Shadowfax: Combiners for Deniability

Phillip Gajland
Max Planck Institute
for Security and Privacy
Ruhr University Bochum
Bochum, Germany

Vincent Hwang
Max Planck Institute
for Security and Privacy
Bochum, Germany
Radboud University
Nijmegen, The Netherlands

Jonas Janneck
Ruhr University Bochum
Bochum, Germany

ABSTRACT

As cryptographic protocols transition to post-quantum security, most adopt hybrid solutions combining pre-quantum and post-quantum assumptions. However, this shift often introduces trade-offs in terms of efficiency, compactness, and in some cases, even security. One such example is *deniability*, which enables users, such as journalists or activists, to deny authorship of potentially incriminating messages. While deniability was once mainly of theoretical interest, protocols like X3DH, used in Signal and WhatsApp, provide it to billions of users. Recent work (Collins et al., PETS'25) has further bridged the gap between theory and real-world applicability. In the post-quantum setting, however, protocols like PQXDH, as well as others such as Apple's iMessage with PQ3, do not support deniability. This work investigates how to preserve deniability in the post-quantum setting by leveraging unconditional (statistical) guarantees instead of computational assumptions - distinguishing deniability from confidentiality and authenticity.

As a case study, we present a hybrid authenticated key encapsulation mechanism (AKEM) that provides statistical deniability, while maintaining authenticity and confidentiality through a combination of pre-quantum and post-quantum assumptions. To this end, we introduce two combiners at different levels of abstraction. First, at the highest level, we propose a black-box construction that combines two AKEMs, showing that deniability is preserved only when both constituent schemes are deniable. Second, we present Shadowfax, a non-black-box combiner that integrates a pre-quantum NIKE, a post-quantum KEM, and a post-quantum ring signature. We demonstrate that Shadowfax ensures deniability in both dishonest and honest receiver settings. When instantiated, we rely on statistical security for the former, and on a pre- or post-quantum assumption in the latter. Finally, we provide an optimised, yet portable, implementation of a specific instantiation of Shadowfax yielding ciphertexts of 1 781 bytes and public keys of 1 449 bytes. Our implementation achieves competitive performance: encapsulation takes 1.9 million cycles and decapsulation takes 800 000 cycles on an Apple M1 Pro.

CCS CONCEPTS

• **Security and privacy** → **Cryptography**.

KEYWORDS

Deniability, Authenticated KEM, Combiner

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 the process nearing completion and four algorithms selected (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. [1]

Significant progress has been made towards adapting widely deployed protocols to be post-quantum secure. Notable examples include X3DH [MP16], which supports billions users on WhatsApp and, more recently Messenger, and has been extended to a post-quantum variant, PQXDH, deployed in Signal [KS24], as well as Apple's iMessage with PQ3 [App24]. Another prominent example is TLS, which has been updated for post-quantum security by using KEMs in multiple papers [BCNS15, PST20, BBCT22] and real-world deployments [Lan16, Lan18, KV19, WR19].

A key aspect of all these adaptations is the *hybrid* approach, which combines post-quantum algorithms with classical cryptographic methods. This is essential, as post-quantum solutions, despite their potential, lack the decades of cryptographic analysis that traditional schemes such as 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]. The German Federal Office for Information Security (BSI) is more explicit, stating 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

Traditionally, a hybrid scheme, or combiner, ensures security as long as at least one of the combined methods remains secure. For example, if cryptographically relevant quantum computers (CRQCs) become available rendering problems like factoring and discrete logarithms tractable [Sho94], the hybrid scheme would still be secure as the post-quantum assumption remains intact. Conversely, if advances in cryptanalysis or implementation issues

[1]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.

break the post-quantum scheme (classically) in polynomial time, the classical security of the hybrid scheme would still hold due to the hardness of the pre-quantum problem. In fact, recent work demonstrated several classical attacks on post-quantum schemes [Beu22, CD23, MMP+23, Rob23] underscoring the importance of hybrids. Furthermore, only the basic post-quantum primitives, such as KEMs and signatures, have been standardised, necessitating the use of non-standard primitives and motivating their adoption in hybrid configurations. Finally, to achieve the goal of *"cryptographic agility"* [OPowp19] in the long run, the permanent use of hybrid solutions may become common practice.

1.1.1 Combiners for Confidentiality and Authenticity. Combiners have been used to achieve both *confidentiality*, for example by combining pre- and post-quantum Key Encapsulation Mechanisms (KEMs), and *authenticity*, such as by combining pre- and post-quantum signature schemes.

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 ciphertext indistinguishability under adaptive chosen ciphertext attacks, 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].

Authenticity. Hybrid solutions for *authenticity*, such as combining digital signature schemes have also been explored [BHMS17, OGP+24]. 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. The works of [BHMS17, OGP+24] examined hybrid digital signatures within 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.

1.2 Deniability

While hybrid solutions for confidentiality and authenticity have been studied in the literature, and require an adversary to break *both* layers to compromise the scheme, the notion of *deniability* does not appear to exhibit this property, and remains largely unexplored. Informally, deniability allows a sender to plausibly deny involvement in a authenticated transaction. It ensures that the sender's actions could have been done by anyone, making it impossible to definitively prove the sender's participation to a third party. In fact, the generic *natural combiner* fails to preserve deniability: if one component loses its deniability, making part of the transcript undeniable, then how can the entire transcript, which also includes this part, still remain deniable? This raises the following question:

"Can combiners preserve deniability?"

To answer this, we must first understand the purpose and motivation behind deniability itself.

1.2.1 The Case for Deniability. While hybrid solutions for confidentiality and authenticity have been studied in the literature, and require an adversary to break *both* layers to compromise the scheme, the notion of *deniability* does not appear to exhibit this property, and remains largely unexplored. Cryptographic deniability, once deemed only of theoretical interest, rose to prominence during the 2016 United States presidential election. In the final weeks leading up to the election, approximately 58,000 emails from Hillary Clinton's campaign were leaked [Wik16]. The campaign vehemently denied the authenticity of the emails, claiming they had been fabricated as part of a smear campaign [Mas16, Car16, BBC16]. Typically, emails are unauthenticated, which provides plausible deniability, allowing senders to deny authorship. However, in this case, the situation was complicated by the fact that the emails were cryptographically signed - not by the authors, but by mail transfer agents, such as Google's servers, using DomainKeys Identified Mail (DKIM), a widely adopted anti-spam measure [LF07]. As a result, the emails were verifiably unaltered, undermining the campaign's claims of forgery. Political emails are just one example where the ability to deny authorship is valuable. This feature has been proposed as a means for dissidents, journalists and activists to protect themselves from persecution by disavowing association with controversial or subversive messages.

Deniability in Practice. Off-the-Record (OTR) [BGB04] was the first protocol allowing encryption and authentication of messages while removing the non-repudiation property of signature-based protocols like GPG and S/MIME, enabling deniability. Since then, deniability in protocols has gained significant attention in both academia and industry. Successors to OTR are now used in over two billion devices globally, through services like Signal [MP16], WhatsApp [Wha20] and Messenger [Met23]. 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.

Cryptographic Deniability. 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 [GJK24], 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. Therefore, we revise the aforementioned question to be:

"Can combiners preserve deniability in a post-quantum setting at minimal additional cost?"

1.3 Deniability for PQC

We argue that deniability is fundamentally different to both authenticity and confidentiality. To understand this distinction, it is useful to first consider the broader context of cryptographic security. To this end, formalising the security of cryptosystems often involves distinguishing between two distributions such as encryptions of two different messages. Shannon's seminal work [Sha49] established that perfect secrecy (confidentiality), achievable by the one-time-pad, requires a key length equal to the message length [Sha49, Sec. 10]. Practical cryptosystems, therefore, necessarily rely on weaker notions of secrecy.

1.3.1 Statistical and Computational Security. A natural relaxation of perfect secrecy is *statistical* security, where an adversary's advantage in identifying the encrypted message is marginally better than random guessing. A scheme is said to be ϵ-statistically indistinguishable if the statistical distance between the two ciphertext distributions is at most ϵ. In other words, an unbounded adversary's probability of correctly distinguishing between two encrypted messages is at most $1/2 + \epsilon$. For small values of ϵ, such as 2^{-80}, this remains a meaningful security notion. However, even with statistical security, we cannot circumvent the impossibility that keys may not be shorter than messages.

As a consequence, deployed cryptography primarily relies on the *computational* infeasibility of certain mathematical problems. Specifically, a classical adversary running in probabilistic polynomial time relative to the input length n cannot distinguish between two distributions with probability greater than $1/2 + \mathsf{negl}(n)$, where $\mathsf{negl}(\cdot)$ denotes some negligible function. Security proofs typically show that achieving a better distinguishing advantage would require breaking the underlying hard problem. An unbounded adversary could, of course, solve these problems by brute force. Similarly, a quantum adversary running in polynomial time could distinguish two distributions if their closeness relies on the hardness of a pre-quantum problem such as factoring integers or solving discrete logarithms over finite fields [Sho94]. Therefore, these problems are only considered computationally hard for classical adversaries. Crucially, if the distributions are statistically close, even a quantum adversary - regardless of whether it is polynomial time or unbounded - cannot distinguish between them. In other words, both perfect and statistical security are unconditional.

1.3.2 The Difference between Confidentiality, Authenticity and Deniability. A key distinction between authenticity, confidentiality, and deniability lies in their reliance on computational hardness assumptions. *Authenticity*, when based on asymmetric primitives like signature schemes, necessarily relies on assumptions such as the discrete logarithm problem (DLOG) [DH76], or the Short Integer Solution (SIS) problem [Ajt96]. The existence of digital signatures, in fact, is equivalent to the existence of one-way functions [Rom90]. Similarly, *confidentiality* in asymmetric primitives, such as KEMs or public key encryption (PKE), requires computational hardness assumptions (likely more than only OWFs [Dac16]), like integer factorisation [RSA78] or Learning With Errors (LWE) [Reg05]. In contrast, deniability does not necessarily depend on hardness assumptions like LWE, though it may in some cases. Unlike authenticity and confidentiality, deniability can often be proven *unconditionally*, without requiring any computational assumptions. The central insight here is that when deniability is perfect or statistical, it is immune to the failure scenarios typically motivating the use of combiners. Since no assumption underpins the deniability, the property holds even against unbounded adversaries. This also resolves the issue that, unlike confidentiality and authenticity, deniability does require both components of a combined scheme to be deniable. Recall that for confidentiality and authenticity, an adversary must break *both* layers corresponding to pre- and post-quantum assumptions. However, in a natural combiner, it is sufficient for the adversary to break only *one* component's deniability to compromise the entire system's deniability. When deniability is unconditional, failure scenarios, such as breaking a computational assumption or encountering a new attack that invalidates the hardness of a post-quantum assumption, become irrelevant. Of course, if one of the schemes fails to provide deniability due to a design flaw rather than a flawed assumption, the combiner will offer no security.

2 AKEM: A CASE STUDY

To illustrate these observations, we focus on a specific primitive: the Authenticated Key Encapsulation Mechanism (AKEM) [ABH+21]. The recent HPKE standard [BBLW22] defines four distinct modes, two of which – Auth and AuthPSK – are formalised via AKEMs [ABH+21]. The AuthPSK mode is currently deployed in the MLS [BBR+23] standard. AKEMs, inspired by the singcryption literature [DZ10], can be viewed as a generalisation of the split-KEM primitive [BFG+20]. [2] Although not yet widely deployed, AKEMs exhibit several desirable properties that make them suitable for many practical applications. In this work, we extend the ideas presented in [GJK24] and use AKEMs as a case study to explore the design of combiners that preserve deniability. While the principles outlined apply to other primitives, AKEMs serve as a concrete example for a detailed examination of these concepts. Informally, an AKEM has the same interfaces as a standard KEM, but 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.

2.1 Deniable AKEM Combiners

Deniability captures scenarios where a sender can plausibly deny having sent a potentially incriminating message to a receiver, while still ensuring the receiver can authenticate the message's origin. The aim is to prevent a third party, the judge (modelled as an adversary), from conclusively attributing the sender's involvement. Formally, we assume the existence of a simulator Sim that can generate a ciphertext c and key k indistinguishable from those produced by the encapsulation process Enc to any polynomial time adversary $\mathcal{A}$. The existence of such a simulator allows the sender to plausibly deny sending specific messages encrypted under k (where k is used in a KEM-DEM scheme for encrypting messages [BBLW22]), as anyone could have generated the same ciphertexts using Sim.

Dishonest vs Honest Receivers. The model of deniability varies depending on the scenario [GJK24]. For a *dishonest receiver*, Sim is given the receiver's secret key, representing a situation where the receiver could forge a ciphertext to make it appear as if c originated from the sender. For *honest receivers* the receiver is assumed to not simulate any values and therefore Sim is not given the receiver's secret key. This distinction is critical: deniability in the dishonest receiver setting does *not imply* deniability in the honest receiver setting, whereas security with honest receivers does imply security with dishonest receivers, as the simulator's capabilities increase while the adversary's remain unchanged. Further distinctions in deniability can be made for both honest and dishonest receivers, based on the keys the adversary/judge is given. For a detailed analysis of AKEM deniability, see [GJK24, Sec. 4.2].

[2] In fact, a symmetric split-KEM [BFG+20, Def. 4] is almost the same as an AKEM [ABH+21, Def. 9].

In the post quantum setting we focus exclusively on the strongest (and most meaningful) scenario, where $\mathcal{A}$ is assumed to be a polynomial time quantum adversary, while Sim remains a classical PPT machine.

Combiners. As noted, capturing deniability for combiners is more complex than for confidentiality and authenticity. In the latter cases, security is maintained as long as one component is secure, requiring the adversary to break both. One might expect a similar property for deniability, where the combiner preserves deniability as long as one component is deniable. However, achieving this in a black-box manner appears challenging. Consider, for instance, the natural approach from [GHP18] where $k := \mathsf{H}(k_1, k_2, c_1, c_2)$ and $c := (c_1, c_2)$. The primary challenge in designing a deniable combiner lies in constructing a simulator for the final scheme, which seems to require simulators for both underlying schemes. In fact, we conjecture that it is impossible to achieve a deniable AKEM by combining two AKEMs in a *black-box* manner if only one of the schemes provides deniability. Thus, we require *both* schemes to be deniable. Nevertheless, we argue that this is not an issue by relying on statistical deniability, which cannot be "lost" if assumptions are later broken unlike computational deniability. Recall that the motivation of a combiner is two fold: First, if quantum computers become capable of solving problems like factoring or discrete logarithms efficiently, the hybrid scheme retains security due to the post-quantum assumption. Second, if advances in classical cryptanalysis or implementation vulnerabilities compromise the post-quantum scheme, the security of the hybrid scheme is maintained by the hardness of the classical problem. By focusing on AKEM constructions where deniability is a statistical property rather than a computational one, we can ensure that deniability for the combiner is preserved come what may.

Dishonest Receivers. In the case of *dishonest receiver* deniability, we can construct a combiner where both AKEMs preserve their deniability by relying on statistical guarantees, ensuring neither breaks. For an AKEM where the authenticity (and confidentiality) depend on a pre-quantum assumption, we can instantiate it using a NIKE, where the dishonest deniability would rely on the correctness of the NIKE (g^{ab} is perfectly indistinguishable from g^{ba}, because they are the same). For the second AKEM where the authenticity (and confidentiality) rely on a post-quantum assumption we could construct it using ring signatures provided the ring signature anonymity is statistical. Indeed Gandalf [GJK24] does satisfy statistical anonymity. This approach is ineffective for schemes where deniability relies on computational assumptions, as the entire combiner's deniability could fail if those assumptions are broken. For instance, this issue arises with ring signatures such as SMILE [LNS21] or Erebor [BLL24] whose anonymity depends on hardness assumptions.

Honest Receivers. If we want to take the same approach in the *honest receiver* setting, we have the following practical problem. To the best of our knowledge there are no efficient post-quantum

AKEMs that unconditionally satisfy honest receiver deniability. [3] For instance, the honest receiver deniability of the post-quantum AKEM from [GJK24] relies on the confidentiality of the underlying KEM and thus on a computational assumption. If we relax the requirement for unconditional security in favour of computational assumptions, we must address the conjectured impossibility by considering a non-black-box construction. To that end, we propose a concrete construction that satisfies honest receiver deniability by relying on the security of just one pre-quantum or post-quantum assumption. This is achieved by basing the confidentiality requirement of the PQ-AKEM not only on its underlying KEM but also on the pre-quantum NIKE, forming what we term a "sub-combiner". Importantly, this approach requires a non-black-box construction, as it involves breaking open the PQ-AKEM rather than assuming black-box access.

2.2 Contributions

We introduce a framework for reasoning about deniability in the context of post-quantum combiners, an area previously unexplored. As detailed above, our key insight is that deniability differs from other security notions, such as confidentiality and authenticity. We demonstrate that primitives with unconditional deniability can be leveraged to achieve the desired combiner properties.

While much of our approach is generalisable to other primitives, we focus on the authenticated key encapsulation mechanism (AKEM) as a concrete example to explore and apply these insights in detail. We present two combiners for AKEMs at different levels of abstraction, each with distinct trade-offs:

- At the highest level of abstraction, we propose a generic, black-box construction that combines two AKEMs. We prove that deniability is preserved only if both underlying schemes are deniable. Moreover, our construction requires only one of the AKEMs to provide confidentiality, and similarly, only one to provide authenticity, consistent with the expected combiner characteristics.
- At a lower level of abstraction, we introduce Shadowfax, a non-black-box combiner that builds on pre-quantum 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 is guaranteed under one computational assumption: either the security of the ephemeral NIKE or the KEM.

Our final contribution is a set of portable C implementations designed for compactness, reproducibility, and easy integration into existing cryptographic libraries. We provide C reference implementations of the Gandalf ring signature scheme and the post-quantum AKEM from [GJK24]. Additionally, we implement our hybrid AKEM, Shadowfax. When instantiated with X25519 [LHT16] as the NIKE, BAT [FKPY22] as the post-quantum KEM and Gandalf [GJK24] as the post-quantum ring signature scheme, Shadowfax features compact ciphertexts (1781 bytes) and public keys (1449 bytes). Our platform-agnostic C implementations leverage recent advancements in lattice-based cryptography, offering competitive performance. On a Firestorm core running at 3 GHz on an Apple M1 Pro, encapsulation takes approximately 1.9 million cycles, while decapsulation takes 800,000 cycles. For detailed parameter sizes and performance metrics, refer to Table 2, Table 3 and the project's GitHub repository at Shadowfax.

