk}_n)$

01 $\mathcal{D},\mathcal{E}_1 := \emptyset$
02 **for** $i \in [n]$
03 $\quad (sk^{(2)}, pk^{(2)}) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$
04 $\quad sk_i := (\bot, sk^{(2)})$
05 $\quad pk_i := (\hat{pk}_i, pk^{(2)})$
06 $b \xleftarrow{\$} \{0,1\}$
07 $b' \leftarrow \mathcal{A}^{\mathtt{Encps},\mathtt{Decps},\mathtt{Chall}}(pk_1,\ldots,pk_n)$
08 **return** $[\![b = b']\!]$

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

09 **if** $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
10 $\quad$ **return** k
11 **parse** $pk \to (pk^{(1)}, pk^{(2)})$
12 **parse** $sk_r \to (\bot, sk^{(2)})$
13 **parse** $c \to (c_1, c_2)$
14 $k_1 \leftarrow \mathtt{Decps}_{\mathcal{B}}(pk^{(1)}, r, c_1)$ // decaps query
15 $k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk^{(2)}, sk^{(2)}, c_2)$
16 $k := \mathsf{H}(k_1, k_2, pk, pk_r, c)$
17 **return** k

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

18 **parse** $sk_s \to (\bot, sk^{(2)})$
19 **parse** $pk \to (pk^{(1)}, pk^{(2)})$
20 $(c_1, k_1) \xleftarrow{\$} \mathtt{Encps}_{\mathcal{B}}(s, pk^{(1)})$ // encaps query
21 $(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
22 $c := (c_1, c_2)$
23 $k := \mathsf{H}(k_1, k_2, pk_s, pk, c)$
24 **return** (c, k)

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

25 **parse** $sk \to (sk^{(1)}, sk^{(2)})$
26 **parse** $pk_r \to (pk^{(1)}, pk^{(2)})$
27 $(c_1, k_1) \xleftarrow{\$} \mathtt{Chall}_{\mathcal{B}}(sk^{(1)}, r)$ // challenge query
28 $(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
29 $c := (c_1, c_2)$
30 $k := \mathsf{H}(k_1, k_2, \mu(sk), pk_r, c)$
31 **if** $b = 1$
32 $\quad k \xleftarrow{\$} \mathcal{K}$
33 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), pk_r, c, k)\}$
34 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), pk_r, c, k)\}$
35 **return** (c, k)

Figure 18. Adversary $\mathcal{B}_1$ against **Ins-CCA** security of AKEM_1, having access to oracles $\mathtt{Encps}_{\mathcal{B}}, \mathtt{Decps}_{\mathcal{B}}, \mathtt{Chall}_{\mathcal{B}}$, and $\mathtt{CorSK}_{\mathcal{B}}$, simulating Game $\mathsf{G}_1/\mathsf{G}_2$ for adversary $\mathcal{A}$ from the proof of Theorem 9.

Game G_3 Game G_3 is the same as G_2 except that the output of the keyed function in the challenge oracle is replaced by a uniformly random output of the output space $\mathcal{K}$.

Claim. There exists an adversary $\mathcal{C}_1$ against the **PRF** security of H_1, i.e. keyed on the first input, such that

$$\left|\Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(\mathtt{Q}_{\mathtt{Chl}}, \mathtt{Q}_{\mathtt{Dec}}+\mathtt{Q}_{\mathtt{Chl}})\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{C}_1}.$$

Proof. We formally construct adversary $\mathcal{C}_1$ in Figure 19. If $\mathcal{C}_1$ is in their own $b = 0$ case of the PRF game, they simulate G_2. In the case $b = 1$, they nearly simulate G_3. Nearly refers to the following distinction: the output of the evaluation oracle of the PRF game is the output of a random function in case $b = 1$ whereas in G_3 the output is randomly sampled from the output space. These two cases are the same if the random function is never queried on the same input as in the challenge oracle again. This is the case in Game G_3 because in the challenge oracle a new PRF key is used and if there was a query to the same random function in the decapsulation oracle, i.e. the same PRF key index ℓ, with the same input $k_2||pk||pk_r||c$, the decapsulation would have returned in Line 12 already and never queried the PRF evaluation oracle. Further, we can see that adversary $\mathcal{C}_1$ needs at most $Q_{\mathtt{Chl}}$ instances and at most $Q_{\mathtt{Dec}} + Q_{\mathtt{Chl}}$ evaluation queries. ■

Game G_4 Game G_4 is the same as G_1 (note that we are not building on top of the last game) except that in the challenge oracle, the shared key of AKEM_2 is replaced by a uniformly random element of the key space $\mathcal{K}_2$ and stored together with ciphertext c_2 in set $\mathcal{E}_2$. Additionally, the decapsulation oracle is changed to check for a corresponding element in $\mathcal{E}_2$ and the actual KEM key k_2 is replaced by the one stored in $\mathcal{E}_2$.

Claim. There exists an adversary $\mathcal{B}_2$ against the **Ins-CCA** security of AKEM_2, such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(n, \mathtt{Q}_{\mathtt{Enc}}\mathtt{Q}_{\mathtt{Dec}}, \mathtt{Q}_{\mathtt{Chl}})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}_2,\mathcal{B}_2}.$$

Proof. The proof is analogue to the one between G_1 and G_2 ■

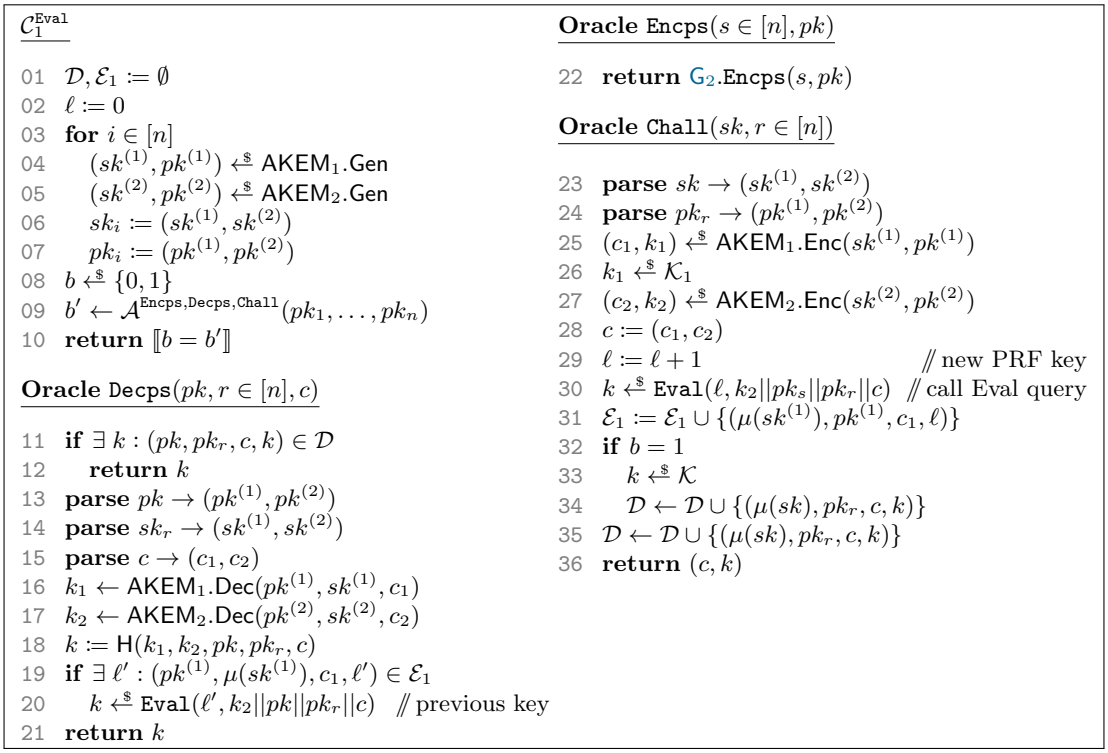

Figure 19. Adversary $\mathcal{C}_1$ against **PRF** security of H_1, having access to oracle `Eval`, simulating Game $\mathsf{G}_2/\mathsf{G}_3$ for adversary $\mathcal{A}$ from the proof of Theorem 9.

Game G_5 Game G_5 is the same as G_4 except that the output of the keyed function in the challenge oracle is replaced by a uniformly random output of the output space $\mathcal{K}$.

Claim. There exists an adversary $\mathcal{C}_2$ against the **PRF** security of H_2, i.e. keyed on the first input, such that

$$\left|\Pr\left[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(\mathtt{Q_{Chl}},\mathtt{Q_{Dec}}+\mathtt{Q_{Chl}})\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{C}_2}.$$

Proof. The proof is analogue to the one between G_2 and G_3. ■

Game G_3 as well as Game G_5 are independent of the challenge bit b. Hence, we obtain

$$\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

■

Theorem 10 (Authenticity). *For any* **Out-Aut** *adversary* $\mathcal{A}$ *against* $\Pi \coloneqq \mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, as depicted in Figure 8, there exists an* **Out-Aut** *adversary* $\mathcal{B}_1$ *against* AKEM_1*, an* **Out-Aut** *adversary* $\mathcal{B}_2$ *against* AKEM_2*, an* **Out-CCA** *adversary* $\mathcal{C}_1$ *against* AKEM_1*, an* **Out-CCA** *adversary* $\mathcal{C}_2$ *against* AKEM_2*, and a* **PRF** *adversary* $\mathcal{D}$ *against* H *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2} \approx t_{\mathcal{C}_1} \approx t_{\mathcal{C}_2} \approx t_{\mathcal{D}}$ *such that*

$$\begin{aligned}\mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}Aut}}_{\Pi,\mathcal{A}} \leq & 2 \cdot \min\left\{\mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}_1,\mathcal{B}_1} + \mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}CCA}}_{\mathsf{AKEM}_1,\mathcal{C}_1},\ \mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}_2,\mathcal{B}_2} + \mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}CCA}}_{\mathsf{AKEM}_2,\mathcal{C}_2}\right\} \\ & + 2 \cdot \mathrm{Adv}^{Q_{\mathsf{H}}\text{-}\mathbf{PRF}}_{\mathsf{H},\mathcal{D}} + Q_{\mathtt{Chl}} \cdot \delta_{\Pi}\end{aligned}$$

for $Q = (n, Q_{\mathtt{Enc}}, Q_{\mathtt{Chl}}), Q_{\mathsf{H}} = (Q_{\mathtt{Enc}} + Q_{\mathtt{Chl}}, Q_{\mathtt{Enc}} + Q_{\mathtt{Chl}})$.

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

Game G_0 This is the **Out-Aut** game for $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$ so by definition

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,\mathtt{Q_{Enc}},\mathtt{Q_{Chl}})\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathcal{A}}.$$

Game G_1 Game G_1 is the same as G_0 except that in the challenge oracle set $\mathcal{D}$ is filled in case $b = 0$ as well. If the scheme is perfectly correct, the change cannot be distinguished since the difference is that $\mathcal{D}$ stores either tuples from encapsulations or from correct decapsulations. Hence, the difference is at most the correctness error per query to the challenge oracle:

$$\left|\Pr\left[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq Q_{\mathtt{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}]}.$$

Game G_2 Game G_2 is the same as G_1 except that the output of the AKEM_1 decapsulation, k_1, is replaced by a uniformly random sample from the key space $\mathcal{K}_1$ if the first receiver public key, $pk^{(1)}$, is honest and the shared key is not $\perp$ (Line 33) and the result is stored together with the sender's and receiver's public key for AKEM_1 as well as the first ciphertext c_1 in set $\mathcal{E}_1$ (Line 34). For consistent outputs, an element of this form is also added to $\mathcal{E}_1$ in an encapsulation query (Line 15) and if there already exists a matching element in $\mathcal{E}_1$ the decapsulation output is replaced by this element instead of randomly choosing a new one (Line 31).

Claim. There exists an adversary $\mathcal{B}_1$ against the **Out-Aut** security of AKEM_1, such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(n,\mathtt{Q_{Enc}},\mathtt{Q_{Chl}})\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}_1,\mathcal{B}_1}.$$

Proof. Adversary $\mathcal{B}_1$ is formally constructed in Figure 21. If they are in case $b = 0$, they simulate Game G_1. In case $b = 1$, they simulate G_2. Further, the number of queries to $\mathtt{Encps}_{\mathcal{B}}$ and $\mathtt{Chall}_{\mathcal{B}}$ is the same as for adversary $\mathcal{A}$.

■

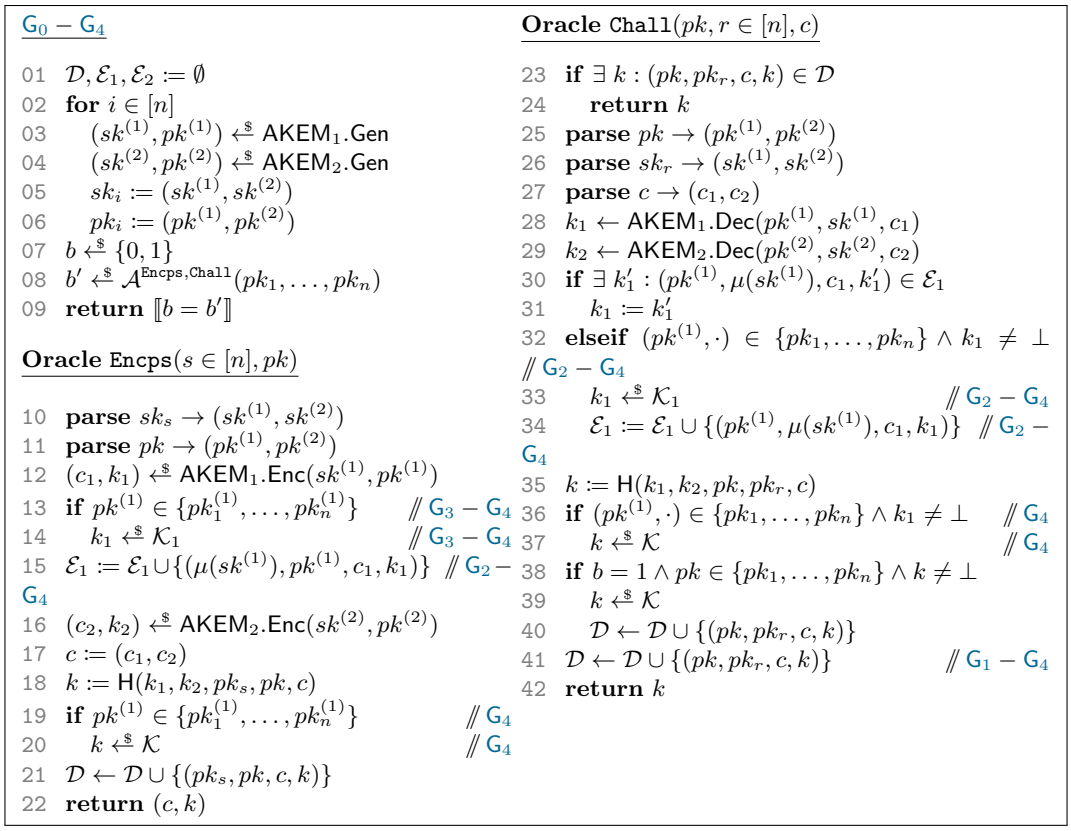

Figure 20. Games for the proof of Theorem 10.

