DAKE: Bandwidth-Efficient (U)AKE from Double-KEM

Hugo Beguinet[1], Céline Chevalier[2,3], Guirec Lebrun[2,5], Thomas Legavre[1,4,5], Thomas Ricosset[1], Maxime Roméas[5], and Éric Sageloli[1,2]

[1] Thales, Gennevilliers, France
[2] DIENS, École normale supérieure, CNRS, PSL University, Inria, Paris, France
[3] CRED, Paris-Panthéon-Assas University
[4] Sorbonne Université, CNRS, LIP6, Paris, France
[5] ANSSI, Paris, France
thomas.legavre@lip6.fr
thomas.ricosset@thalesgroup.com
eric.sageloli@protonmail.com

Abstract. Bandwidth remains a major bottleneck in post-quantum cryptography, particularly for authenticated key exchange (AKE) protocols. In this work, we present DAKE, a bandwidth-efficient AKE framework built from double-KEM constructions. DAKE comes in two main versions achieving, respectively, weak and full perfect forward secrecy, as well as explicit authentication. It further admits two variants: a unilateral version, and another where a signature scheme replaces a KEM. They are proven secure in the standard model under eCK^{w} and eCK-PFS, two strong variants of the extended Canetti–Krawczyk framework.

DAKE employs a double-KEM, a primitive that encapsulates a single key under two public keys simultaneously. Such constructions can achieve smaller encapsulation sizes than two independent KEM encapsulations, offering a significant bandwidth advantage.

To facilitate the design of double-KEMs compatible with DAKE, we introduce a chosen-key Fujisaki–Okamoto (CK-FO) transform proven in the QROM, which upgrades IND-CPA double-PKEs to IND-CCA double-KEMs while ensuring the one-sided chosen-key security required by DAKE.

As a concrete instantiation, we propose Maul, a compact double-KEM derived from ML-KEM under the Hint-MLWE assumption. Maul reuses ciphertext components to cut encapsulation size by up to 42% compared to two parallel ML-KEMs. When instantiated with Maul, DAKE achieves overall communication reductions of about 16% (mutual authentication) and 20% (unilateral), outperforming both the double-KEM AKE of Xue *et al.* (ASIACRYPT 2018) and standard ML-KEM-based AKEs.

Keywords: Post-quantum (U)AKE · Double-KEM · eCK · ML-KEM

1 Introduction

In order to face the looming threat of large-scale quantum computers against public-key cryptography, the U.S. National Institute of Standards and Technology (NIST) launched in 2016 a standardization process to select post-quantum encryption and signature primitives resistant to quantum attacks. In particular, key encapsulation mechanisms (KEMs) were chosen to ensure the encryption functionality. KEMs serve as key exchange primitives: a sender uses an encapsulation algorithm to produce a shared secret and a ciphertext, which is then sent to the recipient. The recipient runs a decapsulation algorithm to recover the same shared secret, enabling secure communication. In mid-2022, NIST announced the first post-quantum standards: Kyber/ML-KEM [52] for key encapsulation, and Dilithium/ML-DSA [53], Falcon/FN-DSA [26], and SPHINCS^{+}/SLH-DSA [54] for digital signatures. More recently, in 2025, NIST also selected HQC [29]—a code-based KEM—as an additional standardized alternative to the lattice-based ML-KEM, in order to diversify the set of hardness assumptions.

Quantum Security Comes at a Cost. While post-quantum (PQ) primitives offer conjectured security against quantum-capable adversaries, they often impose large communication costs. For instance, ML-KEM-512 (128-bit security) has 800-byte public keys and 768-byte ciphertexts, compared to the 32-byte keys and ciphertexts of classical Curve25519 DH-KEM [5]. PQ signatures are even larger, often several kilobytes, which makes signature-based authenticated key exchange (AKE) particularly expensive.

Signature-Free AKEs. To mitigate this overhead, signature-free AKEs rely on KEMs or public-key encryptions (PKEs) for both key exchange and authentication. Early constructions stem from [11] and were later strengthened to the CK+ model in [27].[6] More recent works [32,42] improved computational efficiency by employing PKEs instead of KEMs, applying the Fujisaki–Okamoto transform at the protocol level. Yet, communication costs remain essentially unchanged. These designs have inspired practical adaptations, including post-quantum WireGuard [33] and KEMTLS [49], now under IETF standardization [55]. Related work [12] explores modular AKEs with various authentication mechanisms, including KEM-based ones, and proves their security in the CK+ model.

From Two KEMs to a Double-KEM. Even in the unilaterally authenticated setting, achieving (weak) perfect forward secrecy requires two KEMs—one for key agreement and another for authentication. Greater efficiency can be achieved through a dedicated primitive, the *double-KEM* [57], which replaces two independent KEMs with a single mechanism encapsulating under two public keys simultaneously. This design offers better compactness than parallel composition and underlies the double-KEM-based AKE of [57], proven secure in the random oracle model (ROM) and in the standard model under the [IND-CK-CCA, IND-CK-CPA] assumption. The same work introduces Twin-Kyber, a size-optimized double-KEM derived from ML-KEM/Kyber [10,52], proven [IND-CK-CCA, IND-CK-CCA]-secure in the ROM. Notably, Twin-Kyber is obtained by applying a Fujisaki–Okamoto transformation to a [IND-CK-CPA, IND-CK-CPA]-secure double-key double-message PKE.

Security notions for double-KEMs extend those of traditional KEMs. The standard notions of IND-CPA and IND-CCA are generalized to one-sided variants such as [IND-CPA, -] and [IND-CCA, -], where the adversary is given the right secret key sk_R during the security game. In this setting, the encapsulated key k must remain secret even if sk_R is known. Stronger notions, deCanoted [IND-CK-CPA, -] and [IND-CK-CCA, -], additionally allow the adversary to specify the right public key pk_R used during encapsulation.[7] Analogous notions are defined for the right side, denoted [- , IND-CK-CPA] and [- , IND-CK-CCA], where the adversary chooses the left public key pk_L. These can also be extended to the double-key, double-message public-key encryption (2K2M-PKE) setting.

1.1 Our Contributions

We introduce DAKE, DAKE$^\sigma$, and DUAKE, three bandwidth-efficient AKE protocols built from [IND-CK-CCA, IND-CCA] double-KEMs. The underlying double-KEM can be instantiated either using two standard IND-CCA KEMs or more compact double-KEM constructions. Each protocol comes in a two-message version achieving weak perfect forward secrecy (wPFS) and a three-message version achieving full PFS, both proven secure in the standard model under the strong eCK$^\mathsf{w}$ and eCK-PFS variants [17] of the eCK model [40] respectively. These variants capture attack scenarios that are not covered by other generic post-quantum (U)AKEs analyzed in the literature [10,28,32,49].

DAKE provides mutual authentication by combining a double-KEM with a standard KEM, where the double-KEM ensures initiator authentication, while the standard KEM ensures responder authentication. Responder authentication can alternatively be achieved using a signature scheme, yielding the variant DAKE$^\sigma$. We also propose DUAKE, a unilaterally authenticated variant in which only the initiator is authenticated.

Beyond key secrecy, we analyze additional properties, including *inverse match soundness*—which prevents *unknown key-share* (UKS) attacks [9]—and *explicit authentication*—which ensures that any completed session has a unique honest partner. Among the generic post-quantum (U)AKEs in the literature, the only one we are aware of that analyzes explicit authentication is [49].

To ease the construction of double-KEMs compatible with DAKE and its variants, we introduce a chosen-key Fujisaki–Okamoto (CK-FO) transform tailored to 2K2M-PKEs. Our transform is proven secure in the quantum random oracle model (QROM), achieving CCA security while additionally providing chosen-key security for the left encapsulation, thereby yielding a [IND-CK-CCA, IND-CCA] double-KEM, the central ingredient of DAKE. To the best of our knowledge, this is the first extension of the FO paradigm to this setting, enabling constructions of double-KEMs from a wide class of schemes that lack chosen-key security.