[3] The NIKE-AKEM from [AJKL23] would satisfy such a notion but has prohibitively large public keys.

3 PRELIMINARIES

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

3.1 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 $[n] := \{1, \ldots, n\}$. 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 $(x_1, \ldots)$. We write $\mathcal{A}^{\mathcal{B}}$ to denote that $\mathcal{A}$ has oracle access to $\mathcal{B}$ during its execution. For a randomised algorithm $\mathcal{A}$, we use the notation $y \in \mathcal{A}(x)$ to denote that y is a possible output of $\mathcal{A}$ on input x. The support of a discrete random variable X is defined as $\sup(X) := \{x \in \mathbb{R} \mid \Pr[X = x] > 0\}$.

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

3.2 AKEM

Definition 1 (Authenticated Key Encapsulation Mechanism)**.** 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 $(n, Q_{\mathsf{Enc}}, Q_{\mathsf{Dec}}, Q_{\mathtt{Chl}})$-$\mathbf{Ins\text{-}CCA}_{\mathsf{AKEM}}(\mathcal{A})$ and $(n, Q_{\mathsf{Enc}}, Q_{\mathsf{Dec}})$-$\mathbf{Out\text{-}CCA}_{\mathsf{AKEM}}(\mathcal{A})$, depicted in Figure 1 and Figure 2, respectively. The advantage of adversary $\mathcal{A}$ is defined as

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

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

```
Game (n, Q_Enc, Q_Dec, Q_Chl)-Ins-CCA_AKEM(A)
D := ∅
for i ∈ [n]
    (sk_i, pk_i) <-$ Gen
b <-$ {0, 1}
b' <- A^{Encps,Decps,Chall}(pk_1, ..., pk_n)
return [[b = b']]

Oracle Encps(s ∈ [n], pk)
(c, k) <-$ Enc(sk_s, pk)
return (c, k)

Oracle Chall(sk, r ∈ [n])
(c, k) <-$ Enc(sk, pk_r)
if b = 1
    k <-$ K
    D <- D ∪ {(μ(sk), pk_r, c, k)}
return (c, k)

Oracle Decps(pk, r ∈ [n], c)
if ∃ k : (pk, pk_r, c, k) ∈ D
    return k
k <- Dec(pk, sk_r, c)
return k
```

Figure 1: 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_{Enc} **queries to Encps,** Q_{Dec} **queries to Decps,** Q_{CSK} **queries to CorSK, and** $Q_{\mathtt{Chl}}$ **queries to** **`Chall`.**

```
Game (n, Q_Enc, Q_Dec)-Out-CCA_AKEM(A)
D := ∅
for i ∈ [n]
    (sk_i, pk_i) <-$ Gen
b <-$ {0, 1}
b' <- A^{Encps,Decps}(pk_1, ..., pk_n)
return [[b = b']]

Oracle Encps(s ∈ [n], pk)
(c, k) <-$ Enc(sk_s, pk)
if b = 1 ∧ pk ∈ {pk_1, ..., pk_n}
    k <-$ K
    D <- D ∪ {(pk_s, pk, c, k)}
return (c, k)

Oracle Decps(pk, r ∈ [n], c)
if ∃ k : (pk, pk_r, c, k) ∈ D
    return k
k <- Dec(pk, sk_r, c)
return k
```

Figure 2: 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_{Enc} **queries to Encps and** Q_{Dec} **queries to Decps.**

Authenticity. We consider outsider authenticity from [ABH+21], the strongest notion that is achievable when also seeking deniability [GJK24]. We formalise the notion via the game $(n, Q_{\mathsf{Enc}}, Q_{\mathtt{Chl}})$-$\mathbf{Out\text{-}Aut}_{\mathsf{AKEM}}(\mathcal{A})$ depicted in Figure 3 and define the advantage of an adversary $\mathcal{A}$ as

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

```
Games (n, Q_Enc, Q_Chl)-Out-Aut_AKEM(A)
D := ∅
for i ∈ [n]
    (sk_i, pk_i) <-$ Gen
b <-$ {0, 1}
b' <-$ A^{Encps,Chall}(pk_1, ..., pk_n)
return [[b = b']]

Oracle Encps(s ∈ [n], pk)
(c, k) <-$ Enc(sk_s, pk)
D <- D ∪ {(pk_s, pk, c, k)}
return (c, k)

Oracle Chall(pk, r ∈ [n], c)
if ∃ k : (pk, pk_r, c, k) ∈ D
    return k
k <- Dec(pk, sk_r, c)
if b = 1 ∧ pk ∈ {pk_1, ..., pk_n} ∧ k ≠ ⊥
    k <-$ K
    D <- D ∪ {(pk, pk_r, c, k)}
return k
```

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

Deniability. As in [GJK24], 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 PPT simulator Sim, we define deniability in the *dishonest receiver* setting via game $(n, Q_{\mathtt{Chl}})$-$\mathbf{DR\text{-}Den}$ and in the *honest receiver* setting via game (n, Q_{Chl})-**HR-Den** as depicted in Figure 4. The advantage of an adversary $\mathcal{A}$ is then defined as

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

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

```
Games (n, Q_Chl)-DR-Den_AKEM,Sim(A)
      (n, Q_Chl)-HR-Den_AKEM,Sim(A)
R, C ← ∅
for i ∈ [n]
    (sk_i, pk_i) ←$ Gen
b ←$ {0, 1}
b' ← A^{Rev,Chall}(pk_1, ..., pk_n)
if R ∩ C ≠ ∅                                   / HR-Den
    abort                                      / HR-Den
return [[b = b']]

Oracle Chall(s ∈ [n], r ∈ [n])
if s = r return ⊥
C ← C ∪ {r}
(c, k) ←$ Enc(sk_s, pk_r)
if b = 0
    continue
if b = 1
    (c, k) ←$ Sim(pk_s, pk_r, sk_r)    / DR-Den
    (c, k) ←$ Sim(pk_s, pk_r)          / HR-Den
return (c, k)

Oracle Rev(i ∈ [n])
R ← R ∪ {i}
return sk_i
```

Figure 4: 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_{Chl} **queries to** `Chall`**.**

4 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, as illustrated Figure 5. This construction builds upon the natural approach proposed in [GHP18]. Regarding security, our results are as follows: For confidentiality (see Theorem 2) and authenticity (see Theorem 3) 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 4 shows that if both schemes are *dishonest receiver* deniable, then the combiner inherits this property. Similarly, Theorem 11 establishes that the combiner maintains deniability in the *honest receiver* setting if both underlying schemes are honest receiver deniable.

Lemma 1 (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 2 (Confidentiality)**.** *For any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, depicted in Figure 5, there exists an* **Ins-CCA** *adversary* $\mathcal{B}_1$ *against* AKEM_1*, an* **Ins-CCA** *adversary* $\mathcal{B}_2$ *against* AKEM_2*, and a* **mPRF** *adversary* C *against* H *such that*

$$\mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathsf{Chl}})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathcal{A}} \leq \min\left\{\mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathsf{Chl}})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}_1,\mathcal{B}_1}, \mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathsf{Chl}})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}_2,\mathcal{B}_2}\right\} + \mathrm{Adv}^{(Q_{\mathsf{Chl}},Q_{\mathsf{Dec}}+Q_{\mathsf{Chl}})\text{-}\mathbf{mPRF}}_{\mathsf{H},C} + Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}]}.$$

The proof can be found in Appendix B.

Theorem 3 (Authenticity)**.** *For any* **Out-Aut** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, as depicted in Figure 5, 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* C_1 *against* AKEM_1*, an* **Out-CCA** *adversary* C_2 *against* AKEM_2*, and a* **mPRF** *adversary* $\mathcal{D}$ *against* H *such that*

$$\mathrm{Adv}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathcal{A}} \leq \min\left\{\mathrm{Adv}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}_1,\mathcal{B}_1} + \mathrm{Adv}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}CCA}}_{\mathsf{AKEM}_1,C_1}, \mathrm{Adv}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}}_{\mathsf{AKEM}_2,\mathcal{B}_2} + \mathrm{Adv}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}CCA}}_{\mathsf{AKEM}_2,C_2}\right\} + \mathrm{Adv}^{(Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}},Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}})\text{-}\mathbf{mPRF}}_{\mathsf{H},\mathcal{D}} + Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}]}.$$

The proof can be found in Appendix B.

Theorem 4 (Dishonest Deniability)**.** *For all* PPT *simulators* $\mathsf{Sim}_1, \mathsf{Sim}_2$ *there exists a* PPT *simulator* $\mathsf{Sim}[\mathsf{Sim}_1, \mathsf{Sim}_2]$ *such that for any* **DR-Den** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, as depicted in Figure 5, there exists a* **DR-Den** *adversary* $\mathcal{B}_1$ *against* AKEM_1 *and a* **DR-Den** *adversary* $\mathcal{B}_2$ *against* AKEM_2 *such that*

$$\mathrm{Adv}^{(n,Q_{\mathsf{Chl}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathsf{Sim},\mathcal{A}} \leq \mathrm{Adv}^{(n,Q_{\mathsf{Chl}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}_1,\mathsf{Sim}_1,\mathcal{B}_1} + \mathrm{Adv}^{(n,Q_{\mathsf{Chl}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM}_2,\mathsf{Sim}_2,\mathcal{B}_2}.$$

The proof can be found in Appendix B.

5 CONCRETE CONSTRUCTION: SHADOWFAX

In this section, we present a concrete construction for a deniable AKEM combiner based on a non-interactive key exchange NIKE, a key encapsulation mechanism KEM, a ring signature scheme RSig, a symmetric encryption scheme SE, and two (multi-)keyed functions H_1 and H_2, as shown in Figure 6. 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 6) and authenticity (see Theorem 7), 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 8) 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 9) 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

```
Gen
(sk_1, pk_1) <-$ AKEM_1.Gen
(sk_2, pk_2) <-$ AKEM_2.Gen
sk := (sk_1, sk_2)
pk := (pk_1, pk_2)
return (sk, pk)

Enc(sk_s, pk_r)
parse sk_s -> (sk_1, sk_2)
parse pk_r -> (pk_1, pk_2)
(c_1, k_1) <-$ AKEM_1.Enc(sk_1, pk_1)
(c_2, k_2) <-$ AKEM_2.Enc(sk_2, pk_2)
c := (c_1, c_2)
k := H(k_1, k_2, (μ(sk_1), μ(sk_2)), (pk_1, pk_2), c)
return (c, k)

Dec(pk_s, sk_r, c)
parse pk_s -> (pk_1, pk_2)
parse sk_r -> (sk_1, sk_2)
parse c -> (c_1, c_2)
k_1 <- AKEM_1.Dec(pk_1, sk_1, c_1)
k_2 <- AKEM_2.Dec(pk_2, sk_2, c_2)
k := H(k_1, k_2, (pk_1, pk_2), (μ(sk_1), μ(sk_2)), c)
return k
```

Figure 5: Generic Construction of a deniable authenticated key encapsulation mechanism $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

from the public verifiability of the ring signature. [GJK24] 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 5 (Correctness). *If* NIKE *has correctness error* δ_{NIKE}*,* KEM *correctness error* δ_{KEM}*, and* RSig *correctness error* δ_{RSig} *and* SE *is (perfectly) correct, then*

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

Theorem 6 (Confidentiality). *For any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{SE}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 6, there exists an* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, a* **PRF** *adversary* C *against* H_1*, an* **mPRF** *adversary* $\mathcal{D}$ *against* H_2*, and an* **IND-CCA** *adversary* $\mathcal{E}$ *against* KEM *such that*

$$\begin{aligned}
\mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathsf{Chl}})\textbf{-Ins-CCA}}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2],\mathcal{A}} &\leq nQ_{\mathsf{Chl}} \\
\cdot \Big(\min \Big\{&\mathrm{Adv}^{(Q_{\mathsf{Enc}}+2,2Q_{\mathsf{Enc}}+2Q_{\mathsf{Dec}},2Q_{\mathsf{Enc}}+2Q_{\mathsf{Enc}}+1)\textbf{-CKS}}_{\mathsf{NIKE},\mathcal{B}} \\
&+ \mathrm{Adv}^{(1,1)\textbf{-PRF}}_{\mathsf{H}_1,C}, \mathrm{Adv}^{(1,Q_{\mathsf{Dec}},1)\textbf{-IND-CCA}}_{\mathsf{KEM},\mathcal{E}}\Big\} \\
&+ \mathrm{Adv}^{(1,Q_{\mathsf{Dec}}+1)\textbf{-mPRF}}_{\mathsf{H}_2,\mathcal{D}} + (Q_{\mathsf{Enc}} + Q_{\mathsf{Dec}}) \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}} \\
&+ Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2]}\Big).
\end{aligned}$$

The proof can be found in Appendix C.

Theorem 7 (Authenticity). *For any* **Out-Aut** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{SE}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 6, there exists an* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, a* **PRF** *adversary* C *against* H_1*, an* **mPRF** *adversary* $\mathcal{D}$ *against* H_2*, a* **UF-CRA1** *adversary* $\mathcal{E}$ *against* RSig*, and an* **IND-CCA** *adversary* $\mathcal{F}$ *against* KEM*, such that*

$$\begin{aligned}
&\mathrm{Adv}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\textbf{-Out-Aut}}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2],\mathcal{A}} \\
&\quad\leq \min\Big\{\mathrm{Adv}^{(Q_{\mathsf{Enc}}+2Q_{\mathsf{Chl}},Q_{\mathsf{Enc}}+2Q_{\mathsf{Chl}})\textbf{-CKS}}_{\mathsf{NIKE},\mathcal{B}} + \mathrm{Adv}^{(n^2,n^2)\textbf{-PRF}}_{\mathsf{H}_1,C}, \\
&\qquad \mathrm{Adv}^{(n,2,Q_{\mathsf{Enc}})\textbf{-UF-CRA1}}_{\mathsf{RSig},\mathcal{E}} + \mathrm{Adv}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\textbf{-IND-CCA}}_{\mathsf{KEM},\mathcal{F}} \\
&\qquad + Q^2_{\mathsf{Enc}} \cdot \gamma_{\mathsf{KEM}}\Big\} \\
&\quad + \mathrm{Adv}^{(Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}},Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}})\textbf{-mPRF}}_{\mathsf{H}_2,\mathcal{D}} \\
&\quad + Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2]} \\
&\quad + Q_{\mathsf{Enc}} \cdot (Q_{\mathsf{Enc}} + Q_{\mathsf{Chl}}) \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}}.
\end{aligned}$$

The proof can be found in Appendix C.

Theorem 8 (Dishonest Deniability). *There exists a simulator* Sim *such that for any* **DR-Den** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{SE}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 6, there exists a* **MC-Ano** *adversary* $\mathcal{B}$ *against* RSig*, such that*

$$\begin{aligned}
&\mathrm{Adv}^{(n,Q_{\mathsf{Chl}})\textbf{-DR-Den}}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2],\mathsf{Sim},\mathcal{A}} \\
&\qquad \leq \mathrm{Adv}^{(n,2,Q_{\mathsf{Chl}})\textbf{-MC-Ano}}_{\mathsf{RSig},\mathcal{B}} + Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{NIKE}}.
\end{aligned}$$

The proof can be found in Appendix C.

Theorem 9 (Honest Deniability). *There exists a simulator* Sim *such that for any* **HR-Den** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{SE}, \mathsf{H}_1, \mathsf{H}_2]$*, as depicted in Figure 6, there exists a* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, an* **IND-CPA** *adversary* C *against* KEM*,* **mPRF** *adversaries* $\mathcal{D}$ *and* $\mathcal{E}$ *against* H_1 *and* H_2*, and a* **IND-CPA** *adversary* $\mathcal{F}$ *against* SE *such that*

$$\begin{aligned}
&\mathrm{Adv}^{(n,Q_{\mathsf{Chl}})\textbf{-HR-Den}}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2],\mathsf{Sim},\mathcal{A}} \\
&\quad \leq 2n^2 \cdot Q_{\mathsf{Chl}} \cdot \Big(\min\Big\{\mathrm{Adv}^{(2,0,1)\textbf{-CKS}}_{\mathsf{NIKE},\mathcal{B}}, \mathrm{Adv}^{(1,1)\textbf{-IND-CPA}}_{\mathsf{KEM},C}\Big\} \\
&\qquad + \mathrm{Adv}^{(1,1)\textbf{-mPRF}}_{\mathsf{H}_1,\mathcal{D}} + \mathrm{Adv}^{(1,1)\textbf{-mPRF}}_{\mathsf{H}_2,\mathcal{E}} + \mathrm{Adv}^{\textbf{IND-CPA}}_{\mathsf{SE},\mathcal{F}}\Big).
\end{aligned}$$

The proof can be found in Section 5.1.

5.1 Proof of Theorem 9

Proof. Consider the sequence of games depicted in Figure 7 as well as the construction of a simulator Sim.