Game G_3 Game G_3 is the same as G_2 except that the KEM key of AKEM_1 in $\mathtt{Encps}$ is replaced by a uniformly random value of the key space $\mathcal{K}_1$.

Claim. There exists an adversary $\mathcal{C}_1$ against the **Out-CCA** security of AKEM_1, such that

$$\left|\Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(n,\mathrm{Q}_{\mathtt{Enc}},\mathrm{Q}_{\mathtt{Chl}})\text{-}\mathbf{Out\text{-}CCA}}_{\mathsf{AKEM}_1,\mathcal{C}_1}.$$

Proof. Adversary $\mathcal{C}_1$ is constructed in Figure 22. In G_2, the encapsulation is the real encapsulation of AKEM_1, thus querying the oracle $\mathtt{Encps}_{\mathcal{C}}$ simulates Game G_2 for adversary $\mathcal{A}$. In the **Out-CCA** case $b = 1$, the encapsulation oracle $\mathtt{Encps}_{\mathcal{C}}$ returns a uniformly random key of the key space $\mathcal{K}_1$ which perfectly simulates G_3. Note that in Game G_3, the key is randomly chosen for honest receivers only. The number of encapsulation and decapsulation queries of $\mathcal{C}$ equals exactly the ones of $\mathcal{A}$.

■

Game G_4 Game G_4 is the same as G_3 except that the output of keyed function H in the challenge oracle is replaced by a uniformly random output in case the first public key, $pk^{(1)}$, is honest and the KEM key k_1 is not $\perp$. Further, the output of keyed function H is also replaced by a random value in the encapsulation oracle if the receiver is honest.

Claim. There exists an adversary $\mathcal{D}_1$ against the **PRF** security of H_1, such that

$$\left|\Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(\mathrm{Q}_{\mathtt{Enc}}+\mathrm{Q}_{\mathtt{Chl}},\mathrm{Q}_{\mathtt{Enc}}+\mathrm{Q}_{\mathtt{Chl}})\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{D}_1}.$$

$\mathcal{B}_1^{\texttt{Encps}_{\mathcal{B}},\texttt{Chall}_{\mathcal{B}}}(pk_1^{(1)},\ldots,pk_n^{(1)})$

$\mathcal{D} \coloneqq \emptyset$
for $i \in [n]$
  $(sk^{(2)}, pk^{(2)}) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$
  $sk_i \coloneqq (\bot, sk^{(2)})$
  $pk_i \coloneqq (pk_i^{(1)}, pk^{(2)})$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\texttt{Encps},\texttt{Chall}}(pk_1,\ldots,pk_n)$
return $[\![b = b']\!]$

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

parse $sk_s \to (\bot, sk^{(2)})$
parse $pk \to (pk^{(1)}, pk^{(2)})$
$(c_1, k_1) \xleftarrow{\$} \texttt{Encps}_{\mathcal{B}}(s, pk^{(1)})$ // encaps query
$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
$c \coloneqq (c_1, c_2)$
$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk, c)$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

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

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
parse $pk \to (pk^{(1)}, pk^{(2)})$
parse $sk_r \to (\bot, sk^{(2)})$
parse $c \to (c_1, c_2)$
$k_1 \leftarrow \texttt{Chall}_{\mathcal{B}}(pk^{(1)}, r, c_1)$ // challenge query
$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk^{(2)}, sk^{(2)}, c_2)$
$k \coloneqq \mathsf{H}(k_1, k_2, pk, pk_r, c)$
if $b = 1 \wedge pk \in \{pk_1,\ldots,pk_n\} \wedge k \neq \bot$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
return k

Figure 21. Adversary $\mathcal{B}_1$ against **Out-Aut** security of AKEM_2, having access to oracles $\texttt{Encps}_{\mathcal{B}}$ and $\texttt{Chall}_{\mathcal{B}}$, simulating Game $\mathsf{G}_1/\mathsf{G}_2$ for adversary $\mathcal{A}$ from the proof of Theorem 10.

$\mathcal{C}^{\texttt{Encps}_{\mathcal{C}},\texttt{Decps}_{\mathcal{C}}}(pk_1^{(1)},\ldots,pk_n^{(1)})$

$\mathcal{D}, \mathcal{E}_1, \mathcal{E}_2 \coloneqq \emptyset$
for $i \in [n]$
  $(sk^{(2)}, pk^{(2)}) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$
  $sk_i \coloneqq (\bot, sk^{(2)})$
  $pk_i \coloneqq (pk_i^{(1)}, pk^{(2)})$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\texttt{Encps},\texttt{Chall}}(pk_1,\ldots,pk_n)$
return $[\![b = b']\!]$

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

parse $sk_s \to (\bot, sk^{(2)})$
parse $pk \to (pk^{(1)}, pk^{(2)})$
$(c_1, k_1) \xleftarrow{\$} \texttt{Encps}_{\mathcal{C}}(s, pk^{(1)})$ // encaps query
$\mathcal{E}_1 \coloneqq \mathcal{E}_1 \cup \{pk_s^{(1)}, pk^{(1)}, c_1, k_1)\}$
$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
$c \coloneqq (c_1, c_2)$
$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk, c)$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

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

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
parse $pk \to (pk^{(1)}, pk^{(2)})$
parse $sk_r \to (\bot, sk^{(2)})$
parse $c \to (c_1, c_2)$
$k_1 \leftarrow \texttt{Decps}_{\mathcal{C}}(pk^{(1)}, r, c_1)$ // decaps query
$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk^{(2)}, sk^{(2)}, c_2)$
if $\exists\, k_1' : (pk^{(1)}, pk_r^{(1)}, c_1, k_1') \in \mathcal{E}_1$
  $k_1 \coloneqq k_1'$
elseif $(pk^{(1)}, \cdot) \in \{pk_1,\ldots,pk_n\} \wedge k_1 \neq \bot$
  $k_1 \xleftarrow{\$} \mathcal{K}_1$
  $\mathcal{E}_1 \coloneqq \mathcal{E}_1 \cup \{(pk^{(1)}, pk_r^{(1)}, c_1, k_1)\}$
$k \coloneqq \mathsf{H}(k_1, k_2, pk, pk_r, c)$
if $b = 1 \wedge pk \in \{pk_1,\ldots,pk_n\} \wedge k \neq \bot$
  $k \xleftarrow{\$} \mathcal{K}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
return k

Figure 22. Adversary $\mathcal{C}_1$ against **Out-CCA** security of AKEM_2, having access to oracles $\texttt{Encps}_{\mathcal{B}}$ and $\texttt{Decps}_{\mathcal{B}}$, simulating Game $\mathsf{G}_2/\mathsf{G}_3$ for adversary $\mathcal{A}$ from the proof of Theorem 10.

Proof. Adversary $\mathcal{D}_1$ is formally constructed in Figure 23.

If $\mathcal{D}_1$ is in their own $b = 0$ case of the PRF game, they simulate G_3. In the case $b = 1$, they nearly simulate G_4. Nearly refers to the following distinction: the output of the evaluation oracle of the PRF game is the output of a random function in case $b = 1$ whereas in G_4 the output is randomly sampled from the output

space. With the same argument as in Game 3 of the proof of Theorem 9, we obtain a perfect simulation. The maximal number of different PRF keys is the same as maximal evaluation queries and amounts to $Q_{\texttt{Enc}} + Q_{\texttt{Chl}}$.

$\underline{\mathcal{D}_1^{\texttt{Eval}}}$

$\ell := 0$
$\mathcal{D}, \mathcal{E}_1, \mathcal{E}_2 := \emptyset$
for $i \in [n]$
  $(sk^{(1)}, pk^{(1)}) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$
  $(sk^{(2)}, pk^{(2)}) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$
  $sk_i := (sk^{(1)}, sk^{(2)})$
  $pk_i := (pk^{(1)}, pk^{(2)})$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\texttt{Encps},\texttt{Chall}}(pk_1, \ldots, pk_n)$
return $[\![b = b']\!]$

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

parse $sk_s \to (sk^{(1)}, sk^{(2)})$
parse $pk \to (pk^{(1)}, pk^{(2)})$
$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk^{(1)}, pk^{(1)})$
if $pk^{(1)} \in \{pk_1^{(1)}, \ldots, pk_n^{(1)}\}$
  $\ell := \ell + 1$ // new key
  $\mathcal{E}_1 := \mathcal{E}_1 \cup \{(\mu(sk^{(1)}), pk^{(1)}, c_1, \ell)\}$ // store key
$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
$c := (c_1, c_2)$
$k := \mathsf{H}(k_1, k_2, pk_s, pk, c)$
if $pk^{(1)} \in \{pk_1^{(1)}, \ldots, pk_n^{(1)}\}$
  $k_1 \xleftarrow{\$} \texttt{Eval}(\ell, k_2 \| pk_s \| pk \| c)$ // eval query
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

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

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
parse $pk \to (pk^{(1)}, pk^{(2)})$
parse $sk_r \to (sk^{(1)}, sk^{(2)})$
parse $c \to (c_1, c_2)$
$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk^{(1)}, sk^{(1)}, c_1)$
$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk^{(2)}, sk^{(2)}, c_2)$
if $\exists\, \ell' : (pk^{(1)}, \mu(sk^{(1)}), c_1, \ell') \in \mathcal{E}_1$
  $k \xleftarrow{\$} \texttt{Eval}(\ell', k_2 \| pk \| pk_r \| c)$ // eval query
elseif $(pk^{(1)}, \cdot) \in \{pk_1, \ldots, pk_n\} \wedge k_1 \neq \bot$
  $\ell := \ell + 1$ // new key
  $\mathcal{E}_1 := \mathcal{E}_1 \cup \{(pk^{(1)}, \mu(sk^{(1)}), c_1, \ell)\}$ // store key
  $k \xleftarrow{\$} \texttt{Eval}(\ell, k_2 \| pk \| pk_r \| c)$ // eval query
else
  $k := \mathsf{H}(k_1, k_2, pk, pk_r, c)$
if $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \bot$
  $k \xleftarrow{\$} \mathcal{K}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
return k

Figure 23. Adversary $\mathcal{D}_1$ against **PRF** security of H_1, having access to oracle $\texttt{Eval}$, simulating Game $\mathsf{G}_3/\mathsf{G}_4$ for adversary $\mathcal{A}$ from the proof of Theorem 10.

■

We can see that G_4 is independent of the challenge bit b since the shared key in case $b = 0$ is uniformly random under condition $pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \bot$ which is the same output as in case $b = 1$. Thus, we obtain

$$\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Game G_5 Game G_5 is the same as G_1 (not the previous game) except that the same changes from $\mathsf{G}_2 - \mathsf{G}_4$ are applied to AKEM_2 instead of AKEM_1. Note that we did not show these games in Figure 20 to sustain readability.

Claim. There exist an adversaries $\mathcal{B}_2$ against the **Out-Aut** security of AKEM_2, $\mathcal{C}_2$ against the **Out-CCA** security of AKEM_2, and $\mathcal{D}_2$ against the **PRF** security of H_2, such that

$$\begin{aligned}\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1\right]\right| &\leq 2 \cdot \mathrm{Adv}^{(n,\mathrm{Q}_{\texttt{Enc}},\mathrm{Q}_{\texttt{Chl}})\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}_2,\mathcal{B}_2} \\ &+ 2 \cdot \mathrm{Adv}^{(n,\mathrm{Q}_{\texttt{Enc}},\mathrm{Q}_{\texttt{Chl}})\text{-}\mathbf{Out\text{-}CCA}}_{\mathsf{AKEM}_2,\mathcal{C}_2} + 2 \cdot \mathrm{Adv}^{(\mathrm{Q}_{\texttt{Enc}}+\mathrm{Q}_{\texttt{Chl}},\mathrm{Q}_{\texttt{Enc}}+\mathrm{Q}_{\texttt{Chl}})\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{D}_2}.\end{aligned}$$

The claim can be proved analogously to hybrids $\mathsf{G}_2 - \mathsf{G}_4$. Further, it also holds

$$\Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Combining the differences, we obtain the theorem statement.