[6] [11] proved security in the CK model [13], while [27] extended this to CK+ [38], capturing weak forward secrecy (wFS) and maximal exposure attacks (MEX).

[7] These stronger notions are sometimes directly referred to as [IND-CPA, -] and [IND-CCA, -] in the literature, e.g., in [57].

As a concrete instantiation, we propose Maul, a compact lattice-based double-KEM derived from Kyber/ML-KEM. Maul is obtained by applying our CK-FO transform to $\mathsf{Maul}_{\mathsf{PKE}}$, a double-key double-message PKE built by factoring the large polynomial vector $\mathbf{u}$ from Kyber ciphertexts, reducing ciphertext size by up to 42% compared with two parallel ML-KEM instances. We prove $\mathsf{Maul}_{\mathsf{PKE}}$ [IND-CPA, IND-CPA]-secure under MLWE and hint-MLWE [37], and its CK-FO transformation yields Maul, a compact [IND-CK-CCA, IND-CCA] double-KEM. Instantiating DAKE with Maul achieves roughly 20% global communication savings over the double-KEM-based AKE of Xue et al. [57] and over classical ML-KEM-based AKEs, highlighting the practical advantages of tightly integrated double-KEM designs.

1.2 Technical Overview

Our authenticated key exchange protocol, denoted DAKE (for Double-KEM-based AKE), together with its variants DAKE^{σ} and DUAKE, forms a versatile family of AKE constructions built upon the notion of a double-KEM.

A compact double-KEM enables, at essentially the cost of a single standard KEM encapsulation, the derivation of a shared key that depends on two distinct public keys—an operation that would otherwise require two separate encapsulations. We provide such a construction with Maul, our double-KEM variant of ML-KEM constructed via our CK-FO transform. DAKE and its variants exploit this property to achieve a substantial reduction in bandwidth compared to conventional (U)AKEs.

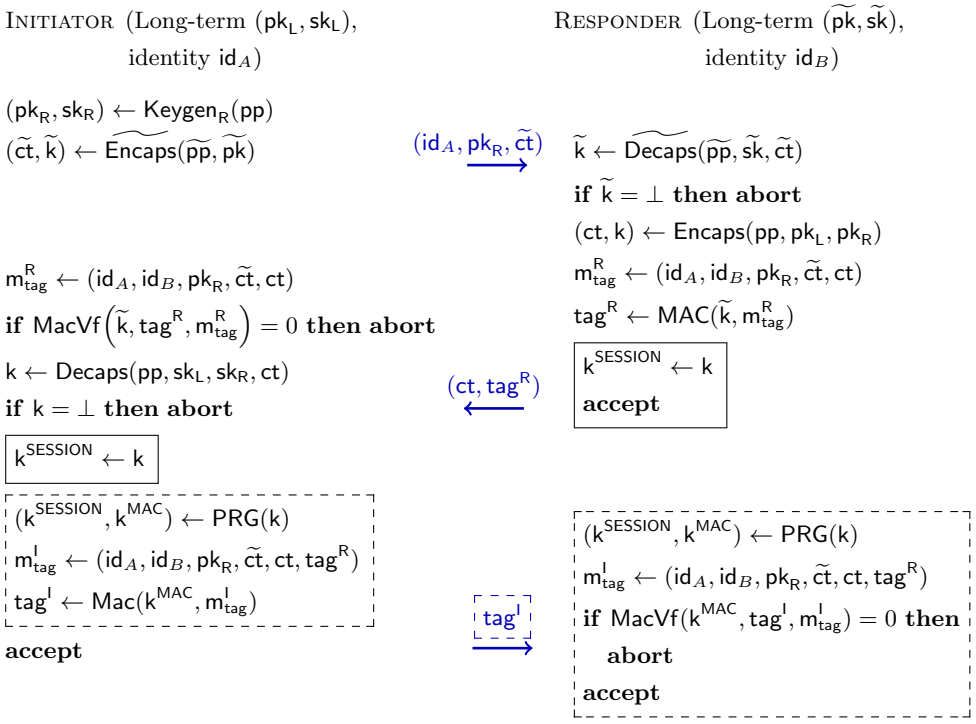

Fig. 1: The DAKE protocol. Public parameters are (1) $(\mathsf{pp}, \widetilde{\mathsf{pp}})$ with $\mathsf{pp} \leftarrow \mathsf{Setup}()$ for the double-KEM and $\widetilde{\mathsf{pp}} \leftarrow \widetilde{\mathsf{Setup}}()$ for the additional KEM, (2) all party identities, and (3) a dictionary mapping identities to public keys. The initiator chooses its peer before sending the first message. DAKE_2 (*resp.* DAKE_3) is obtained by removing the dotted (*resp.* solid) boxes.

AKE from Double-KEM. The main protocol, DAKE, provides mutual authentication by combining a double-KEM with a standard KEM, as specified in Figure 1, where the KEM ensures authentication of the responder. DAKE has two variants: DAKE_2, a two-message protocol achieving weak Perfect Forward Secrecy (wPFS), and DAKE_3, a three-message protocol achieving full PFS. The PFS ensures that a session

key remains secure even if the long-term secrets used in its derivation are compromised after the session completes, the wPFS is discussed below.

All our protocols achieve key secrecy in the standard model: DAKE_2 is proven secure in the $\mathsf{eCK}^{\mathsf{w}}$ model, and DAKE_3 in the eCK-PFS model. These state-of-the-art security models refine the classical eCK model by more precisely capturing the effects of both passive and active adversarial behavior. In most other models, weak PFS requires that a session key remains secure even if the long-term secrets used in its derivation are compromised, under the additional assumptions that the adversary neither (1) modifies messages sent by the session nor (2) forges or modifies messages received by the session. In contrast, $\mathsf{eCK}^{\mathsf{w}}$ and eCK-PFS ensure key secrecy even if the adversary modifies the messages sent by the session, as long as the received messages remain authentic (*i.e.*, neither forged nor modified). These refinements are enabled by the introduction of the *origin session* notion.

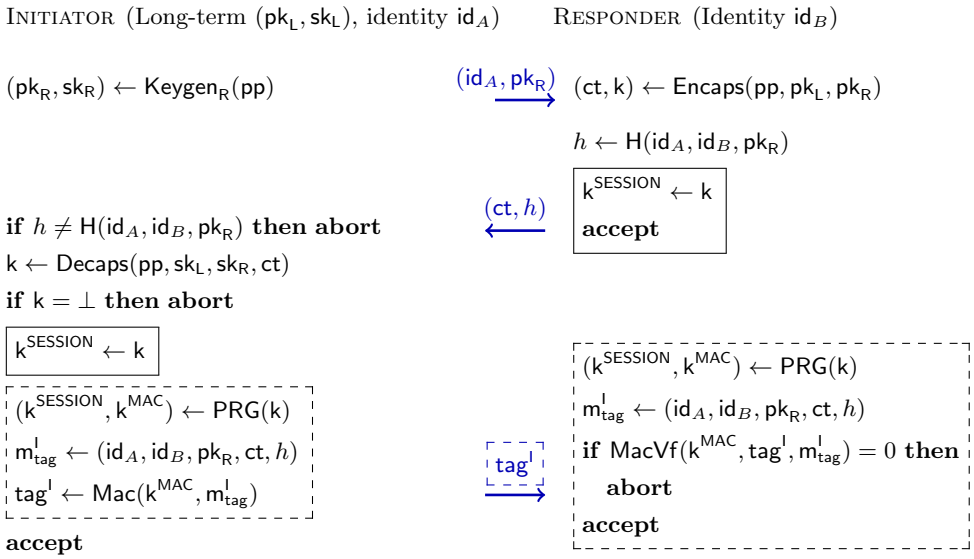

Fig. 2: The DUAKE protocol. Public parameters are (1) $\mathsf{pp} \leftarrow \mathsf{Setup}()$, (2) all party identities, and (3) a dictionary mapping initiators to their public keys. The initiator chooses its peer before sending the first message. DUAKE_2 (*resp.* DUAKE_3) is obtained by removing the dotted (*resp.* solid) boxes.