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

```
Gen
(nsk, npk) ←$ NIKE.Gen
(ksk, kpk) ←$ KEM.Gen
(ssk, spk) ←$ RSig.Gen
sk := (nsk, ksk, ssk)
pk := (npk, kpk, spk)
return (sk, pk)
```

```
Enc(sk_s, pk_r)
parse sk_s → (nsk_s, ksk_s, ssk_s)
parse pk_r → (npk_r, kpk_r, spk_r)
(nsk_e, npk_e) ←$ NIKE.Gen
nk' ← NIKE.Sdk(nsk_s, npk_r)
nk := H_1(nk', "auth")
nk_1‖nk_2 ← NIKE.Sdk(nsk_e, npk_r)
(kct, kk_1‖kk_2) ←$ KEM.Enc(kpk_r)
m ← (kct, kpk_r)
σ ← RSig.Sgn(ssk_s, {μ(ssk_s), spk_r}, m)
k' := H_1(nk_1, kk_1)
sct := SE.Enc(k', σ)
c := (npk_e, kct, sct)
k := H_2(nk, nk_2, kk_2, c, μ(sk_s), pk_r)
return (c, k)
```

```
Dec(pk_s, sk_r, c)
parse pk_s → (npk_s, kpk_s, spk_s)
parse sk_r → (nsk_r, ksk_r, ssk_r)
parse c → (npk_e, kct, sct)
nk' ← NIKE.Sdk(nsk_r, npk_s)
nk := H_1(nk', "auth")
nk_1‖nk_2 ← NIKE.Sdk(nsk_r, npk_e)
kk_1‖kk_2 ← KEM.Dec(ksk_r, kct)
k' := H_1(nk_1, kk_1)
σ := SE.Dec(k', sct)
m ← (kct, μ(ksk_r))
if RSig.Ver(σ, ρ = {spk_s, μ(ssk_r)}, m) ≠ 1
    return ⊥
k := H_2(nk, nk_2, kk_2, c, pk_s, μ(sk_r))
return k
```

Figure 6: Concrete construction of a deniable AKEM $\mathsf{AKEM}[\mathsf{NIKE}, \mathsf{KEM}, \mathsf{RSig}, \mathsf{SE}, \mathsf{H}_1, \mathsf{H}_2]$**. By "‖" we denote that an output is split into two equal parts.**

holds

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

```
G_0 – G_5
i* ←$ {0,1}                                  / G_1 – G_5
R, C ← ∅
for i ∈ {0,1}
    (nsk_i, npk_i) ←$ NIKE.Gen
    (ksk_i, kpk_i) ←$ KEM.Gen
    (ssk_i, spk_i) ←$ RSig.Gen
    sk_i := (nsk_i, ksk_i, ssk_i)
    pk_i := (npk_i, kpk_i, spk_i)
b ←$ {0,1}
b' ← A^{Rev,Chall}(pk_0, pk_1)
if R ∩ C ≠ ∅
    abort
return ⟦b = b'⟧
```

```
Rev(i ∈ {0,1})
if i = i*                                    / G_1 – G_5
    abort                                    / G_1 – G_5
R ← R ∪ {i}
return sk_i
```

```
Sim(pk_s, pk_r)
(nsk_e, npk_e) ←$ NIKE.Gen
(kct, kk) ←$ KEM.Enc(kpk_r)
k' ←$ K_{H_1}
sct := SE.Enc(k', 0)
c := (npk_e, kct, sct)
k ←$ K_{H_2}
return (c, k)
```

```
Oracle Chall(s ∈ {0,1}, r ∈ {0,1})           / one query
if s = r return ⊥
if r ≠ i*                                    / G_1 – G_5
    abort                                    / G_1 – G_5
C ← C ∪ {r}
(nsk_e, npk_e) ←$ NIKE.Gen
nk' ← NIKE.Sdk(nsk_s, npk_r)
nk := H_1(nk', "auth")
nk_1‖nk_2 ← NIKE.Sdk(nsk_e, npk_r)
nk_1‖nk_2 ←$ K_NIKE                          / G_2.1 – G_5
(kct, kk_1‖kk_2) ←$ KEM.Enc(kpk_r)
kk_1‖kk_2 ←$ K_KEM                           / G_2.2 – G_5
m ← (kct, kpk_r)
σ ← RSig.Sgn(ssk_s, {spk_s, spk_r}, m)
k' := H_1(nk_1, kk_1)
k' ←$ K_{H_1}                                / G_3 – G_5
σ := 0                                       / G_5
sct := SE.Enc(k', σ)
c := (npk_e, kct, sct)
k := H_2(nk, nk_2, kk_2, c, pk_s, pk_r)
k ←$ K_{H_2}                                 / G_4 – G_5
if b = 1
    (c, k) ←$ Sim(pk_s, pk_r)
return (c, k)
```

Figure 7: Games $\mathsf{G}_0 - \mathsf{G}_5$ **for the proof of Theorem 9.**

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 1:

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

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 ($\mathcal{R} \cap C \neq \emptyset$), 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 2: There exists an adversary $\mathcal{B}$ against **CKS** security of NIKE such that

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

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 8. 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.

∎

```
𝓑^{RevCor,Test}(npk₁*, npk₂*)
i* ←$ {0,1}
𝓡, 𝓒 ← ∅
npk_{i*} := npk₁*
nsk_{i*} := ⊥
(nsk_{1−i*}, npk_{1−i*}) ←$ NIKE.Gen
for i ∈ {0,1}
    (ksk_i, kpk_i) ←$ KEM.Gen
    (ssk_i, spk_i) ←$ RSig.Gen
    sk_i := (nsk_i, ksk_i, ssk_i)
    pk_i := (npk_i, kpk_i, spk_i)
b ←$ {0,1}
b′ ← 𝓐^{Rev,Chall}(pk₀, pk₁)
if 𝓡 ∩ 𝓒 ≠ ∅
    abort
return ⟦b = b′⟧

Rev(i ∈ {0,1})
if i = i*
    abort
𝓡 ← 𝓡 ∪ {i}
return sk_i

Oracle Chall(s ∈ {0,1}, r ∈ {0,1})                      / one query
if r ≠ i*
    abort
𝓒 ← 𝓒 ∪ {r}
npk_e := npk₂*                                          / embed second honest key
nk′ ← NIKE.Sdk(nsk_{1−i*}, npk_{i*})                    / simulateable due to abort
nk := H₁(nk′, "auth")
nk₁‖nk₂ ←$ Test(2,1)                                    / test query
(kct, kk₁‖kk₂) ←$ KEM.Enc(kpk_r)
m ← (kct, kpk_r)
σ ← RSig.Sgn(ssk_s, {spk_s, spk_r}, m)
k′ := H₁(nk₁, kk₁)
sct := SE.Enc(k′, σ)
c := (npk_e, kct, sct)
k := H₂(nk, nk₂, kk₂, c, pk_s, pk_r)
if b = 1
    (c, k) ←$ Sim(pk_s, pk_r)
return (c, k)
```

Figure 8: Adversary $\mathcal{B}$ against CKS security of NIKE, having access to oracles RevCor and Test, simulating Game G_1/$G_{2.1}$ for adversary $\mathcal{A}$ from the proof of Theorem 9.

Game $G_{2.2}$. 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 3: There exists an adversary C against **IND-CPA** security of KEM such that

$$\left|\Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_{2.2}^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{(1,1)\text{-}\mathbf{IND\text{-}CPA}}_{\mathsf{KEM},C}.$$

Proof. The claim can be proved straightforward 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 $G_{2.1}$/$G_{2.2}$ except that the output of H_1 is replaced by a uniformly random value of the output range $\mathcal{K}_{H_1}$.

Claim 4: There exists an adversary $\mathcal{D}$ against **mPRF** security of H_1 such that for $i \in \{1, 2\}$

$$\left|\Pr\left[G_{2.i}^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{(1,1)\text{-}\mathbf{mPRF}}_{H_1,\mathcal{D}}.$$

Proof. The proof can be done straightforward 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. ■

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 5: There exists an adversary $\mathcal{E}$ against **mPRF** security of H_2 such that

$$\left|\Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{(1,1)\text{-}\mathbf{mPRF}}_{H_2,\mathcal{E}}.$$

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 6: There exists an adversary $\mathcal{F}$ against **IND-CPA** security of SE such that

$$\left|\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{\mathbf{IND\text{-}CPA}}_{\mathsf{SE},\mathcal{F}}.$$

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

$$\Pr[G_5^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

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.

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

■

6 INSTANTIATION

In this paper, we first implement the Gandalf ring signature scheme proposed by [GJK24] with some modifications to the trapdoor generation procedure Mitaka [EFG+22] to ensure functionality. For the post-quantum AKEM [GJK24, Fig. 10], the authors chose the NTRU-A KEM by [DHK+23]. We replace their choice by `bat_257_512` [FKPY22] and integrate the implementation provided by [FKPY22] to reduce the public key and ciphertext sizes. As for the hybrid AKEM Shadowfax, we integrate a `Curve25519` implementation from [DT24] and the `bat_257_512` by [FKPY22] with our implementation of Gandalf. Table 1 summarises the instantiations chosen for each of the primitives of the PQ-AKEM by [GJK24] and the hybrid AKEM Shadowfax. All our source code is publicly available on the GitHub repository Shadowfax.

Table 1: Sizes of different deniable AKEMs.

Scheme	Size (in bytes)			Implementation
	pk	c	σ	
PQ-AKEM [GJK24]				
Gandalf	896	—	1 276	✓ (This work)
NTRU-A	768	768	—	✗
Total	1 664		2 044	✗
PQ-AKEM [GJK24]				
Gandalf	896	—	1 276	✓ (This work)
bat_257_512	521	473	—	✓ [FKPY22]
Total	1 417		1 749	✓ (This work)
Shadowfax [Figure 6]				
Curve25519	32	32	—	✓ [DT24]
Gandalf	896	—	1 276	✓ (This work)
bat_257_512	521	473	—	✓ [FKPY22]
Total	1 449		1 781	✓ (This work)

The signature size of Gandalf is 40-byte larger than the original proposal by [GJK24] due to the difference in compression techniques. The authors of [GJK24] based their claims on the compression technique from [ETWY22, EFG+22], while we deploy the compression technique from the round 3 submission package of Falcon.

Optimisation Goals. We aim for portability and the compactness of public key and ciphertext sizes 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, there are rich C reference implementations for several post-quantum cryptosystems We follow a similar paradigm and implement the AKEMs with the C programming language. Since C is a high-level programming language, our instantiations are portable. Platform-specific optimisations on popular architectures like Armv8-A and x86-64 with AVX2/AVX512 are left as future work.

6.1 Basic Constructs

Hash. Our instantiations use four distinct hash functions. The first is `BLAKE2b` [SA15] which is shipped with our choice of KEM. The second is `shake128` [KjCP16] used internally in our choice of 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 6). 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:

$$\mathsf{H}_2(nk, nk_2, kk_2, c, \mu(sk_s), pk_r) = \texttt{HMAC-SHA3-256}\left(nknk_2, \texttt{[rest]}_{kk_2}\right)$$

where $nknk_2 = \texttt{HMAC-SHA3-256}\,(nk, nk_2)$, $\texttt{[rest]}_{kk_2} = \texttt{HMAC-SHA3-256}\,(kk_2, \texttt{[rest]})$, 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 6. HMAC has been proven to be a dual-PRF [BBGS23] and the consecutive calls as described above instantiate a multi-PRF.

Symmetric Encryption. We choose the CTR mode of `AES-128` for the symmetric encryption.

NTRU Solver. In our choices of KEM and ring signature, we have to solve for polynomials $F, G \in \mathbb{Z}[X] \big/ \left\langle X^N + 1 \right\rangle$ 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 $g, f \in \mathbb{Z}_q[X] \big/ \left\langle X^N + 1 \right\rangle$ with small coefficients. We integrate the latest NTRU solver by [Por23] to our KEM and ring signature.

NIKE. 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.

KEM. We choose the `bat_257_512` parameter set from BAT [FKPY22] for the KEM. BAT is a CCA secure KEM based on NTRU with a GGH-like [GGH97] internal encryption and achieves the smallest ciphertext size among the post-quantum KEMs to the best of our knowledge. We integrate the latest NTRU solver by [Por23], enforce the uses of `BLAKE2b` in encapsulation and decapsulation, and simplify the code base with the C preprocessor. The rest of the KEM remains the same as the reference implementation.

6.1.1 *Ring Signature.*

We choose Gandalf [GJK24] for the ring signature. According to [GJK24], Gandalf achieves the smallest signature size for the ring of size 2, which suits well for constructing our AKEM. For the key generation of Gandalf, we follow the Antrag trapdoor generation [ENS+23] and integrate the latest NTRU solver by [Por23]. For the signature generation, we choose the Mitaka [EFG+22] with hybrid sampler [Pre15] 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 [GJK24]. 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 [GJK24]. 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 Mitatka 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.

6.2 AKEMs

There are three AKEMs implemented in this paper: the pre-quantum one, the post-quantum one, and the hybrid one. For

Table 2: Comparison of different AKEMs along with their security notions and whether they rely on pre-quantum (pre-Q) or post-quantum (post-Q) assumptions.

Scheme (variant)	Confidentiality	Authenticity	Deniability	Assumption		Size (in bytes)	
				pre-Q	post-Q	*c*	*pk*
DH-AKEM (X25519) [ABH+21, Lst. 10]	**Ins-CCA**	**Out-Aut**	**DR-Den***	✓	✗	32	32
EtStH-AKEM (BAT + Antrag) [AJKL23, Lst. 18]	**Ins-CCA**	**Out-Aut**	—	✗	✓	1 119	1 417
NIKE-AKEM (Swoosh) [AJKL23, Lst. 19]	**Ins-CCA**	**Out-Aut**	**DR-Den***	✗	✓	> 221 184	> 221 184
FrodoKEX+ [CHN+24, Fig. 12]	**IND-1BatchCCA**	**UNF-1KCA**	**DR-Den**	✗	✓	72	21 300
PQ-AKEM (NTRU-A + Gandalf) [GJK24, Fig. 10]	**Ins-CCA**	**Out-Aut**	**HR-Den & DR-Den**	✗	✓	2 044	1 664
PQ-AKEM (BAT + Gandalf) [GJK24, Fig. 10]						1 749	1 417
Shadowfax (X25519 + BAT + DualRing) [This work, Fig. 6]	**Ins-CCA**	**Out-Aut**	**HR-Den & DR-Den**	✓	✓	5 093	3 393
Shadowfax (X25519 + BAT + Gandalf) [This work, Fig. 6]						1 781	1 449

Deniability properties marked with a "*" have not been formally proven in the respective works.
The Swoosh [GdKQ+24] size refers to a passively secure NIKE. For an active secure NIKE a NIZK is needed and the size of a proof must be added to the NIKE public key.
DualRing [YEL+21] is included in the table because the parameters of Gandalf would need to be slightly increased for stronger concrete anonymity (see [GJK24] for further details).

the pre-quantum AKEM, we implement the DH-AKEM by [ABH+21]. For the post-quantum AKEM, we implement the AKEM by [GJK24] with the CCA-secure KEM `bat_257_512` [FKPY22] and the ring signature Gandalf [GJK24]. Compared to the original proposal by [GJK24] with CCA-NTRU-A [DHK+23], our post-quantum AKEM with `bat_257_512` achieves smaller public key and ciphertext sizes. For the hybrid AKEM Shadowfax, we choose X25519 for the NIKE, `bat_257_512` for the KEM, and Gandalf for the ring signature. We compare the security notions (confidentiality, authenticity, and deniability), pre-/post-quantum assumption, public key size, and ciphertext size to other AKEMs in Table 2.

7 PERFORMANCE

Table 3: 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.**

AKEM	Unit	Gen	Enc	Dec
DH-AKEM	kcc	227	679	457
[ABH+21, Lst. 10]	ms	0.08	0.23	0.15
PQ-AKEM	kcc	25 420	1 256	349
[GJK24, Fig. 10]	ms	8.47	0.42	0.12
Shadowfax	kcc	25 655	1 936	796
[Fig. 6]	ms	8.55	0.65	0.27
RSig	**Unit**	**Gen**	**Sgn**	**Ver**
Gandalf	kcc	13 423	1 113	100
[GJK24, Fig. 5]	ms	4.47	0.37	0.03
Raptor	kcc	71 420	7 980	505
[LAZ19, Zha20]	ms	23.81	2.66	0.17

For the post-quantum AKEM (PQ-AKEM) from [GJK24], we instantiate the underlying KEM with `bat_257_512`.

Benchmarking Environment. We benchmark our portable C 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 aiming for application uses. It runs at the frequency of 3GHz and comes with a dedicated cryptographic extension. As we aim for portable C 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 3 summarises the cycle counts of the C implementations of DH-AKEM, PQ-AKEM, and Shadowfax, and Table 4 profiles the dominating operations in Shadowfax. For the key generations in PQ-AKEM and Shadowfax, the cycle count is dominated by two calls to the NTRU solver by [Por23]. 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 `bat_257_512` in PQ-AKEM. For completeness, we also give the cycle counts of our C implementation of Gandalf and the C implementation of Raptor by the authors of [LAZ19]. We stress that the C implementation of Raptor by [LAZ19] 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 combiner 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 combiner should always be considered.

Table 4: Cycle counts of our portable C implementation of Shadowfax [Fig. 6].

AKEM.Gen			
NIKE.Gen	227k	0.08 ms	0.88%
KEM.Gen	12 013k	4.00 ms	46.82%
RSig.Gen	13 334k	4.44 ms	51.97%
Hash, SE, and others	—	—	—
Total	25 655k	8.55 ms	100%
AKEM.Enc			
NIKE.Gen	227k	0.08 ms	11.71%
NIKE.Sdk(×2)	454k	0.15 ms	23.44%
KEM.Enc	57k	0.02 ms	2.94%
RSig.Sgn	1 103k	0.37 ms	56.98%
Hash, SE, and others	—	—	4.92%
Total	1 936k	0.65 ms	100%
AKEM.Dec			
NIKE.Sdk(×2)	454k	0.15 ms	57.03%
KEM.Dec	230k	0.08 ms	28.97%
RSig.Ver	85k	0.03 ms	10.63%
Hash, SE, and others	—	—	3.37%
Total	796k	0.27 ms	100%

Cycle counts of hash functions (H_1 and H_2) and symmetric encryption (SE) are omitted since they are not the dominating operations.

ACKNOWLEDGMENTS

We thank the anonymous USENIX reviewers and Daniel Collins for their helpful feedback. 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.

[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.

[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.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108, Philadephia, PA, USA, May 22–24, 1996. ACM Press. doi:10.1145/237814.237838.

[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.

[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/.

[BBC16] BBC. 18 revelations from wikileaks' hacked clinton emails. *BBC*, 2016. URL: https://www.bbc.com/news/world-us-canada-37639370.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[BGB04] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, WPES '04, page 77–84, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/1029179.1029200.

[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.