■

Theorem 11 (Dishonest Deniability). *For any* **DR-Den** *adversary* $\mathcal{A}$ *against* $\Pi := \mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, as depicted in Figure 8, any simulators* $\mathsf{Sim}_1, \mathsf{Sim}_2$*, and simulator* $\mathsf{Sim}[\mathsf{Sim}_1, \mathsf{Sim}_2]$ *as defined in Figure 24 there exists a* **DR-Den** *adversary* $\mathcal{B}_1$ *against* AKEM_1 *and a* **DR-Den** *adversary* $\mathcal{B}_2$ *against* AKEM_2 *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}_1} \approx t_{\mathcal{B}_2}$ *such that*

$$\mathrm{Adv}^{Q\text{-}\mathbf{DR\text{-}Den}}_{\Pi,\mathsf{Sim},\mathcal{A}} \leq 2 \cdot \mathrm{Adv}^{Q\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}_1,\mathsf{Sim}_1,\mathcal{B}_1} + 2 \cdot \mathrm{Adv}^{Q\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}_2,\mathsf{Sim}_2,\mathcal{B}_2}$$

for $Q = (n, Q_{\texttt{Chl}})$.

Proof. Consider the sequence of games depicted in Figure 24.

Game G_0 This is the **DR-Den** game for $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$ and simulator $\mathsf{Sim} = \mathsf{Sim}[\mathsf{Sim}_1, \mathsf{Sim}_2]$ as defined in Figure 24. By definition, it holds

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,\mathrm{Q}_{\texttt{Chl}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathsf{Sim},\mathcal{A}}.$$

Unlike the definition, the adversary is given all the secret keys in the beginning. However, since there is no restriction on the reveal oracle calls in the dishonest deniability setting, G_0 is equivalent to the original definition. Let Sim_1 and Sim_2 be the simulators for AKEM_1 and AKEM_2, respectively. The simulator Sim is then defined in terms of Sim_1 and Sim_2.

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

01 **for** $i \in [n]$
02 $\quad (sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
03 $b \xleftarrow{\$} \{0,1\}$
04 $b' \leftarrow \mathcal{A}^{\texttt{Chall}}((sk_1, pk_1), \ldots, (sk_n, pk_n))$
05 **return** $[\![b = b']\!]$

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

06 **parse** $pk_s \to (pk_s^{(1)}, pk_s^{(2)})$
07 **parse** $pk_r \to (pk_r^{(1)}, pk_r^{(2)})$
08 **parse** $sk_r \to (sk_r^{(1)}, sk_r^{(2)})$
09 $(c_1, k_1) \xleftarrow{\$} \mathsf{Sim}_1(pk_s^{(1)}, pk_r^{(1)}, sk_r^{(1)})$
10 $(c_2, k_2) \xleftarrow{\$} \mathsf{Sim}_2(pk_s^{(2)}, pk_r^{(2)}, sk_r^{(2)})$
11 $c := (c_1, c_2)$
12 $k := \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
13 **return** (c, k)

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

14 **if** $s = r$ **return** $\perp$
15 **parse** $sk_s \to (sk^{(1)}, sk^{(2)})$
16 **parse** $pk_r \to (pk^{(1)}, pk^{(2)})$
17 **parse** $pk_s \to (pk_s^{(1)}, pk_s^{(2)})$
18 **parse** $sk_r \to (sk_r^{(1)}, sk_r^{(2)})$
19 $(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk^{(1)}, pk^{(1)})$
20 $(c_1, k_1) \xleftarrow{\$} \mathsf{Sim}_1(pk_s^{(1)}, pk^{(1)}, sk_r^{(1)})$ // $\mathsf{G}_0 - \mathsf{G}_1$
21 $(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
22 $(c_2, k_2) \xleftarrow{\$} \mathsf{Sim}_2(pk_s^{(2)}, pk^{(2)}, sk_r^{(2)})$ // $\mathsf{G}_0 - \mathsf{G}_2$
23 $c := (c_1, c_2)$
24 $k := \mathsf{H}(k_1, k_2, (\mu(sk^{(1)}), \mu(sk^{(2)})), pk_r, c)$
25 **if** $b = 1$
26 $\quad (c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r, sk_r)$
27 **return** (c, k)

Figure 24. Games for the proof of Theorem 11 and definition of simulator $\mathsf{Sim} = \mathsf{Sim}[\mathsf{Sim}_1, \mathsf{Sim}_2]$.

Game G_1 This is the same as G_0 except that the output of the encapsulation of AKEM_1 is replaced by the output of simulator Sim_1.

Claim. There exists an adversary $\mathcal{B}_1$ and a simulator Sim_1 such that

$$\left|\Pr\left[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(n,\mathrm{Q_{Ch1}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}_1,\mathsf{Sim}_1,\mathcal{B}_1}.$$

Proof. Adversary $\mathcal{B}_1$ can be constructed by simulating the game for $\mathcal{A}$ and querying their own challenge oracle to get (c_1, k_1). If they are in the real game $b = 0$, they are simulating G_0, otherwise they are simulating G_1. ■

Game G_2 This is the same as G_1 except that the output of the encapsulation of AKEM_1 is replaced by the output of simulator Sim_2.

Claim. There exists an adversary $\mathcal{B}_2$ and a simulator Sim_2 such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(n,\mathrm{Q_{Ch1}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}_2,\mathsf{Sim}_2,\mathcal{B}_2}.$$

Proof. The proof can be done analogously to the one of the previous game. ■

The resulting game behaves exactly the same in case $b = 0$ and $b = 1$, thus we have

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

■

D Proofs for Section 4 (Concrete Construction)

Theorem 14 (Confidentiality). *For any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* $\Pi := \mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 9, there exists a* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, a* **PRF** *adversary* $\mathcal{C}$ *against* H_1*, an* **PRF** *adversary* $\mathcal{D}$ *against* H_2*, and an* **IND-CCA** *adversary* $\mathcal{E}$ *against* KEM *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}} \approx t_{\mathcal{C}} \approx t_{\mathcal{D}} \approx t_{\mathcal{E}}$ *such that*

$$\begin{aligned}\mathrm{Adv}^{Q\text{-}\mathbf{Ins\text{-}CCA}}_{\Pi,\mathcal{A}} \leq 2nQ_{Chl} \cdot \Big(&\min\left\{\mathrm{Adv}^{Q_{\mathsf{NIKE}}\text{-}\mathbf{CKS}}_{\mathsf{NIKE},\mathcal{B}} + \mathrm{Adv}^{(1,1)\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{C}}, \mathrm{Adv}^{(1,Q_{Dec},1)\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{E}}\right\} \\ &+ 2 \cdot \mathrm{Adv}^{(1,Q_{Dec}+1)\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{D}} + (Q_{Enc} + Q_{Dec}) \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}} + Q_{Chl} \cdot \delta_{\Pi}\Big)\end{aligned}$$

for $Q = (n, Q_{Enc}Q_{Dec}, Q_{Chl}), Q_{\mathsf{NIKE}} = (Q_{Enc} + 2, 2Q_{Enc} + 2Q_{Dec}, 2Q_{Enc} + 2Q_{Enc} + 1)$.

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

Game G_0 We start with the $\mathbf{Ins\text{-}CCA}_{\mathsf{AKEM}}(\mathcal{A})$ game for $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$ for one user where the adversary is restricted to one challenge query. Another change which does not influence the winning probability is that we sample all the NIKE keys needed for the game in advance and assign them when needed. More specifically, we need $Q_{\mathtt{Enc}} + 2$ keys: one for the challenge user key, $npk^{\star}$, one for the ephemeral key in the challenge, $npk_e^{\star}$, and $Q_{\mathtt{Enc}}$ many ephemeral keys to answer the encapsulation queries. By definition it holds

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(1,Q_{\mathtt{Enc}}Q_{\mathtt{Dec}},1)\text{-}\mathbf{Ins\text{-}CCA})}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{Sym},\mathsf{H}_1,\mathsf{H}_2],\mathcal{A}}.$$

$\underline{\mathsf{G}_0 - \mathsf{G}_7}$

$\mathcal{D}, \mathcal{H} := \emptyset$
$kct^\star, kk^\star := \bot$
for $\ell \in [Q_{\mathtt{Enc}} + 2]$
  $(nsk_\ell, npk_\ell) \xleftarrow{\$} \mathsf{NIKE.Gen}$
$(nsk^\star, npk^\star) := (nsk_1, npk_1)$
$(nsk_e^\star, npk_e^\star) := (nsk_2, npk_2)$
for $i \in [Q_{\mathtt{Enc}} + 2]$ // $\mathsf{G}_3 - \mathsf{G}_5$
  $k_{ii} := \bot$ // $\mathsf{G}_3 - \mathsf{G}_5$
  for $j \in [i+1, Q_{\mathtt{Enc}} + 2]$ // $\mathsf{G}_3 - \mathsf{G}_5$
    $k_{ij} := k_{ji} \xleftarrow{\$} \mathcal{K}_{\mathsf{NIKE}}$ // $\mathsf{G}_3 - \mathsf{G}_5$
$k^\star := k_{12}$ // $\mathsf{G}_3 - \mathsf{G}_5$
$k_{\mathsf{H}_1} \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_1}$ // $\mathsf{G}_4 - \mathsf{G}_5$
$\ell := 2$
$(ksk^\star, kpk^\star) \xleftarrow{\$} \mathsf{KEM.Gen}$
$(ssk^\star, spk^\star) \xleftarrow{\$} \mathsf{RSig.Gen}$
$(sk^\star, pk^\star) := ((nsk^\star, ksk^\star, ssk^\star), (npk^\star, kpk^\star, spk^\star))$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathtt{Encps},\mathtt{Decps},\mathtt{Chall}}(pk^\star)$
return $[\![b = b']\!]$

Oracle $\mathtt{Decps}(pk, c)$

if $\exists\, k : (pk, c, k) \in \mathcal{D}$
  return k
parse $pk \to (npk, kpk, spk)$
parse $c \to (npk_e, kct, sct)$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk^\star, npk)$
if $npk = npk_e^\star$ // $\mathsf{G}_3 - \mathsf{G}_5$
  $nk' := k^\star$ // $\mathsf{G}_3 - \mathsf{G}_5$
elseif $\exists\, i : npk = npk_i$ // $\mathsf{G}_3 - \mathsf{G}_5$
  $nk' := k_{1i}$ // $\mathsf{G}_3 - \mathsf{G}_5$
$nk := \mathsf{H}_1(k_1', \mathtt{"auth"})$
if $npk = npk_e^\star$ // $\mathsf{G}_4 - \mathsf{G}_5$
  $nk := k_{\mathsf{H}_1}$ // $\mathsf{G}_4 - \mathsf{G}_5$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk^\star, npk_e)$
if $npk_e = npk_e^\star$ // $\mathsf{G}_3 - \mathsf{G}_5$
  $nk_1 \| nk_2 := k^\star$ // $\mathsf{G}_3 - \mathsf{G}_5$
elseif $\exists\, i : npk_e = npk_i$ // $\mathsf{G}_3 - \mathsf{G}_5$
  $nk_1 \| nk_2 := k_{1i}$ // $\mathsf{G}_3 - \mathsf{G}_5$
$kk_1 \| kk_2 \leftarrow \mathsf{KEM.Dec}(ksk^\star, kct)$
if $kct = kct^\star$ // $\mathsf{G}_6 - \mathsf{G}_7$
  $kk_1 \| kk_2 := kk^\star$ // $\mathsf{G}_6 - \mathsf{G}_7$
$k' := \mathsf{H}_1(nk_1, kk_1)$
$\sigma := \mathsf{Sym.Dec}(k', sct)$
$m \leftarrow (kct, kpk^\star)$
if $\mathsf{RSig.Ver}(\sigma, \rho = \{spk, spk^\star\}, m) \neq 1$
  return $\bot$
$k := \mathsf{H}_2(nk, nk_2, kk_2, c, pk, pk^\star)$
if $\exists\, k' : (k', nk, nk_2, kk_2, c, pk, pk^\star) \in \mathcal{H}$ // $\mathsf{G}_5, \mathsf{G}_7$
  $k := k'$ // $\mathsf{G}_5, \mathsf{G}_7$
elseif $npk_e = npk_e^\star$ // G_5
  $k \xleftarrow{\$} \mathcal{K}$ // G_5
elseif $kct = kct^\star$ // G_7
  $k \xleftarrow{\$} \mathcal{K}$ // G_7
$\mathcal{H} := \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk, pk^\star)\}$ // $\mathsf{G}_1 - \mathsf{G}_7$
return k

Oracle $\mathtt{Encps}(pk)$

$\ell := \ell + 1$
parse $pk \to (npk, kpk, spk)$
$(nsk_e, npk_e) := (nsk_\ell, npk_\ell)$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk^\star, npk)$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk)$
if $npk = npk_e^\star$ // $\mathsf{G}_3 - \mathsf{G}_5$
  $nk' := k^\star$ // $\mathsf{G}_3 - \mathsf{G}_5$
  $nk_1 \| nk_2 := k_{\ell 2}$ // $\mathsf{G}_3 - \mathsf{G}_5$
elseif $\exists\, i : npk = npk_i$ // $\mathsf{G}_3 - \mathsf{G}_5$
  $nk' := k_{1i}$ // $\mathsf{G}_3 - \mathsf{G}_5$
  $nk_1 \| nk_2 := k_{\ell i}$ // $\mathsf{G}_3 - \mathsf{G}_5$
$nk := \mathsf{H}_1(k_1', \mathtt{"auth"})$
if $npk = npk_e^\star$ // $\mathsf{G}_4 - \mathsf{G}_5$
  $nk := k_{\mathsf{H}_1}$ // $\mathsf{G}_4 - \mathsf{G}_5$
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
$m \leftarrow (kct, kpk)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk^\star, \{spk^\star, spk\}, m)$
$k' := \mathsf{H}_1(nk_1, kk_1)$
$sct := \mathsf{Sym.Enc}(k', \sigma)$
$c := (npk_e, kct, sct)$
$k := \mathsf{H}_2(nk, nk_2, kk_2, c, pk^\star, pk)$
$\mathcal{H} := \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk^\star, pk^)\}$ // $\mathsf{G}_1 - \mathsf{G}_7$
return (c, k)

Oracle $\mathtt{Chall}(sk)$ // one query

parse $sk \to (nsk, ksk, ssk)$
$(nsk_e, npk_e) := (nsk_e^\star, npk_e^\star)$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk, npk^\star)$
$nk := \mathsf{H}_1(k_1', \mathtt{"auth"})$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk^\star)$
$nk_1 \| nk_2 := k^\star$ // $\mathsf{G}_3 - \mathsf{G}_5$
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk^\star)$
$kk_1 \| kk_2 \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$ // $\mathsf{G}_6 - \mathsf{G}_7$
$(kct^\star, kk^\star) := (kct, kk_1 \| kk_2)$
$m \leftarrow (kct, kpk^\star)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk, \{\mu(ssk), spk^\star\}, m)$
$k' := \mathsf{H}_1(nk_1, kk_2)$
$sct := \mathsf{Sym.Enc}(k', \sigma)$
$c := (npk_e, kct, sct)$
if $\exists\, k' : (k', nk, nk_2, kk_2, c, \mu(sk), pk^\star) \in \mathcal{H}$ // $\mathsf{G}_2 - \mathsf{G}_7$
  abort // $\mathsf{G}_2 - \mathsf{G}_7$
$k := \mathsf{H}_2(nk, nk_2, kk_2, c, \mu(sk), pk^\star)$
$k \xleftarrow{\$} \mathcal{K}$ // $\mathsf{G}_5, \mathsf{G}_7$
$\mathcal{H} := \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, \mu(sk), pk^\star)\}$ // $\mathsf{G}_1 - \mathsf{G}_7$
if $b = 1$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), c, k)\}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), c, k)\}$ // $\mathsf{G}_1 - \mathsf{G}_7$
return (c, k)

Figure 25. Games $\mathsf{G}_0 - \mathsf{G}_7$ for the proof of Theorem 14.