Responder authentication can be achieved either via an additional standard KEM—as in the baseline DAKE, which typically minimizes overall bandwidth—or via a signature scheme, yielding the variant DAKE^{σ} (see Appendix D.1). The latter is especially relevant when the initiator's public key is unknown to the responder and therefore the initiator must attach a certificate to its first message. Although replacing a KEM ciphertext with a signature usually increases total communication, it reduces the initiator's first-message size, balancing communication costs when certificates or other extra data must be sent upfront.

Unilaterally authenticated AKEs (UAKEs), in which only one party is authenticated, have been studied in several contexts such as Tor and TLS [51,20,25,30,35,39,49]. DUAKE, specified in Figure 2, is our unilateral variant of DAKE, tailored for such settings. To analyze its security, we introduce the $\mathsf{u\text{-}eCK}^{\mathsf{w}}$ and u-eCK-PFS models—unilateral adaptations of the $\mathsf{eCK}^{\mathsf{w}}$ and eCK-PFS frameworks. Essentially, $\mathsf{u\text{-}eCK}^{\mathsf{w}}$ and u-eCK-PFS remove from $\mathsf{eCK}^{\mathsf{w}}$ and eCK-PFS the attack scenarios that cannot be prevented by a unilateral protocol (for instance, attacks where an active adversary impersonates a party that possesses no long-term keys). While several security models for UAKE have been proposed (*e.g.*, [30] presents a unilateral extension of the eCK model), our models strengthen these definitions by incorporating the refinements introduced in $\mathsf{eCK}^{\mathsf{w}}$ and eCK-PFS, thereby capturing a broader range of attack scenarios. Practical use cases of DUAKE are discussed in more detail in Appendix I.

The security models we consider define a protocol as secure if it satisfies two properties, called *key secrecy* and *match soundness*. The key secrecy guarantees offer resistance to *maximal exposure* (MEX), *key-compromise impersonation* (KCI) attacks and *unknown key-share* (UKS) attacks. They also imply a form of *implicit authentication* by ensuring that the session key remains secret under active attacks when-

ever the peer of the session is not corrupted (in which case the peer is said to be implicitly authenticated)[8]. More precisely, both the initiator and responder are implicitly authenticated in $\mathsf{DAKE}_2, \mathsf{DAKE}_3, \mathsf{DAKE}_2^\sigma$ and DAKE_3^σ, whereas only the initiator benefits from implicit authentication in DUAKE_2 and DUAKE_3. The *match soundness* property is linked to the modelization used by the security model, and is described in Section 3.1.

Table 1: Security properties of our AKE protocols. An asterisk (✓*) indicates that the corresponding property can be achieved through a minor modification that does not affect the exchanged messages. See Section 3.3 for details on how inverse-match soundness can be obtained using an additional KDF.

Protocol	PFS	wPFS	(Match Snd., Inv. Match)	Init-Explicit Authent.	Resp-Explicit Authent.
DAKE_2	✗	✓	(✓, ✓*)	✗	✓
DAKE_3	✓	✓	(✓, ✓*)	✓	✓
DAKE_2^σ	✗	✓	(✓, ✓*)	✗	✓
DAKE_3^σ	✓	✓	(✓, ✓*)	✓	✓
DUAKE_2	✗	✓	(✓, ✓*)	✗	✗
DUAKE_3	✓	✓	(✓, ✓*)	✓	✗

To capture security across a broad range of settings, we further study several additional properties for all our AKE protocols, Table 1 summarizing which ones are achieved by the different variants of DAKE. Specifically, we consider *inverse match soundness* (which prevents UKS attacks under a broader range of adversarial behaviors than key secrecy alone), and *explicit authentication*. We refine the latter by distinguishing between Init-*explicit* and Resp-*explicit* authentication. In short, Init-explicit (resp. Resp-explicit) authentication ensures that if the peer of a responder (resp. initiator) session is not corrupted (*i.e.*, its long-term secret key is unknown to the adversary) and this session has completed the protocol, then there exists an initiator (resp. responder) session that has honestly executed the protocol with it. We show that Init-explicit authentication is provided by $\mathsf{DAKE}_3, \mathsf{DAKE}_3^\sigma$ and DUAKE_3, while Resp-explicit authentication is ensured by $\mathsf{DAKE}_2, \mathsf{DAKE}_3, \mathsf{DAKE}_2^\sigma$ and DAKE_3^σ.

Chosen-key FO Transform. Our Chosen-Key Fujisaki–Okamoto (CK-FO) transform enables the instantiation of DAKE and its variants from a broad class of underlying primitives by providing a generic method to transform a wide class of [IND-CPA, IND-CPA] double-key double-message PKE (2K2M-PKE) schemes into [IND-CK-CCA, IND-CCA] double-KEMs. The resulting construction guarantees chosen-key CCA security on one side and standard CCA security on the other—precisely the security properties required for instantiating DAKE and its variants. Although our concrete instantiation is lattice-based, the technique also paves the way for compact double-KEMs under other assumptions, such as code-based constructions inspired by HQC [29].

A central idea of the CK-FO transform is to restrict attention to a subclass of 2K2M-PKE constructions derived from two underlying PKEs, and to leverage not only the security of the combined scheme but also that of its constituents. The transform admits proofs of security in both the random oracle model (ROM) and the quantum random oracle model (QROM), the latter via semi-classical oracle techniques. Beyond its conceptual novelty, the CK-FO transform significantly broadens the design space: it enables double-KEMs to be instantiated from compact [IND-CPA, IND-CPA] 2K2M-PKEs that do not possess any chosen-key security.

Lattice-based Compact Double-KEM. While DAKE, DAKE^σ, and DUAKE are generic constructions, their practical efficiency depends on the availability of compact double-KEMs. Prior work such as Twin-Kyber [57] demonstrated feasibility but left ample room for optimization. To validate our framework, we introduce Maul, a [IND-CK-CCA, IND-CCA] double-KEM derived from ML-KEM/Kyber, and use it to instantiate DAKE. By exploiting structural asymmetries in Kyber ciphertexts, Maul reduces ciphertext size by up to 42% compared to two parallel ML-KEMs, thereby minimizing message sizes in complete

[8] Some additional conditions that would trivially reveal the session key to the adversary are omitted here for simplicity.

AKE handshakes. Table 2 compares Maul's ciphertext and public-key sizes with those of two parallel ML-KEMs and Twin-Kyber [57].

Table 2: Sizes (bytes) of the ciphers and public keys of our double-KEM Maul, compared to two parallel ML-KEM instances and to Twin-Kyber. Security levels follow the categories defined in the NIST PQC call for proposals.[9]

	Security Level	**Maul (this work)**	**Twin-Kyber** [57]	**ML-KEM** [52]
Ciphertext	1	896	–	1,536
	3	1,440	–	2,176
	5	1,856	2,464	3,136
Public Key	1	826	–	800
	3	1,240	–	1,184
	5	1,691	1,568	1,568

Maul results from applying the CK-FO transform to $\mathsf{Maul}_{\mathsf{PKE}}$, a 2K2M-PKE that we now describe. At a high level, ML-KEM (Kyber) ciphertexts consist of two parts: a vector component $\mathbf{u}$ and a scalar component v, where $\mathbf{u}$ encodes ephemeral randomness and v encodes the message. These components are highly asymmetric in size; for instance, in ML-KEM-768, $\mathbf{u}$ occupies 950 bytes, while v is only 128 bytes. $\mathsf{Maul}_{\mathsf{PKE}}$ exploits this imbalance by factoring the larger vector component $\mathbf{u}$, yielding ciphertexts of the form $(\mathbf{u}, v_1, v_2)$ that encrypt two messages while significantly reducing overall ciphertext size. A related construction was proposed in [36], though its security analysis was limited to a setting where the adversary is not given access to any secret key.