[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.

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

[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.

[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.

[Car16] Lauren Carroll. Are the clinton wikileaks emails doctored, or are they authentic? *PolitiFact*, 2016. URL: https://www.politifact.com/article/2016/oct/23/are-clinton-wikileaks-emails-doctored-or-are-they-/.

[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.

[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.

[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.

[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.

[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.

[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.

[Dac16] Dana Dachman-Soled. Towards non-black-box separations of public key encryption and one way function. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 169–191, Beijing, China, October 31 – November 3, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-53644-5_7.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

[DHK+23] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, Gregor Seiler, and Dominique Unruh. A thorough treatment of highly-efficient NTRU instantiations. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 65–94, Atlanta, GA, USA, May 7–10, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-31368-4_3.

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

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

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski, Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131, Santa Barbara, CA, USA, August 17–21, 1997. Springer Berlin Heidelberg, Germany. doi:10.1007/BFb0052231.

[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.

[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.

[GJK24] 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.

[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.

[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.

[KS24] Ehren Kret and Rolfe Schmidt. The pqxdh key agreement protocol, 2024. URL: https://signal.org/docs/specifications/pqxdh/pqxdh.pdf.

[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.

[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/.

[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.

[Lan16] Adam Langley. CECPQ1 results. Blog post, 2016. https://www.imperialviolet.org/2016/11/28/cecpq1.html.

[Lan18] Adam Langley. CECPQ2. Blog post, 2018. https://www.imperialviolet.org/2018/12/12/cecpq2.html.

[LAZ19] Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 110–130, Bogota, Colombia, June 5–7, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-21568-2_6.

[LF07] Barry Leiba and Jim Fenton. Domainkeys identified mail (dkim): Using digital signatures for domain verification. In *International Conference* *on Email and Anti-Spam*, 2007. URL: https://api.semanticscholar.org/CorpusID:28916525.

[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.

[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.

[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.

[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.

[Mas16] Mike Masnick. The clinton campaign should stop denying that the wikileaks emails are valid; they are and they're real. *Techdirt*, 2016. URL: https://www.techdirt.com/2016/10/25/clinton-campaign-should-stop-denying-that-wikileaks-emails-are-valid-they-are-theyre-real/.

[Met23] Meta. Messenger End-to-End Encryption Overview, v.1, dec 2023. https://engineering.fb.com/wp-content/uploads/2023/12/MessengerEnd-to-EndEncryptionOverview_12-6-2023.pdf.

[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.

[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.

[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.

[MP16] Moxie Marlinspike and Trevor Perrin. The x3dh key agreement protocol, 2016. URL: https://signal.org/docs/specifications/x3dh/x3dh.pdf.

[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.

[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/.

[OPowp19] David Ott, Christopher Peikert, and other workshop participants. Identifying research challenges in post quantum cryptography migration and cryptographic agility, 2019. URL: https://arxiv.org/abs/1909.07353, arXiv:1909.07353.

[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. 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.

[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.

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

[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.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. doi:10.1145/1060590.1060603.

[Riv92] Ronald L. Rivest. *RFC 1321: The MD5 Message-Digest Algorithm*. Internet Activities Board, April 1992.

[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.

[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.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, MD, USA, May 14–16, 1990. ACM Press. doi:10.1145/100216.100269.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978. doi:10.1145/359340.359342.

[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.

[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.

[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.

[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/.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949. doi:10.1002/j.1538-7305.1949.tb00928.x.

[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.

[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.

[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.

[Ste24] Douglas Stebila. Security analysis of the iMessage PQ3 protocol. Cryptology ePrint Archive, Report 2024/357, 2024. URL: https://eprint.iacr.org/2024/357.

[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.

[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.

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

[Wik16] WikiLeaks. The Podesta Emails. *WikiLeaks*, 2016. URL: https://wikileaks.org/podesta-emails/.

[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/.

[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.

[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.

[Zha20] Zhenfei Zhang. Raptor. https://github.com/zhenfeizhang/raptor, 2020.

A ADDITIONAL PRELIMINARIES

A.1 Non-Interactive Key Exchange (NIKE)

Definition 2 ((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 $\mathcal{K}_{\mathsf{NIKE}}$.

$(sk, pk) \xleftarrow{\$} \mathsf{Gen}$**:** 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 $\bot$. We assume that Sdk always returns $\bot$ if sk is the secret key corresponding to pk.

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

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

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 $(n, Q_{\mathsf{RC}}, Q_{\mathsf{T}})\text{-}\mathbf{CKS}_{\mathsf{NIKE},par}(\mathcal{A})$ depicted in Figure 9 and define the advantage of adversary $\mathcal{A}$ as

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

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 10. 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

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

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

Proof. The reduction for the first inequality is depicted in Figure 10, the reduction for the second in Figure 11. ∎

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

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

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

$k \leftarrow \mathsf{Sdk}(sk_i, pk)$
return k

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

if $i = j$ **return** $\bot$
if $b = 0$
  $k \leftarrow \mathsf{Sdk}(sk_i, pk_j)$
if $b = 1$
  if $\exists\, k' : (\{pk_i, pk_j\}, k') \in \mathcal{D}$
    $k \leftarrow k'$
  else
    $k \xleftarrow{\$} \mathcal{K}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\{pk_i, pk_j\}, k)\}$
return k

Figure 9: Games defining CKS for a simplified non-interactive key exchange NIKE with adversary $\mathcal{A}$ making at most Q_{RC} queries to RevCor and at most Q_{T} queries to Test.

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

for $i \in [n]$
  $pk_i \xleftarrow{\$} \mathsf{RegHonUsr}_{\mathcal{B}}()$
$C := \emptyset$
$b' \leftarrow \mathcal{A}^{\mathsf{RevCor},\mathsf{Test}}(pk_1, \ldots, pk_n)$
return b'

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

if $pk \notin C$
  $\mathsf{RegCorUsr}_{\mathcal{B}}(pk)$
  $C := C \cup \{pk\}$
$k \xleftarrow{\$} \mathsf{RevCor}_{\mathcal{B}}(pk_i, pk)$
return k

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

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

Figure 10: Reduction for CKS-Orig ⇒ CKS.

A.2 Key Encapsulation Mechanism

Definition 3 (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 $\bot$.

$\mathcal{B}^{\mathsf{RevCor}_{\mathcal{B}},\mathsf{Test}_{\mathcal{B}}}(pk_1,\dots,pk_n)$

$C := \emptyset$
$i := 0$
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{RegHonUsr},\mathsf{RegCorUsr},\mathsf{RevCor},\mathsf{Test}}$
return b'

Oracle $\mathsf{RegHonUsr}()$

$i := i+1$
return pk_i

Oracle $\mathsf{RegCorUsr}(pk)$

$C := C \cup \{pk\}$

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

if $\exists\, j : pk = pk_j \wedge pk' \in C$
  $k \xleftarrow{\$} \mathsf{RevCor}_{\mathcal{B}}(j, pk')$
  return k
return $\perp$

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

if $\exists\, j, j' : pk = pk_j \wedge pk' = pk_{j'}$
  $k \xleftarrow{\$} \mathsf{Test}_{\mathcal{B}}(j, j')$
  return k
return $\perp$

Figure 11: Reduction for CKS ⇒ CKS-Orig.

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

for $i \in [n]$
  $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{Dec},\mathsf{Chall}}(pk_1,\dots,pk_n)$
return $[\![b = b']\!]$

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

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

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

$(c,k) \xleftarrow{\$} \mathsf{Enc}(pk_r)$
if $b = 0$
  continue
if $b = 1$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_r, c, k)\}$
return (c,k)

Figure 12: Game defining IND-CCA for a key encapsulation mechanism KEM **with adversary $\mathcal{A}$ making at most Q_{Dec} queries to Dec and at most Q_{Chl} queries to Chl.**

The correctness error δ_{KEM} is defined as

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

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

The γ-spreadness of a KEM is defined as

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

where 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 $(n, Q_{\mathsf{Dec}}, Q_{\mathsf{Chl}})$-**IND-CCA**$_{\mathsf{KEM}}(\mathcal{A})$ depicted in Figure 12 and define the advantage of adversary $\mathcal{A}$ as

$$\mathrm{Adv}^{(\mathrm{n},\mathrm{Q_{Dec}},\mathrm{Q_{Chl}})\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{A}} := \left|\Pr\left[(n, Q_{\mathsf{Dec}}, Q_{\mathsf{Chl}})\text{-}\mathbf{IND\text{-}CCA}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2}\right|,$$
$$\mathrm{Adv}^{(\mathrm{n},\mathrm{Q_{Chl}})\text{-}\mathbf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{A}} := \mathrm{Adv}^{(\mathrm{n},0,\mathrm{Q_{Chl}})\text{-}\mathbf{IND\text{-}CCA}}_{\mathsf{KEM},\mathcal{A}}.$$

A.3 Ring Signatures

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

Definition 4 (Ring Signature)**.** A *ring signature* scheme RSig is defined as a tuple (Gen, Sgn, Ver) of the following algorithms.

- $par \xleftarrow{\$} \mathsf{Stp}(\kappa)$**:** Given an upper bound on the ring size ρ, the probabilistic setup algorithm Stp returns system parameters par, where par defines a message space $\mathcal{M}$. We assume that all algorithms are implicitly given access to the system parameters par.
- $(sk, pk) \xleftarrow{\$} \mathsf{Gen}$**:** The probabilistic key generation algorithm returns a secret key sk and a corresponding public key pk.
- $\sigma \xleftarrow{\$} \mathsf{Sgn}(sk, \rho, m)$**:** Given a secret key sk, a ring $\rho = \{pk_1, \dots, pk_k\}$ 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 $\rho := \{pk_1, \dots, pk_k\}$, and the probability is taken over the random choices of Stp, Gen and Sgn.

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

Unforgeability. 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 [GJK24]. The notion is formalised through the game $(n, \kappa, Q_{\mathsf{Sgn}})$-**UF-CRA1**$_{\mathsf{RSig}}(\mathcal{A})$ depicted in Figure 13, where n is the number of users, κ the maximal ring size, and Q_{Sgn} is an upper bound on the signing queries. We define the advantage functions of adversary $\mathcal{A}$ as

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

Anonymity. We consider *multi-challenge anonymity under full key exposures* of a ring signature RSig from [BFG+22, GJK24]. It is defined via the game $(n, \kappa, Q_{\mathsf{Chl}})$-**MC-Ano**$_{\mathsf{RSig}}(\mathcal{A})$ for an

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

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

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

if $pk_i \notin \rho \vee (\rho, m) \in Q$
  return $\perp$
$\sigma \xleftarrow{\$} \mathsf{Sgn}(sk_i, \rho, m)$
$Q \leftarrow Q \cup \{(\rho, m)\}$
return σ

Figure 13: Game UF-CRA1 for a ring signature scheme RSig **and adversary** $\mathcal{A}$**.**

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

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

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

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

Figure 14: Game defining MC-Ano for a ring signature scheme RSig **with adversary** $\mathcal{A}$ **making at most** $Q_{\mathtt{Chl}}$ **queries to** $\mathtt{Chl}$**.**

adversary $\mathcal{A}$, depicted in Figure 14. We define the advantage as

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

A.4 Pseudorandom Function

Definition 5 (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 *pseudorandomness* for a keyed function F via the game $(n, Q_{\mathtt{Eval}})$-**PRF** depicted in Figure 15 and define the advantage of adversary $\mathcal{A}$ as

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

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. We generalise this even further by defining a multi-key PRF. 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 multi-PRF can be generically instantiated by calling a dual-PRF multiple times sequentially.

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

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

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

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

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

Definition 6 (Multi-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 *multi-key pseudorandomess* for a multi-keyed function F_m via the game $(n, Q_{\mathtt{Eval}})$-**mPRF** depicted in Figure 16 and define the advantage of adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as

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

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

$j \xleftarrow{\$} \mathcal{A}_1$
$\mathcal{K}' := \prod_{\ell \in [m] \setminus \{j\}} \mathcal{K}_\ell$
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 16: Game defining mPRF 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** $\mathtt{Eval}$**.**

A.5 Symmetric Encryption

Definition 7 (Symmetric Encryption)**.** A *symmetric encryption* SE is defined as a tuple SE := (Enc, Dec) of the following algorithms.

Game $\mathbf{IND\text{-}CPA}_{\mathsf{SE}}(\mathcal{A})$

$k \xleftarrow{\$} \mathcal{K}_{\mathsf{SE}}$
$b \xleftarrow{\$} \{0, 1\}$
$b' \leftarrow \mathcal{A}^{\mathtt{Chall}}$
return $[\![b = b']\!]$

Oracle $\mathtt{Chl}(m_0, m_1)$ / one query

$c := \mathsf{Enc}(k, m_b)$
return c

Figure 17: Game defining IND-CPA for a symmetric encryption scheme SE with adversary $\mathcal{A}$ making at most one query `Chl`.

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

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

We formalise the notion of ciphertext indistinguishability (**IND-CPA**) for a symmetric encryption scheme SE via the game $\mathbf{IND\text{-}CPA}_{\mathsf{SE}}(\mathcal{A})$ depicted in Figure 17 and define the advantage of adversary $\mathcal{A}$ as

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

B PROOFS FOR SECTION 4 (GENERIC CONSTRUCTION)

Theorem 2 (Confidentiality). *For any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, depicted in Figure 5, there exists an* **Ins-CCA** *adversary* $\mathcal{B}_1$ *against* AKEM_1*, an* **Ins-CCA** *adversary* $\mathcal{B}_2$ *against* AKEM_2*, and a* **mPRF** *adversary* C *against* H *such that*

$$\mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathtt{Ch1}})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathcal{A}} \le \min\Big\{\mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathtt{Ch1}})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}_1,\mathcal{B}_1}, \mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathtt{Ch1}})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}_2,\mathcal{B}_2}\Big\} + \mathrm{Adv}^{(Q_{\mathtt{Ch1}},Q_{\mathsf{Dec}}+Q_{\mathtt{Ch1}})\text{-}\mathbf{mPRF}}_{\mathsf{H},C} + Q_{\mathtt{Ch1}} \cdot \delta_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}]}.$$

Proof. Consider the sequence of games depicted in Figure 18.

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

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathtt{Ch1}})\text{-}\mathbf{Ins\text{-}CCA})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathcal{A}}.$$

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 $Q_{\mathtt{Ch1}}$ queries to the challenge oracle, we obtain

$$\left|\Pr\left[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right]\right| \le Q_{\mathtt{Ch1}} \cdot \delta_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}]}.$$

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

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right]\right| \le \mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathtt{Ch1}})\text{-}\mathbf{Ins\text{-}CCA}}_{\mathsf{AKEM}_1,\mathcal{B}_1}.$$

Proof. Adversary $\mathcal{B}_1$ is formally constructed in Figure 19. 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 $\mathcal{B}_1$. ∎

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

Claim 8: There exists an adversary C_1 against the **PRF** security of H_1, i.e. keyed on the first input, such that

$$\left|\Pr\left[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1\right]\right| \le \mathrm{Adv}^{(Q_{\mathtt{Ch1}},Q_{\mathsf{Dec}}+Q_{\mathtt{Ch1}})\text{-}\mathbf{PRF}}_{\mathsf{H}_1,C_1}.$$

Proof. We formally construct adversary C_1 in Figure 20. If 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

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

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

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

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 $c \to (c_1, c_2)$
$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk^{(1)}, sk^{(1)}, c_1)$
if $\exists\, k_1' : (pk^{(1)}, \mu(sk^{(1)}), c_1, k_1') \in \mathcal{E}_1$ / $\mathsf{G}_2, \mathsf{G}_3$
  $k_1 := k_1'$ / $\mathsf{G}_2, \mathsf{G}_3$
$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk^{(2)}, sk^{(2)}, c_2)$
if $\exists\, k_2' : (pk, \mu(sk), c_2, k_2') \in \mathcal{E}_2$ / $\mathsf{G}_4, \mathsf{G}_5$
  $k_2 := k_2'$ / $\mathsf{G}_4, \mathsf{G}_5$
$k := \mathsf{H}(k_1, k_2, pk, pk_r, c)$
return k

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

parse $sk_s \to (sk^{(1)}, sk^{(2)})$
parse $pk \to (pk^{(1)}, pk^{(2)})$
$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk^{(1)}, pk^{(1)})$
$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
$c := (c_1, c_2)$
$k := \mathsf{H}(k_1, k_2, pk_s, pk, c)$
return (c, k)

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

parse $sk \to (sk^{(1)}, sk^{(2)})$
parse $pk_r \to (pk^{(1)}, pk^{(2)})$
$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk^{(1)}, pk^{(1)})$
$k_1 \xleftarrow{\$} \mathcal{K}_1$ / $\mathsf{G}_2, \mathsf{G}_3$
$\mathcal{E}_1 := \mathcal{E}_1 \cup \{(\mu(sk^{(1)}), pk^{(1)}, c_1, k_1)\}$ / $\mathsf{G}_2, \mathsf{G}_3$
$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
$k_2 \xleftarrow{\$} \mathcal{K}_2$ / $\mathsf{G}_4, \mathsf{G}_5$
$\mathcal{E}_2 := \mathcal{E}_2 \cup \{(\mu(sk^{(2)}), pk^{(2)}, c_2, k_2)\}$ / $\mathsf{G}_4, \mathsf{G}_5$
$c := (c_1, c_2)$
$k := \mathsf{H}(k_1, k_2, \mu(sk), pk_r, c)$
$k \xleftarrow{\$} \mathcal{K}$ / $\mathsf{G}_3, \mathsf{G}_5$
if $b = 1$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), pk_r, c, k)\}$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), pk_r, c, k)\}$ / $\mathsf{G}_1 - \mathsf{G}_5$
return (c, k)

Figure 18: Games $\mathsf{G}_0 - \mathsf{G}_5$ for the proof of Theorem 2.