Game G_1 Game G_1 is the same as G_0 except that in the challenge oracle an element is added to $\mathcal{D}$ independent of challenge bit b. Additionally, all inputs to keyed function H_2 are stored together with their output in set

$\mathcal{H}$. If the scheme is correct, these changes are indistinguishable

$$\left|\Pr\left[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq Q_{\mathtt{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{Sym},\mathsf{H}_1,\mathsf{H}_2]}.$$

Game G_2 This game is the same as G_1 except that the game aborts in the challenge oracle if there already exists an element in set $\mathcal{H}$ with the same inputs.

Claim.

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq (Q_{\mathtt{Enc}} + Q_{\mathtt{Dec}}) \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}}.$$

Proof. If there was a previous query to H_2 on the same inputs, this includes ciphertext c. Part of the ciphertext is the ephemeral NIKE key $pk_e^\star$ chosen in the challenge and the KEM ciphertext $kct^\star$. For one element in $\mathcal{H}$, the probability that these two values are the same is at most $\eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{NIKE}}$. Since for each query to `Encps` and `Decps` an element is added to $\mathcal{H}$, we obtain the bound in the claim. ■

Game G_3 Game G_3 is the same as G_2 except that several NIKE shared keys are replaced by a uniformly random value from the NIKE key space $\mathcal{K}_{\mathsf{NIKE}}$. In the challenge oracle, the second NIKE shared key, $nk_1 \| nk_2$, is replaced (Line 82). In the encapsulation oracle, both shared keys, nk' and $nk_1 \| nk_2$, are replaced if the input NIKE key npk is a public key which was originally created in the beginning of the game. In the decapsulation oracle, the first shared key, nk', is replaced if the input public npk is a public key which was originally created in the beginning of the game and the second shared key, $nk_1 \| nk_2$, if this holds for the ephemeral key k_e being part of the input ciphertext. If the same NIKE key is queried again (or in reverse order of the input keys), the previous result is used to keep consistency. To simplify the depiction of consistent assignments, all possible key combinations are sampled in the beginning of the game (Line 07 - Line 11) and the keys are assigned accordingly when the events trigger.

Claim. There exists an adversary $\mathcal{B}$ against the **CKS** security of NIKE such that

$$\left|\Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(\mathrm{Q}_{\mathtt{Enc}}+2, 2\mathrm{Q}_{\mathtt{Enc}}+2\mathrm{Q}_{\mathtt{Dec}}, 2\mathrm{Q}_{\mathtt{Enc}}+2\mathrm{Q}_{\mathtt{Enc}}+1)\text{-}\mathbf{CKS}}_{\mathsf{NIKE},\mathcal{B}}.$$

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 26. They obtain public keys $npk_1, \ldots, npk_{Q_{\mathtt{Enc}}+2}$ of honest users of the **CKS** game. The first key is given to adversary $\mathcal{A}$ as part of the AKEM public key. The second key is assigned to the ephemeral key in the challenge query. Encapsulation and decapsulation queries can be simulated by using the test or the reveal corrupt oracle depending on the input to the oracle being one of the honest keys or an adversarially chosen (corrupted) one. The challenge oracle is simulated with a test query to the first and second honest public keys. In case $b = 0$ of the **CKS** game, reduction $\mathcal{B}$ is simulating Game G_2, in case $b = 1$ it is exactly Game G_3. Counting the queries yields the stated bound. ■

Game G_4 Game G_4 is the same as G_3 except that the output of H_1 is replaced in the encapsulation or decapsulation oracle by a uniformly random value of the output space $\mathcal{K}_{\mathsf{H}_1}$ if the input NIKE public key, npk equals the ephemeral challenge key $npk_e^\star$.

Claim. There exists an adversary $\mathcal{C}$ against the **PRF** security of H_1 such that

$$\left|\Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(1,1)\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{C}}.$$

Proof. In case condition $npk = npk_e^\star$ holds, the first shared key, nk', always equals $k^\star$. Since this is a uniformly random value, we can reduce to the **PRF** security of H_1 with only one PRF key. Hence, adversary $\mathcal{C}$ can simulate the whole game and querying their own `Eval` oracle once on `"auth"`. This value can then be used to answer encapsulation and decapsulation queries for which the condition holds. Note that this only requires one evaluation query because the input to the query, `"auth"`, is fixed. ■

$\mathcal{B}^{\mathtt{RevCor},\mathtt{Test}}(npk_1,\ldots,npk_{Q_{\mathtt{Enc}}+2})$

$\mathcal{D} := \emptyset$
$(ksk^\star, kpk^\star) \xleftarrow{\$} \mathsf{KEM.Gen}$
$(ssk^\star, spk^\star) \xleftarrow{\$} \mathsf{RSig.Gen}$
$sk^\star := (\bot, ksk^\star, ssk^\star)$
$pk^\star := (npk_1, kpk^\star, spk^\star)$ // use first key for user key
$npk^\star := npk_1$
$npk_e^\star := npk_2$ // use second key for ephemeral challenge key
$\ell := 2$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathtt{Encps},\mathtt{Decps},\mathtt{Chall}}(pk^\star)$
return $[\![b = b']\!]$

Oracle $\mathtt{Encps}(pk)$

$\ell := \ell + 1$
parse $pk \to (npk, kpk, spk)$
$npk_e := npk_\ell$ // use next honest key
if $npk = npk_e^\star$
  $nk' \leftarrow \mathtt{Test}(1,2)$ // Test query
  $nk_1 \| nk_2 \leftarrow \mathtt{Test}(\ell, 2)$ // Test query
elseif $\exists\, i : npk = npk_i$
  $nk' \leftarrow \mathtt{Test}(1,i)$
  $nk_1 \| nk_2 \leftarrow \mathtt{Test}(\ell, i)$
else
  $nk' \xleftarrow{\$} \mathtt{RevCor}(1, npk)$ // RevCor query
  $nk_1 \| nk_2 \xleftarrow{\$} \mathtt{RevCor}(\ell, npk)$ // RevCor query
$nk := \mathsf{H}_1(nk', \texttt{"auth"})$
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
$m \leftarrow (kct, kpk)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk^\star, \{spk^\star, spk\}, m)$
$k' := \mathsf{H}_1(nk_1, kk_1)$
$sct := \mathsf{Sym.Enc}(k', \sigma)$
$c := (npk_e, kct, sct)$
$k := \mathsf{H}_2(nk, nk_2, kk_2, c, pk^\star, pk)$
return (c, k)

Oracle $\mathtt{Decps}(pk, c)$

if $\exists\, k : (pk, c, k) \in \mathcal{D}$
  return k
parse $pk \to (npk, kpk, spk)$
parse $c \to (npk_e, kct, sct)$
if $npk = npk_e^\star$
  $nk' \leftarrow \mathtt{Test}(1,2)$ // Test query
elseif $\exists\, i : npk = npk_i$
  $nk' \leftarrow \mathtt{Test}(1,i)$ // Test query
else
  $nk' \leftarrow \mathtt{RevCor}(1, npk)$ // RevCor query
$nk := \mathsf{H}_1(k_1', \texttt{"auth"})$
if $npk_e = npk_e^\star$
  $nk_1 \| nk_2 \leftarrow \mathtt{Test}(1,2)$ // Test query
elseif $\exists\, i : npk_e = npk_i$
  $nk_1 \| nk_2 \leftarrow \mathtt{Test}(1,i)$ // Test query
else
  $nk_1 \| nk_2 \leftarrow \mathtt{RevCor}(1, npk_e)$ // RevCor query
$kk_1 \| kk_2 \leftarrow \mathsf{KEM.Dec}(ksk^\star, kct)$
$k' := \mathsf{H}_1(nk_1, kk_1)$
$\sigma := \mathsf{Sym.Dec}(k', sct)$
$m \leftarrow (kct, kpk^\star)$
if $\mathsf{RSig.Ver}(\sigma, \rho = \{spk, spk^\star\}, m) \neq 1$
  return $\bot$
$k := \mathsf{H}_2(nk, nk_2, kk_2, c, pk, pk^\star)$
return k

Oracle $\mathtt{Chall}(sk)$ // one query

parse $sk \to (nsk, ksk, ssk)$
$npk_e := npk_e^\star$ // use second honest key
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk, npk^\star)$
$nk := \mathsf{H}_1(nk', \texttt{"auth"})$
$nk_1 \| nk_2 \xleftarrow{\$} \mathtt{Test}(2,1)$ // Test query for $(npk_e^\star, npk^\star)$
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk^\star)$
$m \leftarrow (kct, kpk^\star)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk, \{\mu(ssk), spk^\star\}, m)$
$k' := \mathsf{H}_1(nk_1, kk_1)$
$sct := \mathsf{Sym.Enc}(k', \sigma)$
$c := (npk_e, kct, sct)$
$k := \mathsf{H}_2(nk, nk_2, kk_2, c, \mu(sk), pk^\star)$
if $b = 1$
  $k \xleftarrow{\$} \mathcal{K}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), c, k)\}$
return (c, k)

Figure 26. Adversary $\mathcal{B}$ against **CKS** security of NIKE, having access to oracles `RevCor` and `Test`, simulating Game $\mathsf{G}_2/\mathsf{G}_3$ for adversary $\mathcal{A}$ from the proof of Theorem 14.

Game G_5 Game G_5 is the same as G_4 except that the output of keyed function H_2 in the challenge oracle is replaced by a uniformly sampled element of the output space. The same holds for the output of H_2 in the decapsulation oracle in case $npk_e = npk_e^\star$.

Claim. There exists an adversary $\mathcal{D}_1$ against the **PRF** security of H_2 such that

$$\left|\Pr\left[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(1,Q_{\mathtt{Dec}}+1)\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{D}_1}.$$

Proof. Adversary $\mathcal{D}_1$ is constructed in Figure 27 and keys function H_2 on nk_2. They need one PRF key for the challenge query which is $k_2^\star$. Note that even though $k_2^\star$ is used possibly several times during the experiment, the game can still be simulated due to the changes in the previous game. There might be the need of multiple evaluation queries since the same key can be queried again in the decapsulation oracle.

Note that the queries always need the same PRF key which is also guaranteed by condition $npk_e = npk_e^\star$ in the decapsulation oracle. In case $b = 0$ of the PRF game, adversary $\mathcal{D}_1$ obviously simulates Game G_4 for adversary $\mathcal{A}$. The simulation of Game G_5 in case $b = 1$ is sound if the evaluation oracle is not queried twice on the same input. For two queries from the decapsulation oracle this is not a problem because G_5 checks if there already was such a query and assigns the previous input and this case. The case that a query from the challenge and one from the decapsulation oracle have the same inputs cannot happen as well because a decapsulation query would not reach the PRF evaluation query for the same input again because it would return in Line 44 due to the fact that same inputs to H_2 implies the existence of an element in set $\mathcal{D}$. ■

Game G_6 Game G_6 is the same as G_2 (note that this is not build upon the previous game) except that the output of the KEM encapsulation in the challenge oracle is replaced by a uniformly random KEM key of the key space $\mathcal{K}_{\mathsf{KEM}}$. Further, if the decapsulation oracle is queried on a ciphertext for which the KEM component, kct, is the same as the one output by the challenge oracle, the same KEM key $kk^\star$ is assigned.

Claim. There exists an adversary $\mathcal{E}$ against the **IND-CCA** security of KEM such that

$$\left|\Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_6^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(1,\mathrm{Q_{Dec}},1)\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{E}}.$$

Proof. The reduction queries their own challenge oracle to simulate the AKEM challenge oracle. To answer decapsulation queries, they can use their own KEM decapsulation oracle. Thus, $\mathcal{E}$ simulates G_2 if they are in their own real game, i.e. $b = 0$, because they output the real encapsulation in the challenge oracle. In their case $b = 1$, they simulate Game G_6 because their own challenge is a uniformly random sample. ■

Game G_7 Game G_7 is the same as G_6 except that the output of keyed function H_2 in the challenge oracle is replaced by a uniformly sampled element of the output space.