The security of $\mathsf{Maul}_{\mathsf{PKE}}$ is based on the Hint-MLWE assumption, a natural variant of MLWE in which the adversary receives a partial linear hint about the secret and error. To illustrate this, consider a simplified version of the [IND-CPA, IND-CPA] $\mathsf{Maul}_{\mathsf{PKE}}$ encrypting two messages m_{L} and m_{R} as follows:

$$(\mathbf{u} = \mathbf{A}^{\intercal}\mathbf{s} + \mathbf{e},\ v_1 = \mathbf{t}_{\mathsf{L}}^{\intercal}\mathbf{s} + e_1 + \lceil q/2 \rceil m_{\mathsf{L}},\ v_2 = \mathbf{t}_{\mathsf{R}}^{\intercal}\mathbf{s} + e_2 + \lceil q/2 \rceil m_{\mathsf{R}}),$$

where $\mathbf{t}_{\mathsf{L}} = \mathbf{A}\mathbf{s}_{\mathsf{L}} + \mathbf{e}_{\mathsf{L}}$ and $\mathbf{t}_{\mathsf{R}} = \mathbf{A}\mathbf{s}_{\mathsf{R}} + \mathbf{e}_{\mathsf{R}}$ are public keys derived from secret keys $\mathbf{s}_{\mathsf{L}}, \mathbf{s}_{\mathsf{R}}$ and errors $\mathbf{e}_{\mathsf{L}}, \mathbf{e}_{\mathsf{R}}$. When the adversary knows the right-hand secret key $(\mathbf{s}_{\mathsf{R}}, \mathbf{e}_{\mathsf{R}})$, the component $(\mathbf{u}, v_1)$ becomes a MLWE instance with an additional known noisy linear relation: $v_2 - \mathbf{s}_{\mathsf{R}}^{\intercal}\mathbf{u} - \lceil q/2 \rceil m_{\mathsf{R}} = (\mathbf{e}_{\mathsf{R}}^{\intercal}, -\mathbf{s}_{\mathsf{R}}^{\intercal})\left(\begin{smallmatrix}\mathbf{s}\\ \mathbf{e}\end{smallmatrix}\right) + e_2$.

This precisely matches the definition of Hint-MLWE, and under this assumption, the left ciphertext $(\mathbf{u}, v_1)$ remains indistinguishable from uniform, thus ensuring the security of m_{L}.

We instantiate Maul with concrete parameters and estimate its security by adapting the heuristic framework of [21], which models non-Gaussian error distributions as Gaussians and disregards certain discrete-Gaussian-specific factors in the reduction. To support robust parameter selection, we complement this analysis with an evaluation of the best known attacks against Hint-MLWE.

The techniques underlying Maul can also be directly adapted to integer lattices, yielding a FrodoKEM-inspired [43] double-KEM with analogous security guarantees under the LWE and Hint-LWE assumptions; we leave a detailed treatment of this variant to future work. We further believe that these techniques could extend to a broader class of ElGamal-style KEMs, such as HQC [29], which has been selected for NIST standardization.

Performance of DAKE Instantiations. Beyond the security properties they guarantee, the strength of DAKE, DAKE^{σ}, and DUAKE lies in their ability to transform compact double-KEMs into bandwidth-efficient (U)AKE protocols. Instantiated with Maul, they demonstrate this potential concretely. Table 3 reports the message sizes for DAKE, DAKE^{σ}, and DUAKE instantiated with Maul, under the assumption that MAC tags, hash outputs, and identifiers have a bit-length twice the security parameter.

For comparison, existing ML-KEM-based AKE protocols (e.g., [10,28]) require transmitting at least one ephemeral public key and three ciphertexts, amounting to 6,272 bytes at NIST 5 security. The Twin-Kyber-based AKE of [57] instead transmits one ephemeral public key and two double-KEM ciphertexts,

[9] https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/call-for-proposals

Table 3: Message sizes (bytes) for DAKE, DAKE$^\sigma$, and DUAKE using Maul, ML-KEM, and ML-DSA. The optional third message corresponds to the full-PFS variants. Security levels follow NIST PQC categories.

Protocol	Sec. Level	Msg. 1 (Init.)	Msg. 2 (Resp.)	Msg. 3 (Init.)
DAKE	1	1,626	928	32
	3	2,376	1,488	48
	5	3,323	1,920	64
DAKE$^\sigma$	1	858	3,316	32
	3	1,288	4,749	48
	5	1,755	6,483	64
DUAKE	1	858	928	32
	3	1,288	1,488	48
	5	1,755	1,920	64

totaling 6,496 bytes at the same security level, since it employs two double-KEMs in parallel rather than combining a double-KEM with a standard KEM as in DAKE. In contrast, DAKE instantiated with Maul achieves approximately 16% overall communication savings (5,243 bytes) compared to these protocols (NIST 5), with an average reduction of 16% across NIST 1-5 security levels.

Similarly, existing ML-KEM-based UAKE protocols (e.g., [10,49]) require transmitting at least one ephemeral public key and two ciphertexts, for a total of 4,704 bytes at NIST 5 security. In comparison, DUAKE instantiated with Maul achieves about 21% overall communication savings (3,675 bytes) relative to these protocols (NIST 5), with an average reduction of 20% across NIST 1-5 levels.

1.3 Related Works

The only generic AKE we are aware of that leverages a double-KEM is the construction of Xue et al. [57], which provides weak perfect forward secrecy (wPFS) but no explicit authentication. Although it can, in principle, be instantiated with Maul (see Appendix G), their protocol requires transmitting two double-KEM ciphertexts, which is essentially equivalent to four standard KEM encapsulations, whereas it is well known that wPFS AKEs can be realized with only three standard encapsulations [10]. Consequently, the construction does not fully exploit the bandwidth advantage that double-KEMs can offer over standard KEMs. Xue et al. [57] also proposed an FO transform in the ROM that upgrades any 2K2M-PKE with [IND-CK-CPA, IND-CK-CPA] security into a double-KEM with $\left[\text{IND-CK-CCA}, \text{IND-CK-CCA}\right]$ security. Their transform, however, neither achieves one-sided chosen-key security nor extends to the QROM.

In concurrent and independent work, of which we became aware only after completing ours, Hashimoto, Katsumata, Niot, and Wiggers [31] propose a lattice-based *Reinforced*-KEM (Rebar) that also reuses the **u** component of ML-KEM ciphertexts. Unlike our double-KEM Maul, Rebar employs discrete Gaussian distributions but achieves comparable security under the same hardness assumptions. Their construction relies on an FO transform, which (when translated into the double-KEM/2K2M-PKE setting) upgrades a $\left[\text{IND-CK-CPA}, \text{IND-CPA}\right]$ 2K2M-PKE into a $\left[\text{IND-CK-CCA}, \text{IND-CCA}\right]$ double-KEM. This achieves CCA security on both sides, similar in spirit to the FO of [57], but without adding chosen-key security—our main novelty with the CK-FO transform. While Rebar is applied to optimize post-quantum WireGuard (with proofs in the eCK model), our focus is on designing a general family of (U)AKE protocols. This difference is also reflected in the analyzed properties: they study *identity hiding*, which we leave aside, whereas we emphasize explicit authentication, central to our framework.

Organization

The rest of the paper is organized as follows. Section 2 introduces the necessary notation and background. Section 3 presents our AKE protocols and their security properties. Section 4 describes the CK-FO transform and analyzes its security. Section 5 introduces Maul, establishing its security and parameter choices.