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

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 $b' \leftarrow \mathcal{A}^{\mathsf{Encps},\mathsf{Decps},\mathsf{Chall}}(pk_1, \ldots, pk_n)$
08 **return** $[\![b = b']\!]$

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

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 $k_1 \leftarrow \mathsf{Decps}_{\mathcal{B}}(pk^{(1)}, r, c_1)$ / 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 $\mathsf{Encps}(s \in [n], pk)$

18 **parse** $sk_s \to (\bot, sk^{(2)})$
19 **parse** $pk \to (pk^{(1)}, pk^{(2)})$
20 $(c_1, k_1) \xleftarrow{\$} \mathsf{Encps}_{\mathcal{B}}(s, pk^{(1)})$ / 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 $\mathsf{Chall}(sk, r \in [n])$

25 **parse** $sk \to (sk^{(1)}, sk^{(2)})$
26 **parse** $pk_r \to (pk^{(1)}, pk^{(2)})$
27 $(c_1, k_1) \xleftarrow{\$} \mathsf{Chall}_{\mathcal{B}}(sk^{(1)}, r)$ / 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 19: Adversary $\mathcal{B}_1$ against Ins-CCA security of AKEM_1, having access to oracles $\mathsf{Encps}_{\mathcal{B}}$, $\mathsf{Decps}_{\mathcal{B}}$, $\mathsf{Chall}_{\mathcal{B}}$, and $\mathsf{CorSK}_{\mathcal{B}}$, simulating Game $\mathsf{G}_1/\mathsf{G}_2$ for adversary $\mathcal{A}$ from the proof of Theorem 2.

$\mathcal{C}_1^{\mathsf{Eval}}$

01 $\mathcal{D}, \mathcal{E}_1 := \emptyset$
02 $\ell := 0$
03 **for** $i \in [n]$
04 $\quad (sk^{(1)}, pk^{(1)}) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$
05 $\quad (sk^{(2)}, pk^{(2)}) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$
06 $\quad sk_i := (sk^{(1)}, sk^{(2)})$
07 $\quad pk_i := (pk^{(1)}, pk^{(2)})$
08 $b \xleftarrow{\$} \{0,1\}$
09 $b' \leftarrow \mathcal{A}^{\mathsf{Encps},\mathsf{Decps},\mathsf{Chall}}(pk_1, \ldots, pk_n)$
10 **return** $[\![b = b']\!]$

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

11 **if** $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
12 $\quad$ **return** k
13 **parse** $pk \to (pk^{(1)}, pk^{(2)})$
14 **parse** $sk_r \to (sk^{(1)}, sk^{(2)})$
15 **parse** $c \to (c_1, c_2)$
16 $k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk^{(1)}, sk^{(1)}, c_1)$
17 $k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk^{(2)}, sk^{(2)}, c_2)$
18 $k := \mathsf{H}(k_1, k_2, pk, pk_r, c)$
19 **if** $\exists\, \ell' : (pk^{(1)}, \mu(sk^{(1)}), c_1, \ell') \in \mathcal{E}_1$
20 $\quad k \xleftarrow{\$} \mathsf{Eval}(\ell', k_2 || pk || pk_r || c)$ / call on previous key
21 **return** k

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

22 **return** $\mathsf{G}_2.\mathsf{Encps}(s, pk)$

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

23 **parse** $sk \to (sk^{(1)}, sk^{(2)})$
24 **parse** $pk_r \to (pk^{(1)}, pk^{(2)})$
25 $(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk^{(1)}, pk^{(1)})$
26 $k_1 \xleftarrow{\$} \mathcal{K}_1$
27 $(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
28 $c := (c_1, c_2)$
29 $\ell := \ell + 1$ / new PRF key
30 $k \xleftarrow{\$} \mathsf{Eval}(\ell, k_2 || pk_s || pk_r || c)$ / call Eval query
31 $\mathcal{E}_1 := \mathcal{E}_1 \cup \{(\mu(sk^{(1)}), pk^{(1)}, c_1, \ell)\}$
32 **if** $b = 1$
33 $\quad k \xleftarrow{\$} \mathcal{K}$
34 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), pk_r, c, k)\}$
35 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mu(sk), pk_r, c, k)\}$
36 **return** (c, k)

Figure 20: 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 2.

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_{Chl} instances and at most $Q_{\mathsf{Dec}} + Q_{\mathsf{Chl}}$ 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 9: There exists an adversary $\mathcal{B}_2$ against the **Ins-CCA** security of AKEM_2, such that

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

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

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

Claim 10: There exists an adversary $\mathcal{C}_2$ against the **PRF** security of H_2, i.e. keyed on the first input, such that

$$\left|\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right]\right| \le \mathrm{Adv}_{\mathsf{H}_2,\mathcal{C}_2}^{(Q_{\mathsf{Chl}},Q_{\mathsf{Dec}}+Q_{\mathsf{Chl}})\text{-}\mathbf{PRF}}.$$

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

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \Pr[G_5^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

■

Theorem 3 (Authenticity)**.** *For any* **Out-Aut** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$*, as depicted in Figure 5, 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* **mPRF** *adversary* $\mathcal{D}$ *against* H *such that*

$$\begin{aligned}
&\mathrm{Adv}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathcal{A}}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}} \le \\
&\quad \min\Big\{\mathrm{Adv}_{\mathsf{AKEM}_1,\mathcal{B}_1}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}} + \mathrm{Adv}_{\mathsf{AKEM}_1,\mathcal{C}_1}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}CCA}}, \\
&\qquad \mathrm{Adv}_{\mathsf{AKEM}_2,\mathcal{B}_2}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}} + \mathrm{Adv}_{\mathsf{AKEM}_2,\mathcal{C}_2}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}CCA}}\Big\} \\
&\quad + \mathrm{Adv}_{\mathsf{H},\mathcal{D}}^{(Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}},Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}})\text{-}\mathbf{mPRF}} + Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}]}.
\end{aligned}$$

Proof. Consider the sequence of games depicted in Figure 21.

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

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathcal{A}}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}}.$$

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:

$$\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right]\right| \le Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}]}.$$

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

$$\left|\Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right]\right| \le \mathrm{Adv}_{\mathsf{AKEM}_1,\mathcal{B}_1}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}}.$$

$G_0 - G_4$

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

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

10 **parse** $sk_s \rightarrow (sk^{(1)}, sk^{(2)})$
11 **parse** $pk \rightarrow (pk^{(1)}, pk^{(2)})$
12 $(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk^{(1)}, pk^{(1)})$
13 **if** $pk^{(1)} \in \{pk_1^{(1)}, \ldots, pk_n^{(1)}\}$ / $G_3 - G_4$
14 $\quad k_1 \xleftarrow{\$} \mathcal{K}_1$ / $G_3 - G_4$
15 $\mathcal{E}_1 := \mathcal{E}_1 \cup \{(\mu(sk^{(1)}), pk^{(1)}, c_1, k_1)\}$ / $G_2 - G_4$
16 $(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
17 $c := (c_1, c_2)$
18 $k := \mathsf{H}(k_1, k_2, pk_s, pk, c)$
19 **if** $pk^{(1)} \in \{pk_1^{(1)}, \ldots, pk_n^{(1)}\}$ / G_4
20 $\quad k \xleftarrow{\$} \mathcal{K}$ / G_4
21 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
22 **return** (c, k)

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

23 **if** $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
24 $\quad$ **return** k
25 **parse** $pk \rightarrow (pk^{(1)}, pk^{(2)})$
26 **parse** $sk_r \rightarrow (sk^{(1)}, sk^{(2)})$
27 **parse** $c \rightarrow (c_1, c_2)$
28 $k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk^{(1)}, sk^{(1)}, c_1)$
29 $k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk^{(2)}, sk^{(2)}, c_2)$
30 **if** $\exists\, k_1' : (pk^{(1)}, \mu(sk^{(1)}), c_1, k_1') \in \mathcal{E}_1$
31 $\quad k_1 := k_1'$
32 **elseif** $(pk^{(1)}, \cdot) \in \{pk_1, \ldots, pk_n\} \land k_1 \neq \perp$ / $G_2 - G_4$
33 $\quad k_1 \xleftarrow{\$} \mathcal{K}_1$ / $G_2 - G_4$
34 $\quad \mathcal{E}_1 := \mathcal{E}_1 \cup \{(pk^{(1)}, \mu(sk^{(1)}), c_1, k_1)\}$ / $G_2 - G_4$
35 $k := \mathsf{H}(k_1, k_2, pk, pk_r, c)$
36 **if** $(pk^{(1)}, \cdot) \in \{pk_1, \ldots, pk_n\} \land k_1 \neq \perp$ / G_4
37 $\quad k \xleftarrow{\$} \mathcal{K}$ / G_4
38 **if** $b = 1 \land pk \in \{pk_1, \ldots, pk_n\} \land k \neq \perp$
39 $\quad k \xleftarrow{\$} \mathcal{K}$
40 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
41 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$ / $G_1 - G_4$
42 **return** k

Figure 21: Games for the proof of Theorem 3.

Proof. Adversary $\mathcal{B}_1$ is formally constructed in Figure 22. 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 $\mathsf{Encps}_{\mathcal{B}}$ and $\mathtt{Chall}_{\mathcal{B}}$ is the same as for adversary $\mathcal{A}$.

■

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

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

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

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

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

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

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

Figure 22: Adversary $\mathcal{B}_1$ against Out-Aut security of AKEM_2, having access to oracles $\mathsf{Encps}_{\mathcal{B}}$ and $\mathsf{Chall}_{\mathcal{B}}$, simulating Game $\mathrm{G}_1/\mathrm{G}_2$ for adversary $\mathcal{A}$ from the proof of Theorem 3.

Claim 12: There exists an adversary C_1 against the **Out-CCA** security of AKEM_1, such that

$$\left|\Pr\left[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{AKEM}_1, C_1}^{(n, \mathrm{Q}_{\mathsf{Enc}}, \mathrm{Q}_{\mathsf{Chl}})\text{-}\textbf{Out-CCA}}.$$

Proof. Adversary C_1 is constructed in Figure 23. In G_2, the encapsulation is the real encapsulation of AKEM_1, thus querying the oracle Encps_C simulates Game G_2 for adversary $\mathcal{A}$. In the **Out-CCA** case $b = 1$, the encapsulation oracle Encps_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 C equals exactly the ones of $\mathcal{A}$. ∎

Game G_4. Game G_4 is the same as G_2 except that the output of hash 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 $\bot$. Further, the output of hash function H is also replaced by a random value in the encapsulation oracle if the receiver is honest.

$C^{\mathsf{Encps}_C,\mathsf{Decps}_C}(pk_1^{(1)},\dots,pk_n^{(1)})$

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

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

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

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

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

Figure 23: Adversary C_1 against Out-CCA security of AKEM_2, having access to oracles $\mathsf{Encps}_{\mathcal{B}}$ and $\mathsf{Decps}_{\mathcal{B}}$, simulating Game $\mathrm{G}_2/\mathrm{G}_3$ for adversary $\mathcal{A}$ from the proof of Theorem 3.

Claim 13: There exists an adversary $\mathcal{D}_1$ against the **PRF** security of H_1, such that

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

Proof. Adversary $\mathcal{D}_1$ is formally constructed in Figure 24.

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 2, we obtain a perfect simulation. The maximal number of different PRF keys is the same as maximal evaluation queries and amounts to $Q_{\mathsf{Enc}} + Q_{\mathsf{Chl}}$.

```
D_1^Eval
ℓ := 0
D, E_1, E_2 := ∅
for i ∈ [n]
    (sk^(1), pk^(1)) <-$ AKEM_1.Gen
    (sk^(2), pk^(2)) <-$ AKEM_2.Gen
    sk_i := (sk^(1), sk^(2))
    pk_i := (pk^(1), pk^(2))
b <-$ {0,1}
b' <-$ A^{Encps,Chall}(pk_1, ..., pk_n)
return [[b = b']]

Oracle Encps(s ∈ [n], pk)
parse sk_s → (sk^(1), sk^(2))
parse pk → (pk^(1), pk^(2))
(c_1, k_1) <-$ AKEM_1.Enc(sk^(1), pk^(1))
if pk^(1) ∈ {pk_1^(1), ..., pk_n^(1)}
    ℓ := ℓ + 1                                          / new key
    E_1 := E_1 ∪ {(μ(sk^(1)), pk^(1), c_1, ℓ)}          / store key
(c_2, k_2) <-$ AKEM_2.Enc(sk^(2), pk^(2))
c := (c_1, c_2)
k := H(k_1, k_2, pk_s, pk, c)
if pk^(1) ∈ {pk_1^(1), ..., pk_n^(1)}
    k_1 <-$ Eval(ℓ, k_2||pk_s||pk||c)                   / eval query
D ← D ∪ {(pk_s, pk, c, k)}
return (c, k)

Oracle Chall(pk, r ∈ [n], c)
if ∃ k : (pk, pk_r, c, k) ∈ D
    return k
parse pk → (pk^(1), pk^(2))
parse sk_r → (sk^(1), sk^(2))
parse c → (c_1, c_2)
k_1 ← AKEM_1.Dec(pk^(1), sk^(1), c_1)
k_2 ← AKEM_2.Dec(pk^(2), sk^(2), c_2)
if ∃ ℓ' : (pk^(1), μ(sk^(1)), c_1, ℓ') ∈ E_1
    k <-$ Eval(ℓ', k_2||pk||pk_r||c)                    / eval query
elseif (pk^(1), ·) ∈ {pk_1, ..., pk_n} ∧ k_1 ≠ ⊥
    ℓ := ℓ + 1                                          / new key
    E_1 := E_1 ∪ {(pk^(1), μ(sk^(1)), c_1, ℓ)}          / store key
    k <-$ Eval(ℓ, k_2||pk||pk_r||c)                     / eval query
else
    k := H(k_1, k_2, pk, pk_r, c)
if b = 1 ∧ pk ∈ {pk_1, ..., pk_n} ∧ k ≠ ⊥
    k <-$ K
D ← D ∪ {(pk, pk_r, c, k)}
return k
```

Figure 24: Adversary $\mathcal{D}_1$ against PRF security of H_1, having access to oracle Eval, simulating Game $\mathsf{G}_3/\mathsf{G}_4$ for adversary $\mathcal{A}$ from the proof of Theorem 3.

■

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 $pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \perp$ 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 21 to sustain readability.

Claim 14: There exist an adversaries $\mathcal{B}_2$ against the **Out-Aut** security of AKEM_2, C_2 against the **Out-CCA** security of AKEM_2, and $\mathcal{D}_2$ against the **PRF** security of H_2, such that

$$\left|\Pr\left[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{AKEM}_2,\mathcal{B}_2}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\textbf{Out-Aut}} + \mathrm{Adv}_{\mathsf{AKEM}_2,C_2}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\textbf{Out-CCA}} + \mathrm{Adv}_{\mathsf{H}_2,\mathcal{D}_2}^{(Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}},Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}})\text{-}\textbf{PRF}}.$$

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 4 (Dishonest Deniability)**.** *For all* PPT *simulators* $\mathsf{Sim}_1, \mathsf{Sim}_2$ *there exists a* PPT *simulator* $\mathsf{Sim}[\mathsf{Sim}_1, \mathsf{Sim}_2]$ *such that for any* **DR-Den** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$, *as depicted in Figure 5, there exists a* **DR-Den** *adversary* $\mathcal{B}_1$ *against* AKEM_1 *and a* **DR-Den** *adversary* $\mathcal{B}_2$ *against* AKEM_2 *such that*

$$\mathrm{Adv}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathsf{Sim},\mathcal{A}}^{(n,Q_{\mathsf{Chl}})\text{-}\textbf{DR-Den}} \leq \mathrm{Adv}_{\mathsf{AKEM}_1,\mathsf{Sim}_1,\mathcal{B}_1}^{(n,Q_{\mathsf{Chl}})\text{-}\textbf{DR-Den}} + \mathrm{Adv}_{\mathsf{AKEM}_2,\mathsf{Sim}_2,\mathcal{B}_2}^{(n,Q_{\mathsf{Chl}})\text{-}\textbf{DR-Den}}.$$

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

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 25. By definition, it holds

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

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.