Claim. There exists an adversary $\mathcal{D}_2$ against the **PRF** security of H_2 such that

$$\left|\Pr\left[\mathsf{G}_6^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_7^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(1,\mathrm{Q_{Dec}}+1)\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{D}_2}.$$

Proof. The claim can be proved analogously to the one for G_5 but choosing kk_2 as the PRF key instead. ■

We can see that the output distribution of the challenge oracle in Game G_5 and Game G_7 is the same for $b = 0$ and $b = 1$, thus we obtain

$$\Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G}_7^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Collecting the bounds for Games $\mathsf{G}_0 - \mathsf{G}_5$ and Games $\mathsf{G}_0 - \mathsf{G}_2, \mathsf{G}_6 - \mathsf{G}_7$ gives an upper bound on the single-user-single-challenge **Ins-CCA** game. Using a generic result from [ABH+21], we obtain the stated bound for the multi-user-multi-challenge setting. ■

Theorem 15 (Authenticity). *For any* **Out-Aut** *adversary* $\mathcal{A}$ *against* $\Pi := \mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 9, there exists an* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, a* **PRF** *adversary* $\mathcal{C}$ *against* H_1*, an* **PRF** *adversary* $\mathcal{D}$ *against* H_2*, a* **UF-CRA1** *adversary* $\mathcal{E}$ *against* RSig*, and an* **IND-CCA** *adversary* $\mathcal{F}$ *against* KEM *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}} \approx t_{\mathcal{C}} \approx t_{\mathcal{D}} \approx t_{\mathcal{E}} \approx t_{\mathcal{F}}$ *such that*

$$\mathrm{Adv}^{Q\text{-}\mathbf{Out\text{-}Aut}}_{\Pi,\mathcal{A}} \leq \min\left\{2 \cdot \mathrm{Adv}^{Q_{\mathsf{NIKE}}\text{-}\mathbf{CKS}}_{\mathsf{NIKE},\mathcal{B}} + 2 \cdot \mathrm{Adv}^{(n^2,n^2)\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{C}},\ \mathrm{Adv}^{(n,2,\mathrm{Q_{Enc}})\text{-}\mathbf{UF\text{-}CRA1}}_{\mathsf{RSig},\mathcal{E}} + 2 \cdot \mathrm{Adv}^{Q\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{F}} + Q_{Enc}^2 \cdot \gamma_{\mathsf{KEM}}\right\}$$
$$+ 2 \cdot \mathrm{Adv}^{(Q',Q')\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{D}} + Q_{Chl} \cdot \delta_{\Pi} + Q_{Enc} \cdot Q' \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}}$$

with $Q = (n, Q_{Enc}, Q_{Chl}), Q_{\mathsf{NIKE}} = (Q_{Enc} + 2Q_{Chl}, Q_{Enc} + 2Q_{Chl}), Q' = Q_{Enc} + Q_{Chl}$.

Proof. Consider the sequence of games depicted in Figure 28 and Figure 29.

$\mathcal{D}^{\texttt{Eval}}$

$\mathcal{D}, \mathcal{H} := \emptyset$
$kct^\star, kk^\star := \bot$
for $\ell \in [Q_{\texttt{Enc}} + 2]$
  $(nsk_\ell, npk_\ell) \xleftarrow{\$} \mathsf{NIKE.Gen}$
$(nsk^\star, npk^\star) := (nsk_1, npk_1)$
$(nsk_e^\star, npk_e^\star) := (nsk_2, npk_2)$
for $i \in [Q_{\texttt{Enc}} + 2]$
  $k_{ii} := \bot$
  for $j \in [i+1, Q_{\texttt{Enc}} + 2]$
    $k_{ij} := k_{ji} \xleftarrow{\$} \mathcal{K}_{\mathsf{NIKE}}$
$k^\star := k_{12}$
$k_{\mathsf{H}_1} \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_1}$
$\ell := 2$
$(ksk^\star, kpk^\star) \xleftarrow{\$} \mathsf{KEM.Gen}$
$(ssk^\star, spk^\star) \xleftarrow{\$} \mathsf{RSig.Gen}$
$sk^\star := (nsk^\star, ksk^\star, ssk^\star)$
$pk^\star := (npk^\star, kpk^\star, spk^\star)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\texttt{Encps},\texttt{Decps},\texttt{Chall}}(pk^\star)$
return $[\![b = b']\!]$

Oracle $\texttt{Chall}(sk)$ // one query

parse $sk \to (nsk, ksk, ssk)$
$(nsk_e, npk_e) := (nsk_e^\star, npk_e^\star)$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk, npk^\star)$
$nk := \mathsf{H}_1(nk', \texttt{"auth"})$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk^\star)$
$nk_2 := \star$ // key unknown
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk^\star)$
$(kct^\star, kk^\star) := (kct, kk_1 \| kk_2)$
$m \leftarrow (kct, kpk^\star)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk, \{\mu(ssk), spk^\star\}, m)$
$k' := \mathsf{H}_1(nk_1, kk_1)$
$sct := \mathsf{Sym.Enc}(k', \sigma)$
$c := (npk_e, kct, sct)$
if $\exists\, k' : (k', nk, nk_2, kk_2, c, \mu(sk), pk^\star) \in \mathcal{H}$
  abort
$k \xleftarrow{\$} \texttt{Eval}(1, nk \| kk_2 \| c \| \mu(sk) \| pk^\star)$ // eval query
$\mathcal{H} := \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, \mu(sk), pk^\star)\}$
if $b = 1$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), c, k)\}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), c, k)\}$
return (c, k)

Oracle $\texttt{Decps}(pk, c)$

if $\exists\, k : (pk, c, k) \in \mathcal{D}$
  return k
parse $pk \to (npk, kpk, spk)$
parse $c \to (npk_e, kct, sct)$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk^\star, npk)$
if $npk = npk_e^\star$
  $nk' := \bot$ // key unknown
elseif $\exists\, i : npk = npk_i$
  $nk' := k_{1i}$
$nk := \mathsf{H}_1(nk', \texttt{"auth"})$
if $npk = npk_e^\star$
  $nk := k_{\mathsf{H}_1}$ // key can be simulated
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk^\star, npk_e)$
if $npk_e = npk_e^\star$
  $nk_2 := \star$ // key unknown
elseif $\exists\, i : npk_e = npk_i$
  $nk_1 \| nk_2 := k_{1i}$
$kk_1 \| kk_2 \leftarrow \mathsf{KEM.Dec}(ksk^\star, kct)$
$k' := \mathsf{H}_1(nk_1, kk_1)$
$\sigma := \mathsf{Sym.Dec}(k', sct)$
$m \leftarrow (kct, kpk^\star)$
if $\mathsf{RSig.Ver}(\sigma, \rho = \{spk, spk^\star\}, m) \neq 1$
  return $\bot$
$k := \mathsf{H}_2(nk, nk_2, kk_2, c, pk, pk^\star)$
if $npk_e = npk_e^\star$
  $k \xleftarrow{\$} \texttt{Eval}(1, nk \| kk_2 \| c \| pk \| pk^\star)$ // eval query
$\mathcal{H} := \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk, pk^\star)\}$
return k

Oracle $\texttt{Encps}(pk)$

return $\mathsf{G}_4.\texttt{Encps}(pk)$

Figure 27. Adversary $\mathcal{C}_1$ against **PRF** security of H_2 queried on the second input, having access to oracle `Eval`, simulating Game $\mathsf{G}_4/\mathsf{G}_5$ for adversary $\mathcal{A}$ from the proof of Theorem 14.

Game G_0 We start with the **Out-Aut** game for $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$.

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n, Q_{\texttt{Enc}}, Q_{\texttt{Chl}})\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{Sym},\mathsf{H}_1,\mathsf{H}_2],\mathcal{A}}.$$

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

$\mathcal{D}, \mathcal{D}_1, \mathcal{H}, \mathcal{H}' := \emptyset$
$\mathtt{BAD}_1 := \mathbf{false}$
for $i \in [n]$
  $(nsk_i, npk_i) \xleftarrow{\$} \mathsf{NIKE.Gen}$
  $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
  $sk_i := (nsk_i, ksk_i, ssk_i)$
  $pk_i := (npk_i, kpk_i, spk_i)$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\mathtt{Encps},\mathtt{Chall}}(pk_1, \ldots, pk_n)$
return $[\![b = b']\!]$

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

parse $pk \to (npk, kpk, spk)$
$(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_s, npk)$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk)$
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
$m \leftarrow (kct, kpk)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk\}, m)$
$k' := \mathsf{H}_1(nk_1, kk_1)$
$sct := \mathsf{Sym.Enc}(k', \sigma)$
$c := (npk_e, kct, sct)$
if $\exists\, \hat{nk} : (\hat{nk}, \{npk_s, npk\}) \in \mathcal{D}_1$ // $\mathsf{G}_4 - \mathsf{G}_6$
  $nk' := \hat{nk}$ // $\mathsf{G}_4 - \mathsf{G}_6$
elseif $npk \in \{npk_1, \ldots, npk_n\}$ // $\mathsf{G}_4 - \mathsf{G}_6$
  $nk' \xleftarrow{\$} \mathcal{K}_{\mathsf{NIKE}}$ // $\mathsf{G}_4 - \mathsf{G}_6$
  $\mathcal{D}_1 := \mathcal{D}_1 \cup \{(nk', \{npk_s, npk\})\}$ // $\mathsf{G}_4 - \mathsf{G}_6$
$nk := \mathsf{H}_1(nk', \mathtt{"auth"})$
if $\exists\, \hat{nk} : (\hat{nk}, \{npk_s, npk\}) \in \mathcal{H}'$ // $\mathsf{G}_5 - \mathsf{G}_6$
  $nk := \hat{nk}$ // $\mathsf{G}_5 - \mathsf{G}_6$
elseif $npk \in \{npk_1, \ldots, npk_n\}$ // $\mathsf{G}_5 - \mathsf{G}_6$
  $nk \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_1}$ // $\mathsf{G}_5 - \mathsf{G}_6$
  $\mathcal{H}' := \mathcal{H}' \cup \{(nk, \{npk_s, npk\})\}$ // $\mathsf{G}_5 - \mathsf{G}_6$
if $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk_s, pk) \in \mathcal{H}$ // $\mathsf{G}_2 - \mathsf{G}_6$
  $\mathtt{BAD}_1$; **abort** // $\mathsf{G}_2 - \mathsf{G}_6$
$k := \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk)$
if $npk \in \{npk_1, \ldots, npk_n\}$
  $k \xleftarrow{\$} \mathcal{K}$ // G_6
$\mathcal{H} := \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk_s, pk)\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

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

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
parse $pk \to (npk, kpk, spk)$
parse $c \to (npk_e, kct, sct)$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk)$
$nk := \mathsf{H}_1(k'_1, \mathtt{"auth"})$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_e)$
if $\exists\, \hat{nk} : (\hat{nk}, \{npk_r, npk_e\}) \in \mathcal{D}_1$ // $\mathsf{G}_4 - \mathsf{G}_6$
  $nk_1 \| nk_2 := \hat{nk}$ // $\mathsf{G}_4 - \mathsf{G}_6$
elseif $npk_e \in \{npk_1, \ldots, npk_n\}$ // $\mathsf{G}_4 - \mathsf{G}_6$
  $nk_1 \| nk_2 \xleftarrow{\$} \mathcal{K}_{\mathsf{NIKE}}$ // $\mathsf{G}_4 - \mathsf{G}_6$
$kk_1 \| kk_2 \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
$k' := \mathsf{H}_1(nk_1, kk_1)$
$\sigma := \mathsf{Sym.Dec}(k', sct)$
$m \leftarrow (kct, kpk_r)$
if $\mathsf{RSig.Ver}(\sigma, \{spk, spk_r\}, m) \neq 1$
  return $\perp$
if $\exists\, \hat{nk} : (\hat{nk}, \{npk_s, npk\}) \in \mathcal{D}_1$ // $\mathsf{G}_4 - \mathsf{G}_6$
  $nk' := \hat{nk}$ // $\mathsf{G}_4 - \mathsf{G}_6$
elseif $npk \in \{npk_1, \ldots, npk_n\}$ // $\mathsf{G}_4 - \mathsf{G}_6$
  $nk' \xleftarrow{\$} \mathcal{K}_{\mathsf{NIKE}}$ // $\mathsf{G}_4 - \mathsf{G}_6$
  $\mathcal{D}_1 := \mathcal{D}_1 \cup \{(nk', \{npk, npk_r\})\}$ // $\mathsf{G}_4 - \mathsf{G}_6$
$nk := \mathsf{H}_1(nk', \mathtt{"auth"})$
if $\exists\, \hat{nk} : (\hat{nk}, \{npk, npk_r\}) \in \mathcal{H}'$ // $\mathsf{G}_5 - \mathsf{G}_6$
  $nk := \hat{nk}$ // $\mathsf{G}_5 - \mathsf{G}_6$
elseif $npk \in \{npk_1, \ldots, npk_n\}$ // $\mathsf{G}_5 - \mathsf{G}_6$
  $nk \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_1}$ // $\mathsf{G}_5 - \mathsf{G}_6$
  $\mathcal{H}' := \mathcal{H}' \cup \{(nk, \{npk, npk_r\})\}$ // $\mathsf{G}_5 - \mathsf{G}_6$
if $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk, pk_r) \in \mathcal{H}$ // $\mathsf{G}_3 - \mathsf{G}_6$
  abort // $\mathsf{G}_3 - \mathsf{G}_6$
$k := \mathsf{H}_2(nk, nk_2, kk_2, c, pk, pk_r)$
if $npk \in \{npk_1, \ldots, npk_n\}$
  $k \xleftarrow{\$} \mathcal{K}$ // G_6
$\mathcal{H} := \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk, pk_r)\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
if $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \perp$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$ // $\mathsf{G}_1 - \mathsf{G}_6$
return k

Figure 28. Games $\mathsf{G}_0 - \mathsf{G}_6$ for the proof of Theorem 15.