Acknowledgements

This work has been partially supported by the European Union - Next Generation EU through the France Relance program and by the French government through the France 2030 program under the project RESQUE. This work has been further supported by the French Agence Nationale de la Recherche through the SecNISQ project (ANR-21-CE47-0014).

2 Preliminaries

We denote by $\{0,1\}^*$ the set of binary strings. For any elements a, b from a set, we sometimes write $a = b$ to denote the Boolean predicate that evaluates to 1 if $a = b$ and 0 otherwise. We use the notation $\mathsf{Alg}(\mathsf{arg}_1, \ldots, \mathsf{arg}_n; r)$ to denote the execution of a probabilistic algorithm Alg on inputs $\mathsf{arg}_1, \ldots, \mathsf{arg}_n$ using randomness r. When the randomness is sampled internally by the algorithm, we omit r for brevity. When a tuple $(x_1, \ldots, x_n)$ is used as an argument to a function expecting a string – such as $\mathsf{H}((\mathsf{k}, m))$ or $\mathsf{Mac}(\mathsf{k}, t, (m_1, m_2))$ – we implicitly refer to an injective concatenation of $x_1, \ldots, x_n$, meaning that the original components can be unambiguously recovered from it. When the lengths of $x_1, \ldots, x_n$ are fixed and publicly known, a simple concatenation suffices. Otherwise, an appropriate encoding must be applied to ensure injectivity.

Double-KEM (2K-KEM). Introduced in [57], a double-KEM, also referred to as a double-key KEM, abbreviated as 2K-KEM in this paper, is a public-key encapsulation mechanism that operates with two pairs of public and secret keys.

Formally, a 2K-KEM $\mathsf{dKEM} = (\mathsf{Setup}, \mathsf{Keygen}_\mathsf{L}, \mathsf{Keygen}_\mathsf{R}, \mathsf{Encaps}, \mathsf{Decaps})$ is a tuple of five probabilistic polynomial-time (PPT) algorithms. The $\mathsf{Setup}()$ algorithm is responsible for generating public parameters pp. With these parameters, $\mathsf{Keygen}_\mathsf{L}(\mathsf{pp})$ generates a left pair of public and secret keys $(\mathsf{pk}_\mathsf{L}, \mathsf{sk}_\mathsf{L}) \in \mathsf{PK}_\mathsf{L} \times \mathsf{SK}_\mathsf{L}$, while $\mathsf{Keygen}_\mathsf{R}(\mathsf{pp})$ provides a right pair $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R}) \in \mathsf{PK}_\mathsf{R} \times \mathsf{SK}_\mathsf{R}$. The $\mathsf{Encaps}(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R})$ algorithm takes the public parameters and both public keys pk_L and pk_R to produce a pair $(\mathsf{ct}, \mathsf{k}) \in \mathsf{CT} \times \mathsf{Keys}$, consisting of a ciphertext and a key. The $\mathsf{Decaps}(\mathsf{pp}, \mathsf{sk}_\mathsf{L}, \mathsf{sk}_\mathsf{R}, \mathsf{ct})$ algorithm then uses the public parameters, secret keys sk_L and sk_R, along with the ciphertext ct, to derive and output the key k or $\perp$ to indicate a decryption failure.

The security games we consider are defined in Figure 3. These games are variations of the indistinguishability games for KEMs. As explained in the introduction, these variations depend on three factors: (1) which side (left or right) is considered, (2) whether we consider security with chosen-key (CK), and (3) whether a decapsulation oracle is provided to the adversary, leading to CPA or CCA variations. We say that a 2K-KEM is $[\mathsf{X}, \mathsf{Y}]$ secure if it is $[\mathsf{X}, \text{-}]$ secure and $[\text{-}, \mathsf{Y}]$ secure. The advantage in these games, following the notations of Figure 3, is expressed as $|\Pr[b = b'] - 1/2|$.

We also consider key-recovery games $[\mathsf{Key\text{-}Recov}, \cdot]$ and $[\cdot, \mathsf{Key\text{-}Recov}]$. By symmetry, we only define $[\mathsf{Key\text{-}Recov}, \cdot]$. First, we suppose the existence of a function $\mathsf{RecovPKL}$, such that, for any $\mathsf{pp} \leftarrow \mathsf{Setup}()$ and $(\mathsf{pk}_\mathsf{L}, \mathsf{sk}_\mathsf{L}) \leftarrow \mathsf{Keygen}_\mathsf{L}()$, we have $\mathsf{pk}_\mathsf{L} = \mathsf{RecovPKL}(\mathsf{pp}, \mathsf{sk}_\mathsf{L})$. The advantage of the game $[\mathsf{Key\text{-}Recov}, \cdot]$ is defined as

$$\Pr\left[\mathsf{pk}_\mathsf{L} = \mathsf{RecovPKL}(\mathsf{pp}, \mathsf{sk}'_\mathsf{L}) : \begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(), (\mathsf{pk}_\mathsf{L}, \mathsf{sk}_\mathsf{L}) \leftarrow \mathsf{Keygen}_\mathsf{L}(\mathsf{pp}),\\ \mathsf{sk}'_\mathsf{L} \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}_\mathsf{L})\end{array}\right].$$

The notions of correctness and spreadness for a 2K-KEM are defined as follows:

- For $\delta \geq 0$, a 2K-KEM is δ-correct if for $\mathsf{pp} \leftarrow \mathsf{Setup}()$,$(\mathsf{pk}_\mathsf{L}, \mathsf{sk}_\mathsf{L}) \leftarrow \mathsf{Keygen}_\mathsf{L}(\mathsf{pp})$, $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R}) \leftarrow \mathsf{Keygen}_\mathsf{R}(\mathsf{pp})$, and $(\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Encaps}(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R})$, the probability that $\mathsf{Decaps}(\mathsf{pp}, \mathsf{sk}_\mathsf{L}, \mathsf{sk}_\mathsf{R}, \mathsf{ct}) = \mathsf{k}$ holds is at least $1 - \delta$.
- For $\gamma > 0$, a 2K-KEM is γ-spread if, for any public parameters $\mathsf{pp} \leftarrow \mathsf{Setup}()$, any $\mathsf{pk}'_\mathsf{L} \in \mathsf{PK}_\mathsf{L}$, $\mathsf{pk}'_\mathsf{R} \in \mathsf{PK}_\mathsf{R}$, and any ciphertext $\mathsf{ct}' \in \mathsf{CT}$, we have: $\Pr\left[\mathsf{ct} = \mathsf{ct}' : (\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Encaps}(\mathsf{pp}, \mathsf{pk}'_\mathsf{L}, \mathsf{pk}'_\mathsf{R})\right] \leq 2^{-\gamma}$.

Double-key Double-message PKE (2K2M-PKE). We consider a variant of the double-key PKE of [57] that encrypts a pair of messages. A double-key double-message PKE (2K2M-PKE), functions as a PKE with two pairs of public and secret keys to encrypt two messages into a single ciphertext. Formally, a 2K2M-PKE $\mathsf{DPKE} = (\mathsf{Setup}, \mathsf{Keygen}_\mathsf{L}, \mathsf{Keygen}_\mathsf{R}, \mathsf{dEnc}, \mathsf{dDec})$ consists of five PPT algorithms. The $\mathsf{Setup}()$ algorithm generates and outputs the public parameters pp. With these parameters, $\mathsf{Keygen}_\mathsf{L}(\mathsf{pp})$ generates a left pair of public and secret keys $(\mathsf{pk}_\mathsf{L}, \mathsf{sk}_\mathsf{L}) \in \mathsf{PK}_\mathsf{L} \times \mathsf{SK}_\mathsf{L}$, while $\mathsf{Keygen}_\mathsf{R}(\mathsf{pp})$ produces a right pair $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R}) \in \mathsf{PK}_\mathsf{R} \times \mathsf{SK}_\mathsf{R}$. The $\mathsf{dEnc}(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})$ algorithm takes the public parameters, the public keys pk_L and pk_R, and two messages $(m_\mathsf{L}, m_\mathsf{R}) \in \mathcal{M}_\mathsf{L} \times \mathcal{M}_\mathsf{R}$ to generate a ciphertext $\mathsf{ct} \in \mathsf{CT}$. The $\mathsf{dDec}(\mathsf{pp}, \mathsf{sk}_\mathsf{L}, \mathsf{sk}_\mathsf{R}, \mathsf{ct})$ algorithm then uses the public parameters, secret keys sk_L and sk_R, along with the ciphertext ct, to recover and output the messages $(m_\mathsf{L}, m_\mathsf{R}) \in \mathcal{M}_\mathsf{L} \times \mathcal{M}_\mathsf{R}$.