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

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

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. ■

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

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

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

$\quad$ **parse** $pk_s \to (pk_s^{(1)}, pk_s^{(2)})$
$\quad$ **parse** $pk_r \to (pk_r^{(1)}, pk_r^{(2)})$
$\quad$ **parse** $sk_r \to (sk_r^{(1)}, sk_r^{(2)})$
$\quad (c_1, k_1) \xleftarrow{\$} \mathsf{Sim}_1(pk_s^{(1)}, pk_r^{(1)}, sk_r^{(1)})$
$\quad (c_2, k_2) \xleftarrow{\$} \mathsf{Sim}_2(pk_s^{(2)}, pk_r^{(2)}, sk_r^{(2)})$
$\quad c := (c_1, c_2)$
$\quad k := \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
$\quad$ **return** (c, k)

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

$\quad$ **if** $s = r$ **return** $\perp$
$\quad$ **parse** $sk_s \to (sk^{(1)}, sk^{(2)})$
$\quad$ **parse** $pk_r \to (pk^{(1)}, pk^{(2)})$
$\quad$ **parse** $pk_s \to (pk_s^{(1)}, pk_s^{(2)})$
$\quad$ **parse** $sk_r \to (sk_r^{(1)}, sk_r^{(2)})$
$\quad (c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk^{(1)}, pk^{(1)})$
$\quad (c_1, k_1) \xleftarrow{\$} \mathsf{Sim}_1(pk_s^{(1)}, pk^{(1)}, sk_r^{(1)})$ $\quad$ / $\mathsf{G}_0 - \mathsf{G}_1$
$\quad (c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk^{(2)}, pk^{(2)})$
$\quad (c_2, k_2) \xleftarrow{\$} \mathsf{Sim}_2(pk_s^{(2)}, pk^{(2)}, sk_r^{(2)})$ $\quad$ / $\mathsf{G}_0 - \mathsf{G}_2$
$\quad c := (c_1, c_2)$
$\quad k := \mathsf{H}(k_1, k_2, (\mu(sk^{(1)}), \mu(sk^{(2)})), pk_r, c)$
$\quad$ **if** $b = 1$
$\qquad (c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r, sk_r)$
$\quad$ **return** (c, k)

Figure 25: Games for the proof of Theorem 4 and definition of simulator $\mathsf{Sim} = \mathsf{Sim}[\mathsf{Sim}_1, \mathsf{Sim}_2]$**.**

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

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

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

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

■

Theorem 11 (Honest Deniability)**.** *For all* PPT *simulators* $\mathsf{Sim}_1, \mathsf{Sim}_2$ *there exists a* PPT *simulator* $\mathsf{Sim}[\mathsf{Sim}_1, \mathsf{Sim}_2]$ *such that for any* **HR-Den** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$, *as depicted in Figure 5, there exists a* **HR-Den** *adversary* $\mathcal{B}_1$ *against* AKEM_1 *and a* **HR-Den** *adversary* $\mathcal{B}_2$ *against* AKEM_2 *such that*

$$\mathrm{Adv}_{\mathsf{AKEM}[\mathsf{AKEM}_1,\mathsf{AKEM}_2,\mathsf{H}],\mathsf{Sim},\mathcal{A}}^{(n,Q_{\mathsf{Chl}})\textbf{-HR-Den}} \leq \mathrm{Adv}_{\mathsf{AKEM}_1,\mathsf{Sim}_1,\mathcal{B}_1}^{(n,Q_{\mathsf{Chl}})\textbf{-HR-Den}} + \mathrm{Adv}_{\mathsf{AKEM}_2,\mathsf{Sim}_2,\mathcal{B}_2}^{(n,Q_{\mathsf{Chl}})\textbf{-HR-Den}}.$$

Proof. The theorem can be proved analogously to Theorem 4. ■

C PROOFS FOR SECTION 5 (CONCRETE CONSTRUCTION)

Theorem 6 (Confidentiality). *For any* **Ins-CCA** *adversary* $\mathcal{A}$ *against* AKEM[NIKE, KEM, RSig, SE, H_1, H_2], *as depicted in Figure 6, there exists an* **CKS** *adversary* $\mathcal{B}$ *against* NIKE, *a* **PRF** *adversary* C *against* H_1, *an* **mPRF** *adversary* $\mathcal{D}$ *against* H_2, *and an* **IND-CCA** *adversary* $\mathcal{E}$ *against* KEM *such that*

$$\begin{aligned}\mathrm{Adv}^{(n,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},Q_{\mathsf{Chl}})\textbf{-Ins-CCA}}_{\mathsf{AKEM[NIKE,KEM,RSig,SE,H_1,H_2]},\mathcal{A}} &\leq nQ_{\mathsf{Chl}} \\ \cdot \Big(\min\Big\{&\mathrm{Adv}^{(Q_{\mathsf{Enc}}+2,2Q_{\mathsf{Enc}}+2Q_{\mathsf{Dec}},2Q_{\mathsf{Enc}}+2Q_{\mathsf{Enc}}+1)\textbf{-CKS}}_{\mathsf{NIKE},\mathcal{B}} \\ &+ \mathrm{Adv}^{(1,1)\textbf{-PRF}}_{\mathsf{H_1},C}, \mathrm{Adv}^{(1,Q_{\mathsf{Dec}},1)\textbf{-IND-CCA}}_{\mathsf{KEM},\mathcal{E}}\Big\} \\ &+ \mathrm{Adv}^{(1,Q_{\mathsf{Dec}}+1)\textbf{-mPRF}}_{\mathsf{H_2},\mathcal{D}} + (Q_{\mathsf{Enc}}+Q_{\mathsf{Dec}})\cdot\eta_{\mathsf{NIKE}}\cdot\gamma_{\mathsf{KEM}} \\ &+ Q_{\mathsf{Chl}}\cdot\delta_{\mathsf{AKEM[NIKE,KEM,RSig,SE,H_1,H_2]}}\Big).\end{aligned}$$

Proof. Consider the sequence of games depicted in Figure 26.

Game G_0. We start with the **Ins-CCA**$_{\mathsf{AKEM}}(\mathcal{A})$ game for AKEM[NIKE, KEM, RSig, SE, H_1, 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 $Q_{\mathsf{Enc}} + 2$ keys: one for the challenge user key, $npk^\star$, one for the ephemeral key in the challenge, $npk_e^\star$, and Q_{Enc} many ephemeral keys to answer the encapsulation queries. By definition it holds

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(1,Q_{\mathsf{Enc}}Q_{\mathsf{Dec}},1)\textbf{-Ins-CCA})}_{\mathsf{AKEM[NIKE,KEM,RSig,SE,H_1,H_2]},\mathcal{A}}.$$

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. Additionally, all inputs to hash function H_2 are stored together with their output in set $\mathcal{H}$. If the scheme is correct, these changes are indistinguishable

$$\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{AKEM[NIKE,KEM,RSig,SE,H_1,H_2]}}.$$

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 hash set $\mathcal{H}$ with the same inputs.

Claim 17:

$$\left|\Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq (Q_{\mathsf{Enc}} + Q_{\mathsf{Dec}}) \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}}.$$

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

$$\left|\Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{(Q_{\mathsf{Enc}}+2,2Q_{\mathsf{Enc}}+2Q_{\mathsf{Dec}},2Q_{\mathsf{Enc}}+2Q_{\mathsf{Enc}}+1)\textbf{-CKS}}_{\mathsf{NIKE},\mathcal{B}}.$$

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 27. They obtain public keys $npk_1, \ldots, npk_{Q_{\mathsf{Enc}}+2}$ 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 $\mathcal{K}_{H_1}$ if the input NIKE public key, npk equals the ephemeral challenge key $npk_e^\star$.

Claim 19: There exists an adversary C against the **PRF** security of H_1 such that

$$\left|\Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{(1,1)\textbf{-PRF}}_{\mathsf{H_1},C}.$$

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 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. ∎

```
G_0 – G_7
𝒟, ℋ := ∅
kct*, kk* := ⊥
for ℓ ∈ [Q_Enc + 2]
    (nsk_ℓ, npk_ℓ) ←$ NIKE.Gen
(nsk*, npk*) := (nsk_1, npk_1)
(nsk*_e, npk*_e) := (nsk_2, npk_2)
for i ∈ [Q_Enc + 2]                                   / G_3 – G_5
    k_ii := ⊥                                         / G_3 – G_5
    for j ∈ [i + 1, Q_Enc + 2]                        / G_3 – G_5
        k_ij := k_ji ←$ 𝒦_NIKE                        / G_3 – G_5
k* := k_12                                            / G_3 – G_5
k_H1 ←$ 𝒦_H1                                          / G_4 – G_5
ℓ := 2
(ksk*, kpk*) ←$ KEM.Gen
(ssk*, spk*) ←$ RSig.Gen
(sk*, pk*) := ((nsk*, ksk*, ssk*), (npk*, kpk*, spk*))
b ←$ {0, 1}
b' ← 𝒜^{Encps,Decps,Chall}(pk*)
return ⟦b = b'⟧

Oracle Decps(pk, c)
if ∃ k : (pk, c, k) ∈ 𝒟
    return k
parse pk → (npk, kpk, spk)
parse c → (npk_e, kct, sct)
nk' ← NIKE.Sdk(nsk*, npk)
if npk = npk*_e                                       / G_3 – G_5
    nk' := k*                                         / G_3 – G_5
elseif ∃ i : npk = npk_i                              / G_3 – G_5
    nk' := k_1i                                       / G_3 – G_5
nk := H_1(k'_1, "auth")
if npk = npk*_e                                       / G_4 – G_5
    nk := k_H1                                        / G_4 – G_5
nk_1‖nk_2 ← NIKE.Sdk(nsk*, npk_e)
if npk_e = npk*_e                                     / G_3 – G_5
    nk_1‖nk_2 := k*                                   / G_3 – G_5
elseif ∃ i : npk_e = npk_i                            / G_3 – G_5
    nk_1‖nk_2 := k_1i                                 / G_3 – G_5
kk_1‖kk_2 ← KEM.Dec(ksk*, kct)
if kct = kct*                                         / G_6 – G_7
    kk_1‖kk_2 := kk*                                  / G_6 – G_7
k' := H_1(nk_1, kk_1)
σ := SE.Dec(k', sct)
m ← (kct, kpk*)
if RSig.Ver(σ, ρ = {spk, spk*}, m) ≠ 1
    return ⊥
k := H_2(nk, nk_2, kk_2, c, pk, pk*)
if ∃ k' : (k', nk, nk_2, kk_2, c, pk, pk*) ∈ ℋ        / G_5, G_7
    k := k'                                           / G_5, G_7
elseif npk_e = npk*_e                                 / G_5
    k ←$ 𝒦                                            / G_5
elseif kct = kct*                                     / G_7
    k ←$ 𝒦                                            / G_7
ℋ := ℋ ∪ {(k, nk, nk_2, kk_2, c, pk, pk*)}            / G_1 – G_7
return k

Oracle Encps(pk)
ℓ := ℓ + 1
parse pk → (npk, kpk, spk)
(nsk_e, npk_e) := (nsk_ℓ, npk_ℓ)
nk' ← NIKE.Sdk(nsk*, npk)
nk_1‖nk_2 ← NIKE.Sdk(nsk_e, npk)
if npk = npk*_e                                       / G_3 – G_5
    nk' := k*                                         / G_3 – G_5
    nk_1‖nk_2 := k_ℓ2                                 / G_3 – G_5
elseif ∃ i : npk = npk_i                              / G_3 – G_5
    nk' := k_1i                                       / G_3 – G_5
    nk_1‖nk_2 := k_ℓi                                 / G_3 – G_5
nk := H_1(k'_1, "auth")
if npk = npk*_e                                       / G_4 – G_5
    nk := k_H1                                        / G_4 – G_5
(kct, kk_1‖kk_2) ←$ KEM.Enc(kpk)
m ← (kct, kpk)
σ ← RSig.Sgn(ssk*, {spk*, spk}, m)
k' := H_1(nk_1, kk_1)
sct := SE.Enc(k', σ)
c := (npk_e, kct, sct)
k := H_2(nk, nk_2, kk_2, c, pk*, pk)
ℋ := ℋ ∪ {(k, nk, nk_2, kk_2, c, pk*, pk)}            / G_1 – G_7
return (c, k)

Oracle Chall(sk)                                      / one query
parse sk → (nsk, ksk, ssk)
(nsk_e, npk_e) := (nsk*_e, npk*_e)
nk' ← NIKE.Sdk(nsk, npk*)
nk := H_1(k'_1, "auth")
nk_1‖nk_2 ← NIKE.Sdk(nsk_e, npk*)
nk_1‖nk_2 := k*                                       / G_3 – G_5
(kct, kk_1‖kk_2) ←$ KEM.Enc(kpk*)
kk_1‖kk_2 ←$ 𝒦_KEM                                    / G_6 – G_7
(kct*, kk*) := (kct, kk_1‖kk_2)
m ← (kct, kpk*)
σ ← RSig.Sgn(ssk, {μ(ssk), spk*}, m)
k' := H_1(nk_1, kk_2)
sct := SE.Enc(k', σ)
c := (npk_e, kct, sct)
if ∃ k' : (k', nk, nk_2, kk_2, c, μ(sk), pk*) ∈ ℋ     / G_2 – G_7
    abort                                             / G_2 – G_7
k := H_2(nk, nk_2, kk_2, c, μ(sk), pk*)
k ←$ 𝒦                                                / G_5, G_7
ℋ := ℋ ∪ {(k, nk, nk_2, kk_2, c, μ(sk), pk*)}         / G_1 – G_7
if b = 1
    k ←$ 𝒦
    𝒟 ← 𝒟 ∪ {(μ(sk), c, k)}
𝒟 ← 𝒟 ∪ {(μ(sk), c, k)}                               / G_1 – G_7
return (c, k)
```

Figure 26: Games $G_0 - G_7$ for the proof of Theorem 6.

Game G_5. Game G_5 is the same as G_4 except that the output of hash 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 $npk_e = npk_e^\star$.

```
𝓑^{RevCor,Test}(npk_1, ..., npk_{Q_Enc+2})
  𝓓 := ∅
  (ksk*, kpk*) ←$ KEM.Gen
  (ssk*, spk*) ←$ RSig.Gen
  sk* := (⊥, ksk*, ssk*)
  pk* := (npk_1, kpk*, spk*)                       / use first key for user key
  npk* := npk_1
  npk_e* := npk_2                 / use second key for ephemeral challenge key
  ℓ := 2
  b ←$ {0, 1}
  b' ← 𝒜^{Encps,Decps,Chall}(pk*)
  return ⟦b = b'⟧

Oracle Encps(pk)
  ℓ := ℓ + 1
  parse pk → (npk, kpk, spk)
  npk_e := npk_ℓ                                   / use next honest key
  if npk = npk_e*
    nk' ← Test(1, 2)                               / Test query
    nk_1‖nk_2 ← Test(ℓ, 2)                         / Test query
  elseif ∃ i : npk = npk_i
    nk' ← Test(1, i)
    nk_1‖nk_2 ← Test(ℓ, i)
  else
    nk' ←$ RevCor(1, npk)                          / RevCor query
    nk_1‖nk_2 ←$ RevCor(ℓ, npk)                    / RevCor query
  nk := H_1(nk', "auth")
  (kct, kk_1‖kk_2) ←$ KEM.Enc(kpk)
  m ← (kct, kpk)
  σ ← RSig.Sgn(ssk*, {spk*, spk}, m)
  k' := H_1(nk_1, kk_1)
  sct := SE.Enc(k', σ)
  c := (npk_e, kct, sct)
  k := H_2(nk, nk_2, kk_2, c, pk*, pk)
  return (c, k)

Oracle Decps(pk, c)
  if ∃ k : (pk, c, k) ∈ 𝓓
    return k
  parse pk → (npk, kpk, spk)
  parse c → (npk_e, kct, sct)
  if npk = npk_e*
    nk' ← Test(1, 2)                               / Test query
  elseif ∃ i : npk = npk_i
    nk' ← Test(1, i)                               / Test query
  else
    nk' ← RevCor(1, npk)                           / RevCor query
  nk := H_1(k'_1, "auth")
  if npk_e = npk_e*
    nk_1‖nk_2 ← Test(1, 2)                         / Test query
  elseif ∃ i : npk_e = npk_i
    nk_1‖nk_2 ← Test(1, i)                         / Test query
  else
    nk_1‖nk_2 ← RevCor(1, npk_e)                   / RevCor query
  kk_1‖kk_2 ← KEM.Dec(ksk*, kct)
  k' := H_1(nk_1, kk_1)
  σ := SE.Dec(k', sct)
  m ← (kct, kpk*)
  if RSig.Ver(σ, ρ = {spk, spk*}, m) ≠ 1
    return ⊥
  k := H_2(nk, nk_2, kk_2, c, pk, pk*)
  return k

Oracle Chall(sk)                                   / one query
  parse sk → (nsk, ksk, ssk)
  npk_e := npk_e*                                  / use second honest key
  nk' ← NIKE.Sdk(nsk, npk*)
  nk := H_1(nk', "auth")
  nk_1‖nk_2 ←$ Test(2, 1)             / Test query for (npk_e*, npk*)
  (kct, kk_1‖kk_2) ←$ KEM.Enc(kpk*)
  m ← (kct, kpk*)
  σ ← RSig.Sgn(ssk, {μ(ssk), spk*}, m)
  k' := H_1(nk_1, kk_1)
  sct := SE.Enc(k', σ)
  c := (npk_e, kct, sct)
  k := H_2(nk, nk_2, kk_2, c, μ(sk), pk*)
  if b = 1
    k ←$ 𝒦
  𝓓 ← 𝓓 ∪ {(μ(sk), c, k)}
  return (c, k)
```

Figure 27: Adversary $\mathcal{B}$ against CKS security of NIKE, having access to oracles RevCor and Test, simulating Game G_2/G_3 for adversary $\mathcal{A}$ from the proof of Theorem 6.

Claim 20: There exists an adversary $\mathcal{D}_1$ against the **mPRF** security of H_2 such that

$$\left|\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{H_2,\mathcal{D}_1}^{(1,Q_{\mathsf{Dec}}+1)\text{-}\mathbf{PRF}}.$$

Proof. Adversary $\mathcal{D}_1$ is constructed in Figure 28 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 23 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 21: There exists an adversary $\mathcal{E}$ against the **IND-CCA** security of KEM such that

$$\left|\Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_6^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{KEM},\mathcal{E}}^{(1,Q_{\mathsf{Dec}},1)\text{-}\mathbf{IND\text{-}CCA}}.$$

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 hash function H_2 in the challenge oracle is replaced by a uniformly sampled element of the output space.

Claim 22: There exists an adversary $\mathcal{D}_2$ against the **mPRF** security of H_2 such that

$$\left|\Pr\left[G_6^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_7^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{H}_2,\mathcal{D}_2}^{(1,Q_{\mathsf{Dec}}+1)\text{-}\mathbf{mPRF}}.$$

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

$$\Pr[G_5^{\mathcal{A}} \Rightarrow 1] = \Pr[G_7^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Collecting the bounds for Games G_0 – G_5 and Games G_0 – G_2, G_6 – 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 7 (Authenticity)**.** *For any* **Out-Aut** *adversary* $\mathcal{A}$ *against* $\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2]$*, as depicted in Figure 6, there exists an* **CKS** *adversary* $\mathcal{B}$ *against* NIKE*, a* **PRF** *adversary* $\mathcal{C}$ *against* H_1*, an* **mPRF** *adversary* $\mathcal{D}$ *against* H_2*, a* **UF-CRA1** *adversary* $\mathcal{E}$ *against* RSig*, and an* **IND-CCA** *adversary* $\mathcal{F}$ *against* KEM*, such that*

$$\begin{aligned}
&\mathrm{Adv}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2],\mathcal{A}}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}} \\
&\quad\leq \min\Big\{\mathrm{Adv}_{\mathsf{NIKE},\mathcal{B}}^{(Q_{\mathsf{Enc}}+2Q_{\mathsf{Chl}},Q_{\mathsf{Enc}}+2Q_{\mathsf{Chl}})\text{-}\mathbf{CKS}} + \mathrm{Adv}_{\mathsf{H}_1,\mathcal{C}}^{(n^2,n^2)\text{-}\mathbf{PRF}}, \\
&\qquad \mathrm{Adv}_{\mathsf{RSig},\mathcal{E}}^{(n,2,Q_{\mathsf{Enc}})\text{-}\mathbf{UF\text{-}CRA1}} + \mathrm{Adv}_{\mathsf{KEM},\mathcal{F}}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{IND\text{-}CCA}} \\
&\qquad + Q_{\mathsf{Enc}}^2 \cdot \gamma_{\mathsf{KEM}}\Big\} \\
&\quad + \mathrm{Adv}_{\mathsf{H}_2,\mathcal{D}}^{(Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}},Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}})\text{-}\mathbf{mPRF}} \\
&\quad + Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2]} \\
&\quad + Q_{\mathsf{Enc}} \cdot (Q_{\mathsf{Enc}} + Q_{\mathsf{Chl}}) \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}}.
\end{aligned}$$

Proof. Consider the sequence of games depicted in Figure 29 and Figure 30.