Game G_1 This is the same as G_0 except that in the challenge oracle an element is added to $\mathcal{D}$ independent of challenge bit b. Further, we introduce a set $\mathcal{H}$ to store the output as well as all the inputs for every query on H_2. If the scheme is perfectly correct, the changes cannot be distinguished since the difference is that $\mathcal{D}$ stores either tuples from encapsulations or from correct decapsulations. Hence, the difference is at most the correctness error per query to the challenge oracle:

$$\left|\Pr\left[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq Q_{\mathtt{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{Sym},\mathsf{H}_1,\mathsf{H}_2]}.$$

Games $\mathsf{G}_3, \mathsf{G}_7 - \mathsf{G}_{10}$

$\mathcal{D}, \mathcal{D}_2\mathcal{H}, \mathcal{Q} \coloneqq \emptyset$
$\mathtt{BAD}_2, \mathtt{BAD}_3 \coloneqq \mathbf{false}$
for $i \in [n]$
  $(nsk_i, npk_i) \xleftarrow{\$} \mathsf{NIKE.Gen}$
  $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
  $sk_i \coloneqq (nsk_i, ksk_i, ssk_i)$
  $pk_i \coloneqq (npk_i, kpk_i, spk_i)$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\mathtt{Encps},\mathtt{Chall}}(pk_1, \ldots, pk_n)$
return $[\![b = b']\!]$

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

parse $pk \to (npk, kpk, spk)$
$(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
$k'_1 \leftarrow \mathsf{NIKE.Sdk}(nsk_s, npk)$
$k_1 \coloneqq \mathsf{H}_1(k'_1, \texttt{"auth"})$
$k'_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk)$
$(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
if $kpk \in \{kpk_1, \ldots, kpk_n\}$ // $\mathsf{G}_9 - \mathsf{G}_{10}$
  $kk \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$ // $\mathsf{G}_9 - \mathsf{G}_{10}$
  $\mathcal{D}_2 \coloneqq \mathcal{D}_2 \cup \{(kpk, kct, kk)\}$ // $\mathsf{G}_9 - \mathsf{G}_{10}$
$m \leftarrow (kct, kpk)$
if $(\{spk_s, spk\}, m, \cdot) \in \mathcal{Q}$ // $\mathsf{G}_7 - \mathsf{G}_{10}$
  $\mathtt{BAD}_2 \coloneqq \mathbf{true}; \mathbf{abort}$ // $\mathsf{G}_7 - \mathsf{G}_{10}$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{\mu(ssk_s), spk\}, m)$
$\mathcal{Q} \coloneqq \mathcal{Q} \cup \{(\{spk_s, spk\}, m, \sigma)\}$ // $\mathsf{G}_7 - \mathsf{G}_{10}$
$k' \coloneqq \mathsf{H}_1(k_2, kk)$
$sct \coloneqq \mathsf{Sym.Enc}(k', \sigma)$
$c \coloneqq (npk_e, kct, sct)$
if $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk, pk_r) \in \mathcal{H}$
  abort
$k \coloneqq \mathsf{H}_2(k_1, k_2, kk, c, pk_s, pk)$
if $kpk \in \{kpk_1, \ldots, kpk_n\}$ // G_{10}
  $k \xleftarrow{\$} \mathcal{K}$ // G_{10}
$\mathcal{H} \coloneqq \mathcal{H} \cup \{(k, k_1, k_2, kk, c, pk_s, pk)\}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

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

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
parse $pk \to (npk, kpk, spk)$
parse $c \to (npk_e, kct, sct)$
$k'_1 \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk)$
$k_1 \coloneqq \mathsf{H}_1(k'_1, \texttt{"auth"})$
$k'_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_e)$
$kk \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
if $\exists\, kk' : (kpk_r, kct, kk') \in \mathcal{D}_2$ // $\mathsf{G}_9 - \mathsf{G}_{10}$
  $kk \coloneqq kk'$ // $\mathsf{G}_9 - \mathsf{G}_{10}$
$k' \coloneqq \mathsf{H}_1(k_2, kk)$
$\sigma \coloneqq \mathsf{Sym.Dec}(k', sct)$
$m \leftarrow (kct, kpk_r)$
$k \coloneqq \mathsf{H}_2(k_1, k_2, kk, c, pk, pk_r)$
if $\mathsf{RSig.Ver}(\sigma, \{spk, spk_r\}, m) \neq 1$
  return $\perp$
elseif $\exists\, i : spk = spk_i \wedge (\{spk, spk_r\}, m, \cdot) \notin \mathcal{Q}$ // $\mathsf{G}_8 - \mathsf{G}_{10}$
  $\mathtt{BAD}_3 \coloneqq \mathbf{true}; \mathbf{abort}$ // $\mathsf{G}_8 - \mathsf{G}_{10}$
if $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk, pk_r) \in \mathcal{H}$
  abort
if $spk \in \{spk_1, \ldots, spk_n\} \wedge k \neq \perp$ // G_{10}
  $k \xleftarrow{\$} \mathcal{K}$ // G_{10}
$\mathcal{H} \coloneqq \mathcal{H} \cup \{(k, k_1, k_2, kk, c, pk, pk_r)\}$
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 29. Games $\mathsf{G}_3, \mathsf{G}_7 - \mathsf{G}_{10}$ for the proof of Theorem 15.

Game G_2 This game is the same as G_1 except that the game aborts in the encapsulation oracle if there already exists an element in set $\mathcal{H}$ with the same inputs.

Claim.

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq Q_{\mathtt{Enc}} \cdot (Q_{\mathtt{Enc}} + Q_{\mathtt{Chl}}) \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}}.$$

Proof. If there was a previous query to H_2 on the same inputs, this includes ciphertext c. Part of the ciphertext is the ephemeral NIKE key npk_e and the KEM ciphertext kct. For one element in $\mathcal{H}$, the probability that these two values are the same is at most $\eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}}$. Since for each query to $\mathtt{Encps}$ and $\mathtt{Chall}$ at most one element is added to $\mathcal{H}$, we obtain the claimed bound. ■

Game G_3 This game is the same as G_2 except that the game aborts in the challenge oracle if there already exists an element in set $\mathcal{H}$ with the same inputs.

Claim.

$$\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1].$$

Proof. If there was a previous query to H_2 on the same inputs, this includes ciphertext c and the public keys pk and pk_r which must be the same in the previous query. However, this implies that there is also a corresponding element in $\mathcal{D}$ and the challenge oracle would have aborted in Line 38. ■

Game G_4 This is the same as G_3 except that NIKE shared key nk' is replaced by a uniformly random value of the key space $\mathcal{K}_{\mathsf{NIKE}}$ and stored together with the two corresponding public keys in set $\mathcal{D}_1$. For an encapsulation query this is only done in the case of an honest receiver. In case the shared key between two parties was already computed before, it is taken from set $\mathcal{D}_1$.

Claim. There exists an adversary $\mathcal{B}$ against the **CKS** security of NIKE such that

$$\left|\Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(\mathtt{Q_{Enc}}+2\mathtt{Q_{Chl}},\mathtt{Q_{Enc}}+2\mathtt{Q_{Chl}})\text{-}\mathbf{CKS}}_{\mathsf{NIKE},\mathcal{B}}.$$

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 30. The encapsulation oracle can be simulated by either making a test or a corrupt reveal query depending on the receiver key pk being honest (test query) or dishonest (corrupt reveal query). The same needs to be done in the challenge oracle but we need an additional test or reveal corrupt query for the second NIKE key, $nk_1 \| nk_2$, since the adversary can input honest NIKE keys as part of the ciphertext. Depending on the challenge bit of the NIKE adversary $\mathcal{B}$, they simulate either Game G_3 or Game G_4. There is at most one test or corrupt reveal per query to `Encps` and at most two test or reveal corrupt queries per query to $Q_{\mathtt{Chl}}$.

■

Game G_5 This game is the same as G_4 except that the output of keyed function H_1 in the encapsulation oracle (challenge oracle resp.) is replaced by a uniformly sampled value from the domain $\mathcal{K}_{\mathsf{H}_1}$ if the NIKE public of the receiver (sender resp.) is honest (Line 31, Line 31 resp.). If there was a query on the same inputs before, this value is taken instead.

Claim. There exists an adversary $\mathcal{C}$ against the **PRF** security of H_1 such that

$$\left|\Pr\left[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(n^2,n^2)\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{C}}.$$

Proof. Due to the changes in the previous game, the first NIKE shared key, nk', is uniformly random for honest public keys. Note that in the case where we take a stored shared key, we also have an element in $\mathcal{H}'$ and also take a previously stored output because the parameters are matching. This ensures that PRF evaluation queries to the same PRF key and input correctly simulate the games. There are up to $Q_{\mathtt{Enc}} + Q_{\mathtt{Chl}}$ many keys and evaluation queries. However, there is at most one query per key since the input is always the same and there at most $\binom{n}{2} \leq n^2$ many keys since the derivation of a shared NIKE key is deterministic. ■

Game G_6 This game is the same as G_5 except that the output of the keyed function H_2 in the encapsulation oracle (challenge oracle resp.) is replaced by a uniformly sampled value from the domain $\mathcal{K}$ if the NIKE public of the receiver (sender resp.) is honest (Line 37, Line 73 resp.).

Claim. There exists an adversary $\mathcal{D}_1$ against the **PRF** security of H_2 such that

$$\left|\Pr\left[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_6^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(\mathtt{Q_{Enc}}+\mathtt{Q_{Chl}},\mathtt{Q_{Enc}}+\mathtt{Q_{Chl}})\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{D}_1}.$$

Proof. Adversary $\mathcal{D}_1$ is formally constructed in Figure 31 choosing the first component as their PRF key. The reduction needs at most one PRF key per encapsulation and challenge query. The same holds for the evaluation queries. In case $b = 0$ of the PRF game, adversary $\mathcal{D}_1$ simulates Game G_5 for adversary $\mathcal{A}$. In case $b = 1$ of the PRF game, they simulate Game G_6. Since the evaluation oracle is never queried on the same input twice (since the game aborts otherwise), the simulation of Game G_6 (outputting uniformly random values in each query) is sound.

■

$\mathcal{B}^{\texttt{RevCor},\texttt{Test}}(npk_1,\ldots,npk_n)$

$\mathcal{D},\mathcal{D}_1,\mathcal{H} \coloneqq \emptyset$
for $i \in [n]$
  $(ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
  $(ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
  $sk_i \coloneqq (\bot, ksk_i, ssk_i)$
  $pk_i \coloneqq (npk_i, kpk_i, spk_i)$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\texttt{Encps},\texttt{Chall}}(pk_1,\ldots,pk_n)$
return $[\![b = b']\!]$

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

parse $pk \to (npk, kpk, spk)$
$(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk)$
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
$m \leftarrow (kct, kpk)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{\mu(ssk_s), spk\}, m)$
$k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
$sct \coloneqq \mathsf{Sym.Enc}(k', \sigma)$
$c \coloneqq (npk_e, kct, sct)$
if $\exists\, \hat{nk} : (\hat{nk}, \{npk_s, npk\}) \in \mathcal{D}_1$
  $nk' \coloneqq \hat{nk}$
elseif $\exists\, r : npk = npk_r$
  $nk' \xleftarrow{\$} \texttt{Test}(s, r)$ // test query
  $\mathcal{D}_1 \coloneqq \mathcal{D}_1 \cup \{(nk', \{npk_s, npk\})\}$
else
  $nk' \xleftarrow{\$} \texttt{RevCor}(s, npk)$ // corrupt reveal query
$nk \coloneqq \mathsf{H}_1(nk', \texttt{"auth"})$
$k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk)$
$\mathcal{H} \coloneqq \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk_s, pk)\}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

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

if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
parse $pk \to (npk, kpk, spk)$
parse $c \to (npk_e, kct, sct)$
if $\exists\, i : npk_e = npk_i$
  $nk_1 \| nk_2 \xleftarrow{\$} \texttt{Test}(r, i)$ // test query
else
  $nk_1 \| nk_2 \xleftarrow{\$} \texttt{RevCor}(r, npk_e)$ // corrupt reveal query
$kk_1 \| kk_2 \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
$k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
$\sigma \coloneqq \mathsf{Sym.Dec}(k', sct)$
$m \leftarrow (kct, kpk_r)$
if $\mathsf{RSig.Ver}(\sigma, \{spk, spk_r\}, m) \neq 1$
  return $\bot$
if $\exists\, \hat{nk} : (\hat{nk}, \{npk, npk_r\}) \in \mathcal{D}_1$
  $nk' \coloneqq \hat{nk}$
elseif $\exists\, s : npk = npk_s$
  $nk' \xleftarrow{\$} \texttt{Test}(r, s)$ // test query
else
  $nk' \xleftarrow{\$} \texttt{RevCor}(r, pk)$ // corrupt reveal query
$nk \coloneqq \mathsf{H}_1(nk', \texttt{"auth"})$
$k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk, pk_r)$
$\mathcal{H} \coloneqq \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk, pk_r)\}$
if $b = 1 \wedge pk \in \{pk_1,\ldots,pk_n\} \wedge k \neq \bot$
  $k \xleftarrow{\$} \mathcal{K}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
return k

Figure 30. Adversary $\mathcal{B}$ against **CKS** security of NIKE, having access to oracles `RevCor` and `Test`, simulating Game $\mathsf{G}_3/\mathsf{G}_4$ for adversary $\mathcal{A}$ from the proof of Theorem 15.

Game G_7 This is the same as G_3 (note that this does not build upon the previous game) except that flag $\texttt{BAD}_2$ is set to **true** and the game aborts if the same message m is signed twice. To keep track of the signing queries, we introduce set $\mathcal{Q}$ storing the ring, the message, and the output signature.

Claim.

$$\left|\Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_7^{\mathcal{A}} \Rightarrow 1\right]\right| \leq Q_{\texttt{Enc}}^2 \cdot \gamma_{\mathsf{KEM}}.$$

Proof. The message being signed in the encapsulation oracle consists of several components where one of them is the KEM ciphertext kct. Hence, $\texttt{BAD}_2$ is only set to **true** if there is a collision in KEM ciphertexts. For one query and one element in set $\mathcal{Q}$ the probability is at most γ_{KEM}. Since there are at most $Q_{\texttt{Enc}}$ queries to the encapsulation oracle and at most the same number of elements in set $\mathcal{Q}$, it holds

$$\Pr[\texttt{BAD}_2 = \textbf{true}] \leq Q_{\texttt{Enc}}^2 \cdot \gamma_{\mathsf{KEM}}.$$

■

Game G_8 This game is the same as G_7 except that flag $\texttt{BAD}_3$ is set to **true** and the game aborts if the signature in the challenge oracle verifies, the sender signature public key is honest, and the ring/message was not input to a signing query before.