The security games for a 2K2M-PKE are outlined in Figure 4. As for the security games of 2K-KEM, these games are variations of the indistinguishability games for PKE, adapted to the presence of two secret and public keys. For clarity, we use the following naming convention for security games: names of the form $[\mathsf{X}, \text{-}]$ (for some text X, such as IND-CPA) denote games whose security relies on the secrecy of

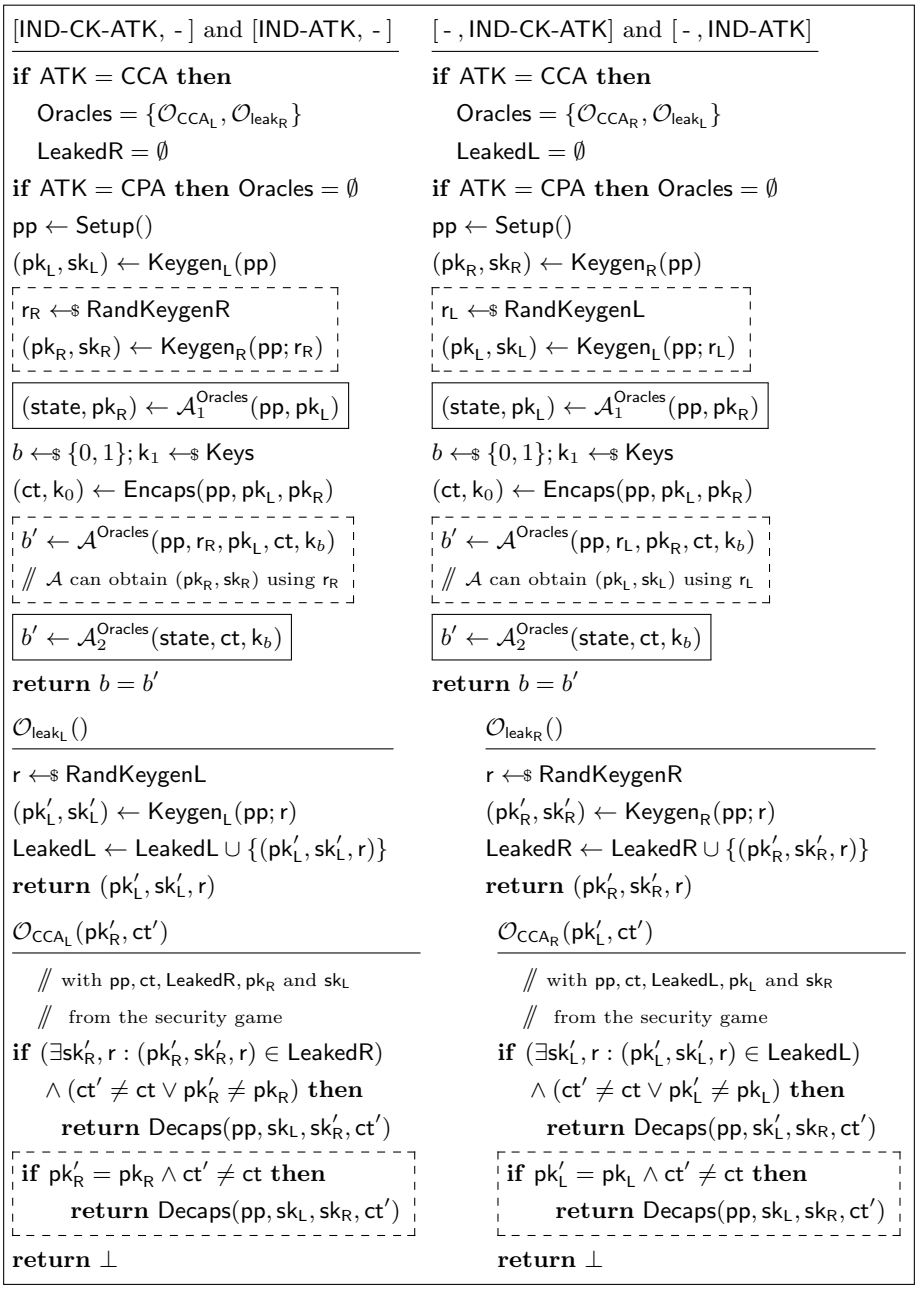

Fig. 3: 2K-KEM security games. $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. Solid boxes are taken in account only for chosen-key security games (*i.e.* whose name contains CK) while dashed boxes are taken in account only for non-chosen-key games. RandKeygenL/RandKeygenR denote the randomness distributions used by the $\mathsf{Keygen_L}$/$\mathsf{Keygen_R}$ algorithms.

the left secret key, while [-, Y] denotes those in which security relies on the secrecy of the right secret key. We say that a 2K2M-PKE is [X, Y]-secure if it is [X, -]-secure and [-, Y]-secure.

The advantage in these games, following the notations of Figure 4, is expressed as $|\Pr[b = b'] - 1/2|$. Finally, for $\delta \geq 0$, a 2K2M-PKE is δ-correct if:

$$\mathbb{E}\left(\max_{\mathsf{m} \in \mathcal{M}_\mathsf{L} \times \mathcal{M}_\mathsf{R}} \Pr[\mathsf{dDec}(\mathsf{pp}, \mathsf{sk_L}, \mathsf{sk_R}, \mathsf{dEnc}(\mathsf{pp}, \mathsf{pk_L}, \mathsf{pk_R}, \mathsf{m})) = \mathsf{m})]\right) \geq 1 - \delta\,,$$

where expected value is taken over $\mathsf{pp} \leftarrow \mathsf{Setup}()$, $(\mathsf{pk_L}, \mathsf{sk_L}) \leftarrow \mathsf{Keygen_L}(\mathsf{pp})$, and $(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{Keygen_R}(\mathsf{pp})$.

$[\mathsf{IND\text{-}CPA}, \text{-}]$	$[\text{-}, \mathsf{IND\text{-}CPA}]$
$\mathsf{pp} \leftarrow \mathsf{Setup}()$	$\mathsf{pp} \leftarrow \mathsf{Setup}()$
$(\mathsf{pk_L}, \mathsf{sk_L}) \leftarrow \mathsf{Keygen_L}(\mathsf{pp})$	$(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{Keygen_R}(\mathsf{pp})$
$\mathsf{r_R} \leftarrow\!\!\$\ \mathsf{RandKeygenR}$	$\mathsf{r_L} \leftarrow\!\!\$\ \mathsf{RandKeygenL}$
$(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{Keygen_R}(\mathsf{pp}; \mathsf{r_R})$	$(\mathsf{pk_L}, \mathsf{sk_L}) \leftarrow \mathsf{Keygen_L}(\mathsf{pp}; \mathsf{r_L})$
$(\mathsf{state}, m_{\mathsf{L},0}, m_{\mathsf{L},1}, m_{\mathsf{R}})$	$(\mathsf{state}, m_{\mathsf{R},0}, m_{\mathsf{R},1}, m_{\mathsf{L}})$
$\leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk_R}, \mathsf{sk_R}, \mathsf{r_R}, \mathsf{pk_L})$	$\leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk_L}, \mathsf{sk_L}, \mathsf{r_L}, \mathsf{pk_R})$
$\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk_L}, \mathsf{pk_R}, (m_{\mathsf{L},0}, m_{\mathsf{R}}))$	$\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk_L}, \mathsf{pk_R}, (m_{\mathsf{L}}, m_{\mathsf{R},0}))$
$\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk_L}, \mathsf{pk_R}, (m_{\mathsf{L},1}, m_{\mathsf{R}}))$	$\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk_L}, \mathsf{pk_R}, (m_{\mathsf{L}}, m_{\mathsf{R},1}))$
$b \leftarrow\!\!\$\ \{0,1\}$	$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow \mathcal{A}_2(\mathsf{state}, \mathsf{ct}_b)$	$b' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{state}, \mathsf{ct}_b)$
$\mathbf{return}\ b = b'$	$\mathbf{return}\ b = b'$