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

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2],\mathcal{A}}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{Out\text{-}Aut}}.$$

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:

$$\begin{aligned}
&\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right]\right| \\
&\quad \leq Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{AKEM}[\mathsf{NIKE},\mathsf{KEM},\mathsf{RSig},\mathsf{SE},\mathsf{H}_1,\mathsf{H}_2]}.
\end{aligned}$$

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 hash set $\mathcal{H}$ with the same inputs.

Claim 23:

$$\begin{aligned}
&\left|\Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right]\right| \\
&\quad \leq Q_{\mathsf{Enc}} \cdot (Q_{\mathsf{Enc}} + Q_{\mathsf{Chl}}) \cdot \eta_{\mathsf{NIKE}} \cdot \gamma_{\mathsf{KEM}}.
\end{aligned}$$

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 `Encps` and `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 hash set $\mathcal{H}$ with the same inputs.

Claim 24:

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \Pr[G_3^{\mathcal{A}} \Rightarrow 1].$$

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

```
D^Eval
01 D, H := ∅
02 kct*, kk* := ⊥
03 for ℓ ∈ [Q_Enc + 2]
04   (nsk_ℓ, npk_ℓ) <-$ NIKE.Gen
05 (nsk*, npk*) := (nsk_1, npk_1)
06 (nsk*_e, npk*_e) := (nsk_2, npk_2)
07 for i ∈ [Q_Enc + 2]
08   k_ii := ⊥
09   for j ∈ [i + 1, Q_Enc + 2]
10     k_ij := k_ji <-$ K_NIKE
11 k* := k_12
12 k_H1 <-$ K_H1
13 ℓ := 2
14 (ksk*, kpk*) <-$ KEM.Gen
15 (ssk*, spk*) <-$ RSig.Gen
16 sk* := (nsk*, ksk*, ssk*)
17 pk* := (npk*, kpk*, spk*)
18 b <-$ {0, 1}
19 b' <- A^{Encps,Decps,Chall}(pk*)
20 return [[b = b']]

Oracle Encps(pk)
21 return G4.Encps(pk)

Oracle Decps(pk, c)
22 if ∃ k : (pk, c, k) ∈ D
23   return k
24 parse pk -> (npk, kpk, spk)
25 parse c -> (npk_e, kct, sct)
26 nk' <- NIKE.Sdk(nsk*, npk)
27 if npk = npk*_e
28   nk' := ⊥                                  / key unknown
29 elseif ∃ i : npk = npk_i
30   nk' := k_1i
31 nk := H1(nk', "auth")
32 if npk = npk*_e
33   nk := k_H1                                / key can be simulated
34 nk1 || nk2 <- NIKE.Sdk(nsk*, npk_e)
35 if npk_e = npk*_e
36   nk2 := ★                                  / key unknown
37 elseif ∃ i : npk_e = npk_i
38   nk1 || nk2 := k_1i
39 kk1 || kk2 <- KEM.Dec(ksk*, kct)
40 k' := H1(nk1, kk1)
41 σ := SE.Dec(k', sct)
42 m <- (kct, kpk*)
43 if RSig.Ver(σ, ρ = {spk, spk*}, m) ≠ 1
44   return ⊥
45 k := H2(nk, nk2, kk2, c, pk, pk*)
46 if npk_e = npk*_e
47   k <-$ Eval(1, nk || kk2 || c || pk || pk*)         / eval query
48 H := H ∪ {(k, nk, nk2, kk2, c, pk, pk*)}
49 return k

Oracle Chall(sk)                                / one query
50 parse sk -> (nsk, ksk, ssk)
51 (nsk_e, npk_e) := (nsk*_e, npk*_e)
52 nk' <- NIKE.Sdk(nsk, npk*)
53 nk := H1(nk', "auth")
54 nk1 || nk2 <- NIKE.Sdk(nsk_e, npk*)
55 nk2 := ★                                      / key unknown
56 (kct, kk1 || kk2) <-$ KEM.Enc(kpk*)
57 (kct*, kk*) := (kct, kk1 || kk2)
58 m <- (kct, kpk*)
59 σ <- RSig.Sgn(ssk, {μ(ssk), spk*}, m)
60 k' := H1(nk1, kk1)
61 sct := SE.Enc(k', σ)
62 c := (npk_e, kct, sct)
63 if ∃ k' : (k', nk, nk2, kk2, c, μ(sk), pk*) ∈ H
64   abort
65 k <-$ Eval(1, nk || kk2 || c || μ(sk) || pk*)         / eval query
66 H := H ∪ {(k, nk, nk2, kk2, c, μ(sk), pk*)}
67 if b = 1
68   k <-$ K
69   D <- D ∪ {(μ(sk), c, k)}
70 D <- D ∪ {(μ(sk), c, k)}
71 return (c, k)
```

Figure 28: Adversary $\mathcal{C}_1$ against PRF security of H_2 queried on the second input, having access to oracle Eval, simulating Game G_4/G_5 for adversary $\mathcal{A}$ from the proof of Theorem 6.

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

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

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 31. 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_{Chl}. ∎

Game G_5. This game is the same as G_4 except that the output of hash 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 26: There exists an adversary $\mathcal{C}$ against the **PRF** security of H_1 such that

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

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 hash 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 $Q_{\mathsf{Enc}} + Q_{\mathsf{Chl}}$ many keys and evaluation queries. However, there is at most one query per key since the input is

Games $G_0 - G_6$

01 $\mathcal{D}, \mathcal{D}_1, \mathcal{H}, \mathcal{H}' := \emptyset$
02 $\mathsf{BAD}_1 :=$ **false**
03 **for** $i \in [n]$
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 $\quad sk_i := (nsk_i, ksk_i, ssk_i)$
08 $\quad pk_i := (npk_i, kpk_i, spk_i)$
09 $b \xleftarrow{\$} \{0,1\}$
10 $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Encps},\mathsf{Chall}}(pk_1, \ldots, pk_n)$
11 **return** $[\![b = b']\!]$

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

12 **parse** $pk \rightarrow (npk, kpk, spk)$
13 $(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
14 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_s, npk)$
15 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk)$
16 $(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
17 $m \leftarrow (kct, kpk)$
18 $\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk\}, m)$
19 $k' := \mathsf{H}_1(nk_1, kk_1)$
20 $sct := \mathsf{SE.Enc}(k', \sigma)$
21 $c := (npk_e, kct, sct)$
22 **if** $\exists\, \hat{nk} : (\hat{nk}, \{npk_s, npk\}) \in \mathcal{D}_1$ $\quad$ / $G_4 - G_6$
23 $\quad nk' := \hat{nk}$ $\quad$ / $G_4 - G_6$
24 **elseif** $npk \in \{npk_1, \ldots, npk_n\}$ $\quad$ / $G_4 - G_6$
25 $\quad nk' \xleftarrow{\$} \mathcal{K}_{\mathsf{NIKE}}$ $\quad$ / $G_4 - G_6$
26 $\quad \mathcal{D}_1 := \mathcal{D}_1 \cup \{(nk', \{npk_s, npk\})\}$ $\quad$ / $G_4 - G_6$
27 $nk := \mathsf{H}_1(nk', \text{“auth”})$
28 **if** $\exists\, \hat{nk} : (\hat{nk}, \{npk_s, npk\}) \in \mathcal{H}'$ $\quad$ / $G_5 - G_6$
29 $\quad nk := \hat{nk}$ $\quad$ / $G_5 - G_6$
30 **elseif** $npk \in \{npk_1, \ldots, npk_n\}$ $\quad$ / $G_5 - G_6$
31 $\quad nk \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_1}$ $\quad$ / $G_5 - G_6$
32 $\quad \mathcal{H}' := \mathcal{H}' \cup \{(nk, \{npk_s, npk\})\}$ $\quad$ / $G_5 - G_6$
33 **if** $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk_s, pk) \in \mathcal{H}$ $\quad$ / $G_2 - G_6$
34 $\quad \mathsf{BAD}_1$; **abort** $\quad$ / $G_2 - G_6$
35 $k := \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk)$
36 **if** $npk \in \{npk_1, \ldots, npk_n\}$
37 $\quad k \xleftarrow{\$} \mathcal{K}$ $\quad$ / G_6
38 $\mathcal{H} := \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk_s, pk)\}$ $\quad$ / $G_1 - G_6$
39 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
40 **return** (c, k)

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

41 **if** $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
42 $\quad$ **return** k
43 **parse** $pk \rightarrow (npk, kpk, spk)$
44 **parse** $c \rightarrow (npk_e, kct, sct)$
45 $nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk)$
46 $nk := \mathsf{H}_1(k_1', \text{“auth”})$
47 $nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_e)$
48 **if** $\exists\, \hat{nk} : (\hat{nk}, \{npk_r, npk_e\}) \in \mathcal{D}_1$ $\quad$ / $G_4 - G_6$
49 $\quad nk_1 \| nk_2 := \hat{nk}$ $\quad$ / $G_4 - G_6$
50 **elseif** $npk_e \in \{npk_1, \ldots, npk_n\}$ $\quad$ / $G_4 - G_6$
51 $\quad nk_1 \| nk_2 \xleftarrow{\$} \mathcal{K}_{\mathsf{NIKE}}$ $\quad$ / $G_4 - G_6$
52 $kk_1 \| kk_2 \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
53 $k' := \mathsf{H}_1(nk_1, kk_1)$
54 $\sigma := \mathsf{SE.Dec}(k', sct)$
55 $m \leftarrow (kct, kpk_r)$
56 **if** $\mathsf{RSig.Ver}(\sigma, \{spk, spk_r\}, m) \neq 1$
57 $\quad$ **return** $\perp$
58 **if** $\exists\, \hat{nk} : (\hat{nk}, \{npk_s, npk\}) \in \mathcal{D}_1$ $\quad$ / $G_4 - G_6$
59 $\quad nk' := \hat{nk}$ $\quad$ / $G_4 - G_6$
60 **elseif** $npk \in \{npk_1, \ldots, npk_n\}$ $\quad$ / $G_4 - G_6$
61 $\quad nk' \xleftarrow{\$} \mathcal{K}_{\mathsf{NIKE}}$ $\quad$ / $G_4 - G_6$
62 $\quad \mathcal{D}_1 := \mathcal{D}_1 \cup \{(nk', \{npk, npk_r\})\}$ $\quad$ / $G_4 - G_6$
63 $nk := \mathsf{H}_1(nk', \text{“auth”})$
64 **if** $\exists\, \hat{nk} : (\hat{nk}, \{npk, npk_r\}) \in \mathcal{H}'$ $\quad$ / $G_5 - G_6$
65 $\quad nk := \hat{nk}$ $\quad$ / $G_5 - G_6$
66 **elseif** $npk \in \{npk_1, \ldots, npk_n\}$ $\quad$ / $G_5 - G_6$
67 $\quad nk \xleftarrow{\$} \mathcal{K}_{\mathsf{H}_1}$ $\quad$ / $G_5 - G_6$
68 $\quad \mathcal{H}' := \mathcal{H}' \cup \{(nk, \{npk, npk_r\})\}$ $\quad$ / $G_5 - G_6$
69 **if** $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk, pk_r) \in \mathcal{H}$ $\quad$ / $G_3 - G_6$
70 $\quad$ **abort** $\quad$ / $G_3 - G_6$
71 $k := \mathsf{H}_2(nk, nk_2, kk_2, c, pk, pk_r)$
72 **if** $npk \in \{npk_1, \ldots, npk_n\}$
73 $\quad k \xleftarrow{\$} \mathcal{K}$ $\quad$ / G_6
74 $\mathcal{H} := \mathcal{H} \cup \{(k, nk, nk_2, kk_2, c, pk, pk_r)\}$ $\quad$ / $G_1 - G_6$
75 **if** $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \perp$
76 $\quad k \xleftarrow{\$} \mathcal{K}$
77 $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
78 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$ $\quad$ / $G_1 - G_6$
79 **return** k

Figure 29: Games $G_0 - G_6$ for the proof of Theorem 7.

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 hash 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 27: There exists an adversary $\mathcal{D}_1$ against the **mPRF** security of H_2 such that

$$\left|\Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_6^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{H}_2,\mathcal{D}_1}^{(Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}}, Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}})\text{-}\mathbf{mPRF}}.$$

Proof. Adversary $\mathcal{D}_1$ is formally constructed in Figure 32 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. ■

Game G_7. This is the same as G_3 (note that this does not build upon the previous game) except that flag BAD_2 is set to **true** and

Games $G_3, G_7 - G_{10}$

01 $\mathcal{D}, \mathcal{D}_2 \mathcal{H}, \mathcal{Q} := \emptyset$
02 $\mathsf{BAD}_2, \mathsf{BAD}_3 :=$ **false**
03 **for** $i \in [n]$
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 $\quad sk_i := (nsk_i, ksk_i, ssk_i)$
08 $\quad pk_i := (npk_i, kpk_i, spk_i)$
09 $b \xleftarrow{\$} \{0, 1\}$
10 $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Encps},\mathsf{Chall}}(pk_1, \ldots, pk_n)$
11 **return** $[\![b = b']\!]$

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

12 **parse** $pk \rightarrow (npk, kpk, spk)$
13 $(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
14 $k_1' \leftarrow \mathsf{NIKE.Sdk}(nsk_s, npk)$
15 $k_1 := \mathsf{H}_1(k_1', \text{“auth”})$
16 $k_2' \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk)$
17 $(kct, kk) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk)$
18 **if** $kpk \in \{kpk_1, \ldots, kpk_n\}$ / $G_9 - G_{10}$
19 $\quad kk \xleftarrow{\$} \mathcal{K}_{\mathsf{KEM}}$ / $G_9 - G_{10}$
20 $\quad \mathcal{D}_2 := \mathcal{D}_2 \cup \{(kpk, kct, kk)\}$ / $G_9 - G_{10}$
21 $m \leftarrow (kct, kpk)$
22 **if** $(\{spk_s, spk\}, m, \cdot) \in \mathcal{Q}$ / $G_7 - G_{10}$
23 $\quad \mathsf{BAD}_2 :=$ **true**; **abort** / $G_7 - G_{10}$
24 $\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{\mu(ssk_s), spk\}, m)$
25 $\mathcal{Q} := \mathcal{Q} \cup \{(\{spk_s, spk\}, m, \sigma)\}$ / $G_7 - G_{10}$
26 $k' := \mathsf{H}_1(k_2, kk)$
27 $sct := \mathsf{SE.Enc}(k', \sigma)$
28 $c := (npk_e, kct, sct)$
29 **if** $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk, pk_r) \in \mathcal{H}$
30 $\quad$ **abort**
31 $k := \mathsf{H}_2(k_1, k_2, kk, c, pk_s, pk)$
32 **if** $kpk \in \{kpk_1, \ldots, kpk_n\}$ / G_{10}
33 $\quad k \xleftarrow{\$} \mathcal{K}$ / G_{10}
34 $\mathcal{H} := \mathcal{H} \cup \{(k, k_1, k_2, kk, c, pk_s, pk)\}$
35 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
36 **return** (c, k)

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

37 **if** $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
38 $\quad$ **return** k
39 **parse** $pk \rightarrow (npk, kpk, spk)$
40 **parse** $c \rightarrow (npk_e, kct, sct)$
41 $k_1' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk)$
42 $k_1 := \mathsf{H}_1(k_1', \text{“auth”})$
43 $k_2' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_e)$
44 $kk \leftarrow \mathsf{KEM.Dec}(ksk_r, kct)$
45 **if** $\exists\, kk' : (kpk_r, kct, kk') \in \mathcal{D}_2$ / $G_9 - G_{10}$
46 $\quad kk := kk'$ / $G_9 - G_{10}$
47 $k' := \mathsf{H}_1(k_2, kk)$
48 $\sigma := \mathsf{SE.Dec}(k', sct)$
49 $m \leftarrow (kct, kpk_r)$
50 $k := \mathsf{H}_2(k_1, k_2, kk, c, pk, pk_r)$
51 **if** $\mathsf{RSig.Ver}(\sigma, \{spk, spk_r\}, m) \neq 1$
52 $\quad$ **return** $\perp$
53 **elseif** $\exists\, i : spk = spk_i \wedge (\{spk, spk_r\}, m, \cdot) \notin \mathcal{Q}$ / $G_8 - G_{10}$
54 $\quad \mathsf{BAD}_3 :=$ **true**; **abort** / $G_8 - G_{10}$
55 **if** $\exists\, k : (k, \cdot, \cdot, \cdot, c, pk, pk_r) \in \mathcal{H}$
56 $\quad$ **abort**
57 **if** $spk \in \{spk_1, \ldots, spk_n\} \wedge k \neq \perp$ / G_{10}
58 $\quad k \xleftarrow{\$} \mathcal{K}$ / G_{10}
59 $\mathcal{H} := \mathcal{H} \cup \{(k, k_1, k_2, kk, c, pk, pk_r)\}$
60 **if** $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \perp$
61 $\quad k \xleftarrow{\$} \mathcal{K}$
62 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
63 **return** k

Figure 30: Games $G_3, G_7 - G_{10}$ for the proof of Theorem 7.