Figure 31. Adversary $\mathcal{D}_1$ against **PRF** security of H_2, having access to oracle $\texttt{Eval}$, simulating Game $\mathsf{G}_5/\mathsf{G}_6$ for adversary $\mathcal{A}$ from the proof of Theorem 15.

Claim. There exists an adversary $\mathcal{E}$ against the **UF-CRA1** security of RSig such that

$$\left|\Pr\left[\mathsf{G}_7^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_8^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{(n,2,Q_{\mathrm{Enc}})\text{-}\mathbf{UF\text{-}CRA1}}_{\mathsf{RSig},\mathcal{E}}.$$

Proof. Adversary $\mathcal{E}$ is formally constructed in Figure 32. The encapsulation oracle can be completely simulated since the game aborts if there was a signing query on the same message again and one of the

public keys in the ring is honest, namely spk_s. Further, adversary $\mathcal{E}$ wins the game if they return $(\sigma, \{spk_i, spk_r\}, m)$ in the challenge oracle: the output is valid (check in Line 42), was not subject to a signing query before (check in Line 44), and the challenge ring contains only honest users.

$\mathcal{E}^{\mathtt{Sgn}}(par, spk_1, \ldots, spk_n)$

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

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

10 **parse** $pk \to (npk, kpk, spk)$
11 $(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
12 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_s, npk)$
13 $nk \coloneqq \mathsf{H}_1(nk', \mathtt{"auth"})$
14 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk)$
15 $(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
16 $m \leftarrow (kct, kpk)$
17 **if** $(\{spk_s, spk\}, m, \cdot) \in \mathcal{Q}$
18 $\quad$ **abort**
19 $\sigma \leftarrow \mathsf{Sgn}(s, \{spk_s, spk\}, m)$ // signing query
20 $\mathcal{Q} \coloneqq \mathcal{Q} \cup \{(\{spk_s, spk\}, m, \sigma)\}$
21 $k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
22 $sct \coloneqq \mathsf{Sym.Enc}(k', \sigma)$
23 $c \coloneqq (npk_e, kct, sct)$
24 **if** $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk_s, pk) \in \mathcal{H}$
25 $\quad$ **abort**
26 $k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk)$
27 $\mathcal{H} \coloneqq \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk_s, pk)\}$
28 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
29 **return** (c, k)

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

30 **if** $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
31 $\quad$ **return** k
32 **parse** $pk \to (npk, kpk, spk)$
33 **parse** $c \to (npk_e, kct, sct)$
34 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk)$
35 $nk \coloneqq \mathsf{H}_1(nk', \mathtt{"auth"})$
36 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_e)$
37 $kk_1 \| kk_2 \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
38 $k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
39 $\sigma \coloneqq \mathsf{Sym.Dec}(k', sct)$
40 $m \leftarrow (kct, kpk_r)$
41 $k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk, pk_r)$
42 **if** $\mathsf{RSig.Ver}(\sigma, \{spk, spk_r\}, m) \neq 1$
43 $\quad$ **return** $\bot$
44 **elseif** $\exists\, i : spk = spk_i \wedge (\{spk, spk_r\}, m, \cdot) \notin \mathcal{Q}$
45 $\quad$ **return** $(\sigma, \{spk_i, spk_r\}, m)$ // return forgery
46 **if** $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk, pk_r) \in \mathcal{H}$
47 $\quad$ **abort**
48 $\mathcal{H} \coloneqq \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk, pk_r)\}$
49 **if** $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \bot$
50 $\quad k \xleftarrow{\$} \mathcal{K}$
51 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
52 **return** k

Figure 32. Adversary $\mathcal{E}$ against **UF-CRA1** security of RSig, having access to oracle Sgn, simulating Game $\mathsf{G}_7/\mathsf{G}_8$ for adversary $\mathcal{A}$ from the proof of Theorem 15.

■

Game G_9 This is the same as G_8 except that KEM key in the encapsulation oracle is replaced by a uniformly random output for honest receivers. The result is stored together with public key and ciphertext in set $\mathcal{D}_1$ to answer decapsulation calls in the challenge oracle consistently.

Claim. There exists an adversary $\mathcal{F}$ against the **IND-CCA** security of KEM such that

$$\left|\Pr\left[\mathsf{G}_8^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_9^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(n,\mathrm{Q}_{\mathtt{Enc}},\mathrm{Q}_{\mathtt{Chl}})\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{F}}.$$

Proof. Adversary $\mathcal{F}$ can simulate the encapsulation oracle oracle by querying their own challenge oracle for honest receiver keys. The challenge oracle can be simulated by a query to their own decapsulation oracle.

Thus, $\mathcal{F}$ simulates G_8 if they are in their own real game, i.e. $b = 0$, because they output the real encapsulation in the `Encps` oracle. In their case $b = 1$, they simulate Game G_9 because their own challenge is a uniformly random sample.

■

Game G_{10} This game is the same as G_9 except that the output of keyed function H_2 in the encapsulation and challenge oracle is replaced by a uniformly sampled value from the domain $\mathcal{K}$. For the encapsulation oracle this is only done if the KEM key of the receiver is honest and in the challenge oracle if the signature verification key of the sender, spk, is honest and the shared key k is not $\perp$.

Claim. There exists an adversary $\mathcal{D}_2$ against the **PRF** security of H_2 such that

$$\left|\Pr\left[\mathsf{G}_9^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{10}^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}_{\mathsf{H}_2,\mathcal{D}_2}^{(Q_{\mathtt{Enc}}+Q_{\mathtt{Chl}},Q_{\mathtt{Enc}}+Q_{\mathtt{Chl}})\text{-}\mathbf{PRF}}.$$

Proof. The claim can be proved analogously to the one from G_6 except that the reduction chooses the third element, kk_2, to be the PRF key. For the encapsulation oracle, the reduction is sound since a new KEM key is sampled uniformly for each query. It functions as the PRF key for the reduction and an index for that key can be stored in set $\mathcal{D}_2$. For the challenge oracle, this key can be reused and the PRF can be queried on the stored index. Note that the condition $spk \in \{spk_1, \ldots, spk_n\}$ implies that a random key from set $\mathcal{D}_2$ was taken: if Line 57 is reached, the game did not set flag $\mathtt{BAD}_3$ to **true** and abort. This means that the sender verification is dishonest or there existing a matching element in $\mathcal{Q}$, i.e. the message/public keys pair was signed before. Checking for honest sender verification key, leaves us with the second possibility. However, if there is a matching element in $\mathcal{Q}$ there must have been a corresponding query to `Encps` because $\mathcal{Q}$ is only filed there. Further, this query must have added an element to $\mathcal{D}_2$ because the receiver KEM key of such a query was honest because the challenge oracle can only be queried on honest receivers. It is also not possible to change the order of sender and receiver (which would yield at least the same ring) since the message being signed contains the KEM key of the receiver kpk/kpk_r. ■

We now analyse the winning probability of Games G_6 and G_{10}:

Claim.

$$\Pr[\mathsf{G}_6^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{G}_{10}^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Proof. For Game G_6, the shared key output by the challenge oracle is uniformly random in the case $b = 0$ if the sender NIKE key is honest. Case $b = 1$ only triggers for honest sender keys (and decapsulations $\neq \perp$). Since honest sender keys imply an honest sender NIKE key, the output distribution is the same for case $b = 0$ and $b = 1$ and thus independent of the challenge bit.

For Game G_{10} and $b = 0$, there are two things that can happen in the challenge oracle. First, the signature is not valid then the oracle returns $\perp$ in Line 52 which happens independent of the challenge bit. Second, for a valid signature the game either aborts or the oracle outputs a uniformly random key if spk is honest and $k \neq \perp$ (Line 57). These conditions are implied by the conditions which are necessary to trigger case $b = 1$ and are therefore true whenever case $b = 1$ could occur. Hence, the output distribution for case $b = 0$ and $b = 1$ does not differ and the game is independent of the challenge bit. ■

We conclude the proof by combining the bounds.

■

Theorem 16 (Dishonest Deniability). *For any* **DR-Den** *adversary* $\mathcal{A}$ *against* $\Pi := \mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 9, and simulator* Sim *as defined in Figure 33 there exists a* **MC-Ano** *adversary* $\mathcal{B}$ *against* RSig *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}}$ *such that*

$$\mathrm{Adv}_{\Pi,\mathsf{Sim},\mathcal{A}}^{(n,Q_{\mathcal{C}hl})\text{-}\mathbf{DR\text{-}Den}} \leq \mathrm{Adv}_{\mathsf{RSig},\mathcal{B}}^{(n,2,Q_{\mathcal{C}hl})\text{-}\mathbf{MC\text{-}Ano}} + Q_{\mathcal{C}hl} \cdot \delta_{\mathsf{NIKE}}.$$

Proof. Consider the sequence of games depicted in Figure 33 as well as the construction of a simulator Sim.

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

01 **for** $i \in [n]$
02 $\quad (nsk_i, npk_i) \xleftarrow{\$} \mathsf{NIKE.Gen}$
03 $\quad (ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
04 $\quad (ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
05 $\quad sk_i \coloneqq (nsk_i, ksk_i, ssk_i)$
06 $\quad pk_i \coloneqq (npk_i, kpk_i, spk_i)$
07 $b \xleftarrow{\$} \{0,1\}$
08 $b' \leftarrow \mathcal{A}^{\texttt{Chall}}((sk_1, pk_1), \ldots, (sk_n, pk_n))$
09 **return** $[\![b = b']\!]$

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

10 **if** $s = r$ **return** $\bot$
11 $(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
12 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_s, npk_r)$
13 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_s)$ // $\mathsf{G}_1 - \mathsf{G}_2$
14 $nk \coloneqq \mathsf{H}_1(nk', \texttt{"auth"})$
15 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk_r)$
16 $(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
17 $m \leftarrow (kct, kpk_r)$
18 $\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk_r\}, m)$
19 $\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_r, \{spk_s, spk_r\}, m)$ // G_2
20 $k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
21 $sct \coloneqq \mathsf{Sym.Enc}(k', \sigma)$
22 $c \coloneqq (npk_e, kct, sct)$
23 $k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk_r)$
24 **if** $b = 1$
25 $\quad (c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r, sk_r)$
26 **return** (c, k)

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

27 **parse** $pk_s \to (npk_s, kpk_s, spk_s)$
28 **parse** $pk_r \to (npk_r, kpk_r, spk_r)$
29 **parse** $sk_r \to (nsk_r, ksk_r, ssk_r)$
30 $(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
31 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_s)$
32 $nk \coloneqq \mathsf{H}_1(nk', \texttt{"auth"})$
33 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk_r)$
34 $(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
35 $m \leftarrow (kct, kpk_r)$
36 $\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_r, \{spk_s, spk_r\}, m)$
37 $k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
38 $sct \coloneqq \mathsf{Sym.Enc}(k', \sigma)$
39 $c \coloneqq (npk_e, kct, sct)$
40 $k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk_r)$
41 **return** (c, k)

Figure 33. Games $\mathsf{G}_0 - \mathsf{G}_2$ for the proof of Theorem 16.

Game G_0 We start with the dishonest receiver deniability game for $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$. Compared to the original definition in Figure 5, we remove the reveal oracle and directly provide the adversary with all the secret keys of the game since there is no restriction on revealing secret keys and thus these games are equivalent. Hence, it holds

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,\mathrm{Q}_{\mathtt{Chl}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{Sym},\mathsf{H}_1,\mathsf{H}_2],\mathsf{Sim},\mathcal{A}}.$$

Game G_1 Game G_1 is the same as G_0 except that the first NIKE shared key in the challenge oracle, nk', is computed between receiver and sender instead of sender and receiver.

Claim.

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

Proof. Since both the sender and receiver keys are honestly generated, the change in one query is exactly the definition of the correctness error. Applying the change for every query to `Chall` proves the claim. ■

Game G_2 Game G_2 is the same as G_1 except that the ring signature is computed with the receiver's signing key instead of the sender's signing key.

Claim. There exists an adversary $\mathcal{B}$ against **MC-Ano** security of RSig such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{(n,2,\mathrm{Q}_{\mathtt{Chl}})\text{-}\mathbf{MC\text{-}Ano}}_{\mathsf{RSig},\mathcal{B}}.$$

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 33. To compute the signature in the challenge oracle, $\mathcal{B}$ can query their own challenge oracle. In case $b = 0$, they simulate Game G_1, otherwise they simulate G_2. The number of challenge queries for the anonymity game equals the number for the deniability game of adversary $\mathcal{A}$.

■

Game G_2 is independent of challenge bit b since syntactically the same operations are executed in case $b = 0$ and $b = 1$:

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

■

Theorem 17 (Honest Deniability). *For any* **HR-Den** *adversary* $\mathcal{A}$ *against* $\Pi := \mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 9,and simulator* Sim *as defined in Figure 35 there exists a* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, an* **IND-CPA** *adversary* $\mathcal{C}$ *against* KEM*,* **PRF** *adversaries* $\mathcal{D}$ *and* $\mathcal{E}$ *against* H_1 *and* H_2*, and a* **IND-CPA** *adversary* $\mathcal{F}$ *against* Sym *with* $t_{\mathcal{A}} \approx t_{\mathcal{B}} \approx t_{\mathcal{C}} \approx t_{\mathcal{D}} \approx t_{\mathcal{E}} \approx t_{\mathcal{F}}$ *such that*

$$\mathrm{Adv}^{(n,\mathrm{Q}_{\mathtt{Chl}})\text{-}\mathbf{HR\text{-}Den}}_{\Pi,\mathsf{Sim},\mathcal{A}} \leq 4n^2 \cdot Q_{\mathtt{Chl}} \cdot \left(\min\left\{\mathrm{Adv}^{(2,0,1)\text{-}\mathbf{CKS}}_{\mathsf{NIKE},\mathcal{B}}, \mathrm{Adv}^{(1,1)\text{-}\mathbf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{C}}\right\} + \mathrm{Adv}^{(1,1)\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{D}} + \mathrm{Adv}^{(1,1)\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{E}} + \mathrm{Adv}^{\mathbf{IND\text{-}CPA}}_{\mathsf{Sym},\mathcal{F}}\right).$$

Proof. Consider the sequence of games depicted in Figure 35 as well as the construction of a simulator Sim.