Fig. 4: Security games of 2K2M-PKE. $\mathsf{RandKeygenL}/\mathsf{RandKeygenR}$ denote the distributions of the randomness used by the $\mathsf{Keygen_L}/\mathsf{Keygen_R}$ algorithms.

3 Authenticated Key Exchanges from Double-KEMs

In this section, we formalize and analyze three authenticated key exchange (AKE) protocols built from $\big[\mathsf{IND\text{-}CK\text{-}CCA}, \mathsf{IND\text{-}CCA}\big]$ double-KEMs. We first recall the security models we use, then present the base protocol DAKE together with its two variants, DAKE^{σ} and DUAKE. In addition to the two properties required by these security models—key secrecy and match soundness—, we also consider inverse match soundness and explicit authentication.

3.1 The eCK Security Model and Its Variants

Our analysis builds on two strengthened variants of the extended Canetti–Krawczyk (eCK) model [40]: $\mathsf{eCK}^{\mathsf{w}}$ and eCK-PFS [17].[10] The former guarantees *weak perfect forward secrecy* (wPFS), and the latter *perfect forward secrecy* (PFS). Both variants ensure key secrecy against a broader class of attack scenarios than those covered by eCK and CK^{+} [38].[11]

We provide an overview of these variants; a full formalization is deferred to Appendix C. In this model, the adversary can create parties (as initiators or responders), activate sessions, and interact arbitrarily with them by sending chosen messages. Each session represents a protocol execution by a party (the actor of the session). A simple protocol execution between two parties is therefore represented by two sessions. In each session, the party that the actor intends to communicate with is called its peer. Each session can be in one of these states: running, completed (the protocol finished without error), or aborted.

Multiple relations can be defined between sessions, and two are of particular importance for the security properties considered in this work:

- Informally, two sessions s, s' are said to be *matching* if they completed the protocol together. More formally, this requires that (1) they both completed, (2) they have distinct roles, (3) the peer of each session equals the actor of the other, and (4) they share the same transcript of sent and received messages.[12]
- A (not necessarily completed) session s' is said to be the *origin* of a session s if the transcript of messages received by s equals the transcript of messages sent by s'.

The eCK model and its variants consider powerful adversarial queries:

- corrupt: reveals a party's long-term secret key.

[10] As explained in Section 3.2, the analysis of DUAKE relies on an adaptation of these models to the unilateral setting.

[11] This strengthening is due to the use of the *origin session* notion, rather than matching sessions, in certain scenarios [17].

[12] When comparing the transcripts of two sessions with distinct roles, sent and received messages are inverted, since the outgoing messages of one session correspond to the incoming messages of the other. This convention is applied throughout the paper when comparing transcripts of sessions with different roles.

– eph-key: reveals some ephemeral elements computed by a running or completed session. Revealed elements are explicitly specified in the protocol[13].
– session-key: reveals the session key of a completed session.

Using this adversarial setting, we can now define multiple security properties. More precisely, we first define *key secrecy* and *match soundness*, which are required for a protocol to be considered secure in the $\mathsf{eCK}^{\mathsf{w}}$/eCK-PFS models. Then, we define two additional properties : inverse-match soundness and explicit authentication, that are also desirable for AKEs.

Key Secrecy. Key secrecy is measured by a real-or-random game: the adversary chooses a *test session* $\mathsf{s}^{\mathsf{Test}}$ of its choice, receives either its true session key or a uniformly sampled one, and must distinguish between the two.

While the adversary can make adversarial queries before and after receiving the challenge key, restrictions on adversarial queries are required to prevent unavoidable attacks (*e.g.*, querying session-key on the tested session) and leads to five attack scenarios summarized in Table 4. More precisely, $\mathsf{eCK}^{\mathsf{w}}$—guaranteeing only weak perfect forward secrecy (wPFS)—only authorizes scenarios 1 to 4 while eCK-PFS authorizes scenarios 1 to 6—guaranteeing full perfect forward secrecy (PFS). Moreover, both models also protect against *maximal exposure* (MEX) attacks and *key-compromise impersonation* (KCI) attacks and (UKS) attacks[14].

The key secrecy game with the scenarios of $\mathsf{eCK}^{\mathsf{w}}$ (*resp.* eCK-PFS) is called $\mathsf{eCK}^{\mathsf{w}}$-KeySec (*resp.* eCK-PFS-KeySec).

Table 4: Summary of the attack scenarios considered. The adversary $\mathcal{A}$ is never allowed to query session-key on $\mathsf{s}^{\mathsf{Test}}$ or on any session matching $\mathsf{s}^{\mathsf{Test}}$. Here, A denotes the actor of $\mathsf{s}^{\mathsf{Test}}$, P its peer, and $\mathsf{s}^{\mathsf{Origin}}$ an origin session of $\mathsf{s}^{\mathsf{Test}}$. *Post-comp.* indicates that the corruption query is made after $\mathsf{s}^{\mathsf{Test}}$ has completed, and *possible* means that the scenario includes both *yes* and *no* cases.

Scenario	corrupt(A)	corrupt(P)	eph-key($\mathsf{s}^{\mathsf{Test}}$)	Origin session $\mathsf{s}^{\mathsf{Origin}}$ exists	eph-key($\mathsf{s}^{\mathsf{Origin}}$)
1	no	no	possible	possible	possible
2	yes	no	no	possible	possible
3	no	yes	possible	yes	no
4	yes	yes	no	yes	no
5	yes	post-comp.	no	no	–
6	no	post-comp.	possible	no	–

Match Soundness. Match soundness is defined through a game in which the adversary faces no restrictions on its queries. The adversary wins if it can produce either of the following:
– two completed sessions that are matching but derive different session keys.
– three distinct completed sessions that share the same message transcript.[15]

Informally, it ensures that the matching relation correctly captures when two sessions have executed the protocol together.

Inverse-Match Soundness. Inverse-Match Soundness is measured by a game in which the adversary faces no restrictions on its queries. The adversary wins if it produces two distinct completed sessions that output the same session key without being matching.

[13] Our definition is more general than the original eCK model and its variants, as the original model fixes this query to reveal exactly the ephemeral secret keys (i.e., the randomness coins) of session s. However, choice of which elements can be leaked is significant and may have non-trivial consequences, as illustrated in [16].
[14] Attacks where two parties derive the same session key but disagree on their peer identities.
[15] Recall that when two sessions have distinct roles, transcript equality is checked up to the inversion of sent and received messages, as explained in Footnote 12.

We believe that one of the main purposes of this property is to ensure UKS security independently of the adversary's queries[16]. While we are not sure about the practical interest of this property, we explain on page 14 how to modify our protocols in a simple way to achieve it.

Explicit Authentication. Explicit Authentication ensures that whenever a session s is completed with an uncorrupted peer, it has done so with an honest partner session whose actor is $\mathsf{s}_{\mathsf{Peer}}$—we denote by $\mathsf{s}_{\mathsf{Actor}}$ and $\mathsf{s}_{\mathsf{Peer}}$ the actor and peer of s. Care is needed in formalizing this notion: if s sends the final message of the exchange, the mere uncorruptedness of the peer cannot guarantee that this message is actually received by the honest partner.