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 28:

$$\left|\Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_7^{\mathcal{A}} \Rightarrow 1\right]\right| \leq Q_{\mathsf{Enc}}^2 \cdot \gamma_{\mathsf{KEM}}.$$

Proof. The message being signed in the encapsulation oracle consists of several components where one of them is the KEM ciphertext kct. Hence, 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_{Enc} queries to the encapsulation oracle and at most the same number of elements in set $\mathcal{Q}$, it holds

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

∎

Game G_8. This game is the same as G_7 except that flag 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.

Claim 29: There exists an adversary $\mathcal{E}$ against the **UF-CRA1** security of RSig such that

$$\left|\Pr\left[G_7^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_8^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{RSig},\mathcal{E}}^{(n,2,Q_{\mathsf{Enc}})\text{-}\mathbf{UF\text{-}CRA1}}.$$

Proof. Adversary $\mathcal{E}$ is formally constructed in Figure 33. 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.

```
B^{RevCor,Test}(npk_1, ..., npk_n)
01 D, D_1, H := ∅
02 for i ∈ [n]
03    (ksk_i, kpk_i) <-$ KEM.Gen
04    (ssk_i, spk_i) <-$ RSig.Gen
05    sk_i := (⊥, ksk_i, ssk_i)
06    pk_i := (npk_i, kpk_i, spk_i)
07 b <-$ {0, 1}
08 b' <-$ A^{Encps,Chall}(pk_1, ..., pk_n)
09 return [[b = b']]
```

```
Oracle Encps(s ∈ [n], pk)
10 parse pk → (npk, kpk, spk)
11 (nsk_e, npk_e) <-$ NIKE.Gen
12 nk_1||nk_2 ← NIKE.Sdk(nsk_e, npk)
13 (kct, kk_1||kk_2) <-$ KEM.Enc(kpk)
14 m ← (kct, kpk)
15 σ ← RSig.Sgn(ssk_s, {μ(ssk_s), spk}, m)
16 k' := H_1(nk_1, kk_1)
17 sct := SE.Enc(k', σ)
18 c := (npk_e, kct, sct)
19 if ∃ n̂k : (n̂k, {npk_s, npk}) ∈ D_1
20    nk' := n̂k
21 elseif ∃ r : npk = npk_r
22    nk' <-$ Test(s, r)                    / test query
23    D_1 := D_1 ∪ {(nk', {npk_s, npk})}
24 else
25    nk' <-$ RevCor(s, npk)      / corrupt reveal query
26 nk := H_1(nk', "auth")
27 k := H_2(nk, nk_2, kk_2, c, pk_s, pk)
28 H := H ∪ {(k, nk, nk_2, kk_2, c, pk_s, pk)}
29 D ← D ∪ {(pk_s, pk, c, k)}
30 return (c, k)
```

```
Oracle Chall(pk, r ∈ [n], c)
31 if ∃ k : (pk, pk_r, c, k) ∈ D
32    return k
33 parse pk → (npk, kpk, spk)
34 parse c → (npk_e, kct, sct)
35 if ∃ i : npk_e = npk_i
36    nk_1||nk_2 <-$ Test(r, i)                / test query
37 else
38    nk_1||nk_2 <-$ RevCor(r, npk_e)    / corrupt reveal query
39 kk_1||kk_2 ← KEM.Dec(ksk_r, kct)
40 k' := H_1(nk_1, kk_1)
41 σ := SE.Dec(k', sct)
42 m ← (kct, kpk_r)
43 if RSig.Ver(σ, {spk, spk_r}, m) ≠ 1
44    return ⊥
45 if ∃ n̂k : (n̂k, {npk, npk_r}) ∈ D_1
46    nk' := n̂k
47 elseif ∃ s : npk = npk_s
48    nk' <-$ Test(r, s)                      / test query
49 else
50    nk' <-$ RevCor(r, pk)         / corrupt reveal query
51 nk := H_1(nk', "auth")
52 k := H_2(nk, nk_2, kk_2, c, pk, pk_r)
53 H := H ∪ {(k, nk, nk_2, kk_2, c, pk, pk_r)}
54 if b = 1 ∧ pk ∈ {pk_1, ..., pk_n} ∧ k ≠ ⊥
55    k <-$ K
56 D ← D ∪ {(pk, pk_r, c, k)}
57 return k
```

Figure 31: Adversary $\mathcal{B}$ against CKS security of NIKE, having access to oracles RevCor and Test, simulating Game G_3/G_4 for adversary $\mathcal{A}$ from the proof of Theorem 7.

■

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

$$\left|\Pr\left[G_8^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_9^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{\mathsf{KEM},\mathcal{F}}^{(n,Q_{\mathsf{Enc}},Q_{\mathsf{Chl}})\text{-}\mathbf{IND\text{-}CCA}}.$$

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 hash 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 31: There exists an adversary $\mathcal{D}_2$ against the **mPRF** security of H_2 such that

$$\left|\Pr\left[G_9^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_{10}^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}_{H_2,\mathcal{D}_2}^{(Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}},Q_{\mathsf{Enc}}+Q_{\mathsf{Chl}})\text{-}\mathbf{PRF}}.$$

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 BAD_3 to **true** and abort. This means that the sender verification is dishonest or there existing a matching element in 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 Q there must have been a corresponding query to Encps because 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

$\mathcal{D}_1^{\mathsf{Eval}}$

```
01 ℓ := 0
02 D, D_1, H, H' := ∅
03 for i ∈ [n]
04   (nsk_i, npk_i) ←$ NIKE.Gen
05   (ksk_i, kpk_i) ←$ KEM.Gen
06   (ssk_i, spk_i) ←$ RSig.Gen
07   sk_i := (nsk_i, ksk_i, ssk_i)
08   pk_i := (npk_i, kpk_i, spk_i)
09 b ←$ {0, 1}
10 b' ←$ A^{Encps,Chall}(pk_1, ..., pk_n)
11 return [[b = b']]
```

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

```
12 ℓ' := 0
13 parse pk → (npk, kpk, spk)
14 (nsk_e, npk_e) ←$ NIKE.Gen
15 nk' ← NIKE.Sdk(nsk_s, npk)
16 nk_1‖nk_2 ← NIKE.Sdk(nsk_e, npk)
17 (kct, kk_1‖kk_2) ←$ KEM.Enc(kpk)
18 m ← (kct, kpk)
19 σ ← RSig.Sgn(ssk_s, {spk_s, spk}, m)
20 k' := H_1(nk_1, kk_1)
21 sct := SE.Enc(k', σ)
22 c := (npk_e, kct, sct)
23 if ∃ n̂k : (n̂k, {npk_s, npk}) ∈ D_1
24   nk' := n̂k
25 elseif npk ∈ {npk_1, ..., npk_n}
26   nk' ←$ K_NIKE
27   D_1 := D_1 ∪ {(nk', {npk_s, npk})}
28 nk := H_1(nk', "auth")
29 if ∃ ℓ̂ : (ℓ̂, {npk_s, npk}) ∈ H'
30   ℓ := ℓ̂                                   / previous key
31 elseif npk ∈ {npk_1, ..., npk_n}
32   ℓ := ℓ + 1                               / new key
33   ℓ' := ℓ
34   H' := H' ∪ {(ℓ, {npk_s, npk})}
35 if ∃ k : (k, ·, ·, ·, c, pk_s, pk) ∈ H
36   BAD_1; abort
37 k := H_2(nk, nk_2, kk_2, c, pk_s, pk)
38 if npk ∈ {npk_1, ..., npk_n}
39   k ←$ Eval(ℓ', nk_2‖kk_2‖c‖pk_s‖pk)       / eval query
40 H := H ∪ {(k, nk, nk_2, kk_2, c, pk_s, pk)}
41 D ← D ∪ {(pk_s, pk, c, k)}
42 return (c, k)
```

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

```
43 ℓ' := 0
44 if ∃ k : (pk, pk_r, c, k) ∈ D
45   return k
46 parse pk → (npk, kpk, spk)
47 parse c → (npk_e, kct, sct)
48 nk' ← NIKE.Sdk(nsk_r, npk)
49 nk := H_1(nk', "auth")
50 nk_1‖nk_2 ← NIKE.Sdk(nsk_r, npk_e)
51 if ∃ n̂k : (n̂k, {npk_r, npk_e}) ∈ D_1
52   nk_1‖nk_2 := n̂k
53 elseif npk_e ∈ {npk_1, ..., npk_n}
54   nk_1‖nk_2 ←$ K_NIKE
55 kk_1‖kk_2 ← KEM.Dec(ksk_r, kct)
56 k' := H_1(nk_1, kk_1)
57 σ := SE.Dec(k', sct)
58 m ← (kct, kpk_r)
59 if RSig.Ver(σ, {spk, spk_r}, m) ≠ 1
60   return ⊥
61 if ∃ n̂k : (n̂k, {npk_s, npk}) ∈ D_1
62   nk' := n̂k
63 elseif npk ∈ {npk_1, ..., npk_n}
64   nk' ←$ K_NIKE
65   D_1 := D_1 ∪ {(nk', {npk, npk_r})}
66 nk := H_1(nk', "auth")
67 if ∃ ℓ̂ : (ℓ̂, {npk, npk_r}) ∈ H'
68   ℓ' := ℓ̂                                  / previous key
69 elseif npk ∈ {npk_1, ..., npk_n}
70   ℓ := ℓ + 1                               / new key
71   ℓ' := ℓ
72   H' := H' ∪ {(ℓ, {npk, npk_r})}
73 if ∃ k : (k, ·, ·, ·, c, pk, pk_r) ∈ H
74   abort
75 k := H_2(nk, nk_2, kk_2, c, pk, pk_r)
76 if npk ∈ {npk_1, ..., npk_n}
77   k ←$ Eval(ℓ', nk_2‖kk_2‖c‖pk‖pk_r)       / eval query
78 H := H ∪ {(k, nk, nk_2, kk_2, c, pk, pk_r)}
79 if b = 1 ∧ pk ∈ {pk_1, ..., pk_n} ∧ k ≠ ⊥
80   k ←$ K
81 D ← D ∪ {(pk, pk_r, c, k)}
82 return k
```

Figure 32: Adversary $\mathcal{D}_1$ against mPRF security of H_2, having access to oracle Eval, simulating Game $\mathsf{G}_5/\mathsf{G}_6$ for adversary $\mathcal{A}$ from the proof of Theorem 7.

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 32:

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

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

```
ℰ^Sgn(par, spk_1, ..., spk_n)
𝒟, ℋ, 𝒬 := ∅
for i ∈ [n]
    (nsk_i, npk_i) ←$ NIKE.Gen
    (ksk_i, kpk_i) ←$ KEM.Gen
    sk_i := (nsk_i, ksk_i, ⊥)
    pk_i := (npk_i, kpk_i, spk_i)
b ←$ {0, 1}
b' ←$ 𝒜^{Encps,Chall}(pk_1, ..., pk_n)
return ⟦b = b'⟧
```

```
Oracle Encps(s ∈ [n], pk)
parse pk → (npk, kpk, spk)
(nsk_e, npk_e) ←$ NIKE.Gen
nk' ← NIKE.Sdk(nsk_s, npk)
nk := H_1(nk', "auth")
nk_1‖nk_2 ← NIKE.Sdk(nsk_e, npk)
(kct, kk_1‖kk_2) ←$ KEM.Enc(kpk)
m ← (kct, kpk)
if ({spk_s, spk}, m, ·) ∈ 𝒬
    abort
σ ← Sgn(s, {spk_s, spk}, m)          / signing query
𝒬 := 𝒬 ∪ {({spk_s, spk}, m, σ)}
k' := H_1(nk_1, kk_1)
sct := SE.Enc(k', σ)
c := (npk_e, kct, sct)
if ∃ k : (k, ·, ·, ·, c, pk_s, pk) ∈ ℋ
    abort
k := H_2(nk, nk_2, kk_2, c, pk_s, pk)
ℋ := ℋ ∪ {(k, nk, nk_2, kk_2, c, pk_s, pk)}
𝒟 ← 𝒟 ∪ {(pk_s, pk, c, k)}
return (c, k)
```

```
Oracle Chall(pk, r ∈ [n], c)
if ∃ k : (pk, pk_r, c, k) ∈ 𝒟
    return k
parse pk → (npk, kpk, spk)
parse c → (npk_e, kct, sct)
nk' ← NIKE.Sdk(nsk_r, npk)
nk := H_1(nk', "auth")
nk_1‖nk_2 ← NIKE.Sdk(nsk_r, npk_e)
kk_1‖kk_2 ← KEM.Dec(ksk_r, kct)
k' := H_1(nk_1, kk_1)
σ := SE.Dec(k', sct)
m ← (kct, kpk_r)
k := H_2(nk, nk_2, kk_2, c, pk, pk_r)
if RSig.Ver(σ, {spk, spk_r}, m) ≠ 1
    return ⊥
elseif ∃ i : spk = spk_i ∧ ({spk, spk_r}, m, ·) ∉ 𝒬
    return (σ, {spk_i, spk_r}, m)          / return forgery
if ∃ k : (k, ·, ·, ·, c, pk, pk_r) ∈ ℋ
    abort
ℋ := ℋ ∪ {(k, nk, nk_2, kk_2, c, pk, pk_r)}
if b = 1 ∧ pk ∈ {pk_1, ..., pk_n} ∧ k ≠ ⊥
    k ←$ 𝒦
𝒟 ← 𝒟 ∪ {(pk, pk_r, c, k)}
return k
```

Figure 33: Adversary $\mathcal{E}$ against UF-CRA1 security of RSig, having access to oracle Sgn, simulating Game G_7/G_8 for adversary $\mathcal{A}$ from the proof of Theorem 7.

or the oracle outputs a uniformly random key if spk is honest and $k \neq \bot$ (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 8 (Dishonest Deniability). *There exists a simulator* Sim *such that for any* **DR-Den** *adversary* $\mathcal{A}$ *against* AKEM[NIKE, KEM, RSig, SE, H_1, H_2], *as depicted in Figure 6, there exists a* **MC-Ano** *adversary* $\mathcal{B}$ *against* RSig, *such that*

$$\mathrm{Adv}^{(n,Q_{\mathsf{Chl}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM[NIKE,KEM,RSig,SE,H_1,H_2],Sim},\mathcal{A}} \leq \mathrm{Adv}^{(n,2,Q_{\mathsf{Chl}})\text{-}\mathbf{MC\text{-}Ano}}_{\mathsf{RSig},\mathcal{B}} + Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{NIKE}}.$$

Proof. Consider the sequence of games depicted in Figure 34 as well as the construction of a simulator Sim.

Game G_0. We start with the dishonest receiver deniability game for AKEM[NIKE, KEM, RSig, SE, H_1, H_2]. Compared to the original definition in Figure 4, 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

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,Q_{\mathsf{Chl}})\text{-}\mathbf{DR\text{-}Den}}_{\mathsf{AKEM[NIKE,KEM,RSig,SE,H_1,H_2],Sim},\mathcal{A}}.$$

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 33:

$$\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq Q_{\mathsf{Chl}} \cdot \delta_{\mathsf{NIKE}}.$$

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

$$\left|\Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \mathrm{Adv}^{(n,2,Q_{\mathsf{Chl}})\text{-}\mathbf{MC\text{-}Ano}}_{\mathsf{RSig},\mathcal{B}}.$$

Proof. Adversary $\mathcal{B}$ is formally constructed in Figure 34. 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 $b = 1$:

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

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

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

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

$(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_s)$
$nk \coloneqq \mathsf{H}_1(nk', \text{“auth”})$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk_r)$
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$m \leftarrow (kct, kpk_r)$
$\sigma \xleftarrow{\$} \mathtt{Chl}_{\mathsf{RSig}}(s, r, \{spk_s, spk_r\}, m)$ / challenge query
$k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
$sct \coloneqq \mathsf{SE.Enc}(k', \sigma)$
$c \coloneqq (npk_e, kct, sct)$
$k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk_r)$
if $b = 1$
  $(c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r, sk_r)$
return (c, k)

Figure 35: Adversary $\mathcal{B}$ against MC-Ano security of RSig, having access to oracle $\mathtt{Chl}_{\mathsf{RSig}}$, simulating Game G_1/G_2 for adversary $\mathcal{A}$ from the proof of Theorem 8.

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

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

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

if $s = r$ **return** $\perp$
$(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_s, npk_r)$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_s)$ / $\mathsf{G}_1 - \mathsf{G}_2$
$nk \coloneqq \mathsf{H}_1(nk', \text{“auth”})$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk_r)$
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$m \leftarrow (kct, kpk_r)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_s, \{spk_s, spk_r\}, m)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_r, \{spk_s, spk_r\}, m)$ / G_2
$k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
$sct \coloneqq \mathsf{SE.Enc}(k', \sigma)$
$c \coloneqq (npk_e, kct, sct)$
$k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk_r)$
if $b = 1$
  $(c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r, sk_r)$
return (c, k)

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

parse $pk_s \to (npk_s, kpk_s, spk_s)$
parse $pk_r \to (npk_r, kpk_r, spk_r)$
parse $sk_r \to (nsk_r, ksk_r, ssk_r)$
$(nsk_e, npk_e) \xleftarrow{\$} \mathsf{NIKE.Gen}$
$nk' \leftarrow \mathsf{NIKE.Sdk}(nsk_r, npk_s)$
$nk \coloneqq \mathsf{H}_1(nk', \text{“auth”})$
$nk_1 \| nk_2 \leftarrow \mathsf{NIKE.Sdk}(nsk_e, npk_r)$
$(kct, kk_1 \| kk_2) \xleftarrow{\$} \mathsf{KEM.Enc}(kpk_r)$
$m \leftarrow (kct, kpk_r)$
$\sigma \leftarrow \mathsf{RSig.Sgn}(ssk_r, \{spk_s, spk_r\}, m)$
$k' \coloneqq \mathsf{H}_1(nk_1, kk_1)$
$sct \coloneqq \mathsf{SE.Enc}(k', \sigma)$
$c \coloneqq (npk_e, kct, sct)$
$k \coloneqq \mathsf{H}_2(nk, nk_2, kk_2, c, pk_s, pk_r)$
return (c, k)

Figure 34: Games $\mathsf{G}_0 - \mathsf{G}_2$ for the proof of Theorem 8.

■