Game G_0 We start with a simplified game for honest receiver deniability for $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{Sym}, \mathsf{H}_1, \mathsf{H}_2]$ considering only one challenge query and two users. Hence, it holds

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(2,1)\text{-}\mathbf{HR\text{-}Den}}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{Sym},\mathsf{H}_1,\mathsf{H}_2],\mathsf{Sim},\mathcal{A}}.$$

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

01 **for** $i \in [n]$
02 $\quad (nsk_i, npk_i) \xleftarrow{\$} \mathsf{NIKE.Gen}$
03 $\quad (ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
04 $\quad sk_i \coloneqq (nsk_i, ksk_i, ssk_i)$
05 $\quad pk_i \coloneqq (npk_i, kpk_i, spk_i)$
06 $b \xleftarrow{\$} \{0,1\}$
07 $b' \leftarrow \mathcal{A}^{\mathtt{Chall}}((sk_1, pk_1), \ldots, (sk_n, pk_n))$
08 **return** $[\![b = b']\!]$

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

09 $(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
10 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_s)$
11 $nk \coloneqq \mathsf{H}_1(nk', \texttt{"auth"})$
12 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk_r)$
13 $(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
14 $m \leftarrow (kct, kpk_r)$
15 $\sigma \xleftarrow{\$} \mathtt{Chl}_{\mathsf{RSig}}(s, r, \{spk_s, spk_r\}, m)$ // challenge query
16 $k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
17 $sct \coloneqq \mathsf{Sym.Enc}(k', \sigma)$
18 $c \coloneqq (npk_e, kct, sct)$
19 $k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk_r)$
20 **if** $b = 1$
21 $\quad (c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r, sk_r)$
22 **return** (c, k)

Figure 34. Adversary $\mathcal{B}$ against **MC-Ano** security of RSig, having access to oracle $\mathtt{Chl}_{\mathsf{RSig}}$, simulating Game $\mathsf{G}_1/\mathsf{G}_2$ for adversary $\mathcal{A}$ from the proof of Theorem 16.

$\mathsf{G}_0 - \mathsf{G}_5$

01 $i^\star \xleftarrow{\$} \{0,1\}$ // $\mathsf{G}_1 - \mathsf{G}_5$
02 $\mathcal{R}, \mathcal{C} \leftarrow \emptyset$
03 **for** $i \in \{0,1\}$
04 $\quad (nsk_i, npk_i) \xleftarrow{\$} \mathsf{NIKE.Gen}$
05 $\quad (ksk_i, kpk_i) \xleftarrow{\$} \mathsf{KEM.Gen}$
06 $\quad (ssk_i, spk_i) \xleftarrow{\$} \mathsf{RSig.Gen}$
07 $\quad sk_i \coloneqq (nsk_i, ksk_i, ssk_i)$
08 $\quad pk_i \coloneqq (npk_i, kpk_i, spk_i)$
09 $b \xleftarrow{\$} \{0,1\}$
10 $b' \leftarrow \mathcal{A}^{\mathtt{Rev},\mathtt{Chall}}(pk_0, pk_1)$
11 **if** $\mathcal{R} \cap \mathcal{C} \neq \emptyset$
12 $\quad$ **abort**
13 **return** $[\![b = b']\!]$

$\mathtt{Rev}(i \in \{0,1\})$

14 **if** $i = i^\star$ // $\mathsf{G}_1 - \mathsf{G}_5$
15 $\quad$ **abort** // $\mathsf{G}_1 - \mathsf{G}_5$
16 $\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$
17 **return** sk_i

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

18 $(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
19 $(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
20 $k' \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_1}$
21 $sct \coloneqq \mathsf{Sym.Enc}(k', 0)$
22 $c \coloneqq (npk_e, kct, sct)$
23 $k \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_2}$
24 **return** (c, k)

Oracle $\mathtt{Chall}(s \in \{0,1\}, r \in \{0,1\})$ // one query

25 **if** $s = r$ **return** $\bot$
26 **if** $r \neq i^\star$ // $\mathsf{G}_1 - \mathsf{G}_5$
27 $\quad$ **abort** // $\mathsf{G}_1 - \mathsf{G}_5$
28 $\mathcal{C} \leftarrow \mathcal{C} \cup \{r\}$
29 $(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
30 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_s, npk_r)$
31 $nk \coloneqq \mathsf{H}_1(nk', \texttt{"auth"})$
32 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk_r)$
33 $nk_1 \| nk_2 \xleftarrow{\$} \mathcal{K}_{\mathsf{NIKE}}$ // $\mathsf{G}_{2.1} - \mathsf{G}_5$
34 $(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
35 $kk_1 \| kk_2 \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$ // $\mathsf{G}_{2.2} - \mathsf{G}_5$
36 $m \leftarrow (kct, kpk_r)$
37 $\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk_r\}, m)$
38 $k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
39 $k' \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_1}$ // $\mathsf{G}_3 - \mathsf{G}_5$
40 $\sigma \coloneqq 0$ // G_5
41 $sct \coloneqq \mathsf{Sym.Enc}(k', \sigma)$
42 $c \coloneqq (npk_e, kct, sct)$
43 $k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk_r)$
44 $k \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_2}$ // $\mathsf{G}_4 - \mathsf{G}_5$
45 **if** $b = 1$
46 $\quad (c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r)$
47 **return** (c, k)

Figure 35. Games $\mathsf{G}_0 - \mathsf{G}_5$ for the proof of Theorem 17.

Game G_1 This game is the same as G_0 except that the experiment chooses a random user in the beginning of the game and aborts if the reveal oracle is queried for that user or the challenge oracle is queried for that user as a receiver.

Claim.

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| \leq 2 \cdot \left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|.$$

Proof. An adversary with a non-zero advantage has to query the challenge oracle because otherwise there is no strategy in outputting the correct bit that is better than guessing. For querying the challenge oracle and still fulfilling the winning condition ($\mathcal{R} \cap \mathcal{C} \neq \emptyset$), the receiver's key cannot be revealed. The probability that the challenged receiver is guessed correctly is $\frac{1}{2}$. ■

Remark. We define the following two hybrids ($\mathsf{G}_{2.1}$ and $\mathsf{G}_{2.2}$) in parallel which means that we fork the sequence and indicate the parallel hybrids via a sub index. After the fork we can apply the same proof to obtain a common hybrid again (G_3). This allows us to obtain a minimum when collecting the overall bound in the end without presenting two separate proofs.

Game $\mathsf{G}_{2.1}$ This game is the same as G_1 except that the second NIKE shared key, $nk_1 \| nk_2$, is replaced by a uniformly random value from the NIKE key space.

Claim. There exists an adversary $\mathcal{B}$ against **CKS** security of NIKE such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{2.1}^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(2,0,1)\text{-}\mathbf{CKS}}_{\mathsf{NIKE},\mathcal{B}}.$$

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 36. Note that the shared key nk' in the challenge oracle can be computed by the experiment itself since the sender key is known. Further, there is no need for reveal corrupt queries which allows for a weaker security requirement for the underlying NIKE, namely CKS security with honest key registration or passive secure NIKE. If $\mathcal{B}$ in their real game, they simulate G_1, in their random game they simulate $\mathsf{G}_{2.1}$. Hence, if $\mathcal{A}$ can distinguish the cases, $\mathcal{B}$ can win their game.
■

Game $\mathsf{G}_{2.2}$ This game is the same as G_1 except that the KEM key, $kk_1 \| kk_2$, is replaced by a uniformly random value from the KEM key space.

Claim. There exists an adversary $\mathcal{C}$ against **IND-CPA** security of KEM such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{2.2}^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(1,1)\text{-}\mathbf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{C}}.$$

Proof. The claim can be proved straightforwardly by querying the challenge oracle of the KEM for each call to the AKEM challenge oracle `Chall`. ■

Game G_3 This game is the same as $\mathsf{G}_{2.1}/\mathsf{G}_{2.2}$ except that the output of H_1 is replaced by a uniformly random value of the output range $\mathcal{K}_{\mathsf{H}_1}$.

Claim. There exists an adversary $\mathcal{D}$ against **PRF** security of H_1 such that for $i \in \{1, 2\}$

$$\left|\Pr\left[\mathsf{G}_{2.i}^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(1,1)\text{-}\mathbf{PRF}}_{\mathsf{H}_1,\mathcal{D}}.$$

Proof. The proof can be done straightforwardly by first proving the result for keying H_1 on the first input and then with the same strategy for the second input. Since we only allow for one challenge query, we need one PRF key and one evaluation query. ■

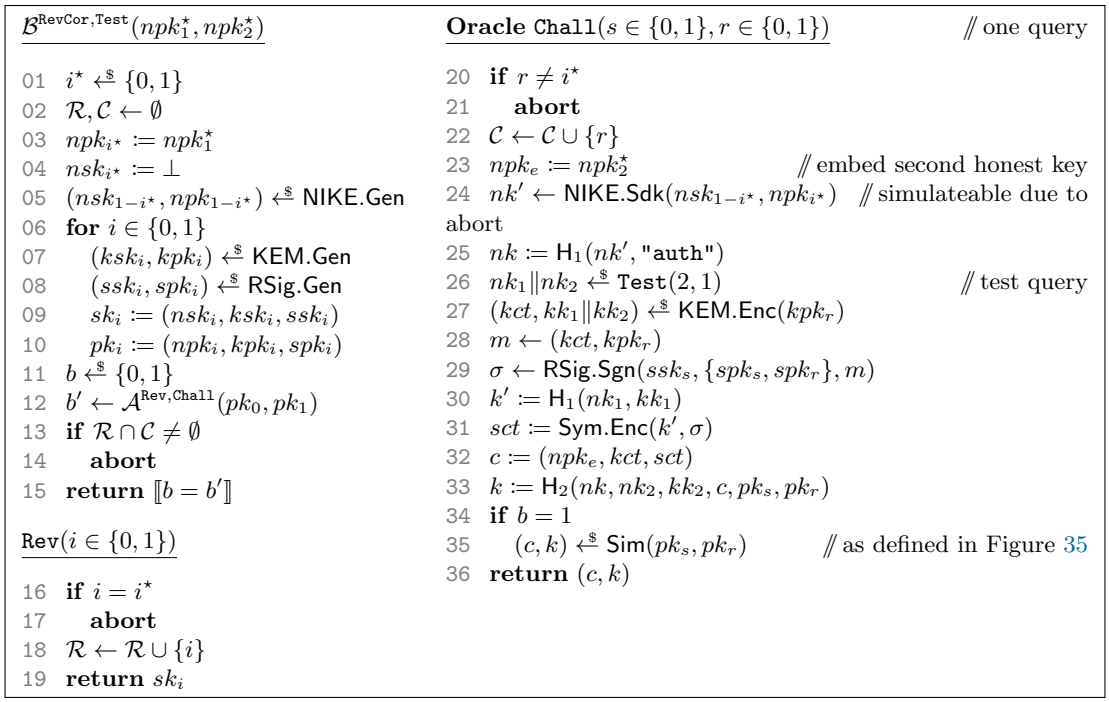

Figure 36. Adversary $\mathcal{B}$ against **CKS** security of NIKE, having access to oracles `RevCor` and `Test`, simulating Game $\mathsf{G}_1/\mathsf{G}_{2.1}$ for adversary $\mathcal{A}$ from the proof of Theorem 17.

Game G_4 This game is the same as G_3 (based on its possible two predecessors) except that the output of H_2 is replaced by a uniformly random value of the output range $\mathcal{K}$.

Claim. There exists an adversary $\mathcal{E}$ against **PRF** security of H_2 such that

$$\left|\Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{(1,1)\text{-}\mathbf{PRF}}_{\mathsf{H}_2,\mathcal{E}}.$$

Proof. The claim can be proved in the same way as for the previous game. ■

Game G_5 This game is the same as G_4 except that the signature σ is replaced by 0.

Claim. There exists an adversary $\mathcal{F}$ against **IND-CPA** security of Sym such that

$$\left|\Pr\left[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2 \cdot \mathrm{Adv}^{\mathbf{IND\text{-}CPA}}_{\mathsf{Sym},\mathcal{F}}.$$

Proof. Adversary $\mathcal{F}$ can simulate the whole game by generating the secret keys themselves. Due to the changes in G_3 the symmetric key is uniformly chosen and independent of the rest of the game. Hence, the reduction can query their own **IND-CPA** challenge oracle on the original σ and 0. In case $b = 0$, $\mathcal{F}$ simulates G_4; otherwise they simulate G_5. Since there is only one challenge query, the claim follows. ■

Since the output distribution of the challenge oracle in case $b = 0$ is the same as for the simulator the resulting game is independent of the challenge bit and thus it holds

$$\Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

To obtain a security for the multi-user multi-challenge setting we can apply a hybrid argument which yields the following upper bound and thus the theorem statement.

$$\mathrm{Adv}^{(n,\mathrm{Q}_{\mathtt{Chl}})\text{-}\mathbf{HR\text{-}Den}}_{\mathsf{AKEM},\mathsf{Sim},\mathcal{A}} \leq n^2 \cdot Q_{\mathtt{Chl}} \cdot \mathrm{Adv}^{(2,1)\text{-}\mathbf{HR\text{-}Den}}_{\mathsf{AKEM},\mathsf{Sim},\mathcal{A}'} .$$

■