For additional expressivity we distinguish two properties: Init-explicit authentication and Resp-explicit authentication. We take a role $\mathsf{X} \in \{\mathsf{Init}, \mathsf{Resp}\}$ and denote by $\overline{\mathsf{X}}$ the role distinct of X. In the X-explicit authentication game $\mathsf{X}\text{-}\mathsf{ExplicitAuth}$, the adversary selects a completed session s with role $\overline{\mathsf{X}}$, under the sole restriction on adversarial queries that $\mathsf{s}_{\mathsf{Peer}}$ was not corrupted before s completed. The adversary wins if there does not exist a session s' with actor $\mathsf{s}_{\mathsf{Peer}}$, peer $\mathsf{s}_{\mathsf{Actor}}$ and role X, such that:

- *If the sessions with role* X *send the final message in the AKE:* s and s' share the same message transcript, except possibly for the final message sent by s and received by s'. Note that s' is not required to be completed.
- *If the sessions with role* X *do not send the final message in the AKE:* s and s' share the same message transcript, and s' is completed. Note that in this case, s and s' are matching.

3.2 Our AKE Protocols

The base protocol DAKE (Figure 1) is a mutually authenticated key exchange that comes in two versions: DAKE_2, a two-message variant achieving wPFS in the $\mathsf{eCK}^{\mathsf{w}}$ model, and DAKE_3, a three-message variant achieving PFS in the $\mathsf{eCK}\text{-}\mathsf{PFS}$ model.

Our analysis of DAKE accounts for an $\mathsf{eph}\text{-}\mathsf{key}$ query that can reveal:

- On a initiator session, the randomness used in $\mathsf{Keygen}_{\mathsf{R}}$ (which implies knowledge of sk_{R}). Moreover, $\widetilde{\mathsf{k}}$ can also be leaked after the session finished to compute its second message.
- On a responder session, nothing is revealed: we assume that the randomness of the encapsulation is securely erased immediately after its computation.

Two additional variants of DAKE are considered. The first, DAKE^{σ}, achieves responder authentication using a digital signature scheme instead of a KEM (Figure 16, Appendix D.1). While this typically increases overall communication bandwidth, it significantly reduces the size of the initiator's first message. The second, DUAKE, is a unilateral version providing authentication only for the initiator (Figure 2). Its security is proven under adapted unilateral models ($\mathsf{u}\text{-}\mathsf{eCK}^{\mathsf{w}}$ and $\mathsf{u}\text{-}\mathsf{eCK}\text{-}\mathsf{PFS}$), defined in Appendix C.4. Further details on both variants are provided in Appendices D.1 and D.2.

3.3 Security of Our AKE Protocols

We summarize here the security properties of DAKE, DAKE^{σ}, and DUAKE. Note that all of them are proven in the standard model.

Key Secrecy. It is achieved for DAKE, DAKE^{σ}, and DUAKE, assuming the $[\mathsf{IND}\text{-}\mathsf{CK}\text{-}\mathsf{CCA}, \mathsf{IND}\text{-}\mathsf{CCA}]$ security of the double-KEM and standard security assumptions on the other primitives. The 2-pass versions provide weak PFS, while the 3-pass versions achieve full PFS. The results for DAKE_3 are stated in Theorem 1, while those for DAKE_2, DAKE^{σ} and DUAKE are detailed in Appendix E.

In addition to the definitions provided in the main body, the theorem relies on standard security properties of a KEM, a PRG, and a MAC, which are recalled in Appendix A. We also use the *message–key bound* security of a MAC, which is ensured when a collision-resistant hash function is used as the underlying MAC primitive. This property is formalized by a game $\mathsf{MacColl}$, in which the adversary must find two distinct pairs of keys and messages, $(\mathsf{k}, m) \neq (\mathsf{k}', m')$, and a tag tag that is valid for both pairs.

[16] In contrast, protocols with key secrecy already provide UKS security for all attacks that fall within the allowed key-secrecy scenarios: otherwise, the adversary could violate key secrecy by creating a non-matching session s with the same key as the test session (thus breaking UKS), and then issuing a session-key query on s.

Theorem 1 (Key Secrecy of DAKE_3 in the Standard Model). *Let $\mathcal{A}$ be a PPT adversary against key secrecy of DAKE_3. Suppose it activates at most N parties and, for each party P, at most l sessions with actor or peer P. Assume that the underlying 2K-KEM is γ-spread and δ-correct while the standard KEM is δ'-correct. Let Ad^{w} denote the advantage of $\mathcal{A}$ against $\mathsf{eCK}^{\mathsf{w}}$-KeySec and $\mathsf{Ad}^{\mathsf{PFS}}$ its advantage against eCK-PFS-KeySec. Then, in the standard model, there exist PPT adversaries $\mathcal{B}_1, \cdots, \mathcal{B}_{11}$ such that:*

$$\mathsf{Ad}^{\mathsf{w}} \leq \alpha + 4\beta + 6Nl^2\Big(\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\,\text{-}\,]}_{\mathsf{dKEM}}(\mathcal{B}_6) + \mathsf{Adv}^{[\,\text{-}\,,\mathsf{IND\text{-}CCA}]}_{\mathsf{dKEM}}(\mathcal{B}_7) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{B}_8)\Big),$$
$$\mathsf{Ad}^{\mathsf{PFS}} \leq \mathsf{Adv}^{\mathsf{w}} + \beta + 2Nl^2\Big(2\,\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\,\text{-}\,]}_{\mathsf{dKEM}}(\mathcal{B}_9) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{B}_{10}) + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{B}_{11})\Big),$$

where $\alpha = \mathsf{Adv}^{\mathsf{MacColl}}_{\mathsf{MAC}}(\mathcal{B}_1) + N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\widetilde{\mathsf{KEM}}}(\mathcal{B}_2) + N^2 l\Big(2\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\widetilde{\mathsf{KEM}}}(\mathcal{B}_3) + l\delta' + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{B}_4)\Big)$ *and* $\beta = Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}_5) + 2Nl^2\,(\delta + 2^{-\gamma})$.

Moreover, (1) $\mathcal{B}_3$ queries $\mathcal{O}_{\mathsf{CCA}}$ at most l times, (2) $\mathcal{B}_4$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most l time. (3) $\mathcal{B}_6$ and $\mathcal{B}_9$ each query $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$ at most l times, (4) $\mathcal{B}_7$ does not query $\mathcal{O}_{\mathsf{leak_L}}$ and queries $\mathcal{O}_{\mathsf{CCA_R}}$ at most one time, and (5) $\mathcal{B}_{11}$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most one time.

Proof. The complete proof is given in Appendix E.[17] In this proof sketch, we provide the core ideas used to prove the key secrecy of DAKE_3. This explanation first covers key secrecy for Scenarios 1–4 (Table 4), and then Scenarios 5–6.

Key secrecy for Scenarios 1–4. These scenarios are proven via a game sequence that progressively replaces the test session key with a uniformly random value, thereby ensuring that no adversary can win the real-or-random game.

The crucial game of this sequence is the one that replaces the key k of the test session $\mathsf{s}^{\mathsf{Test}}$, as well as the key of the unique session that also encapsulates or decapsulates this key (if such a session exists), by a uniform key.

The change in the adversary's advantage introduced by this modification is bounded using one of the security properties of the 2K-KEM dKEM. We specify below which game is used, depending on the specific situation that can arise in Scenarios 1 to 4.

- *First possible situation: $\mathsf{s}^{\mathsf{Test}}$ is an initiator and* eph-key *is not used on it.* We rely on $[\,\text{-}\,,\mathsf{IND\text{-}CPA}]$, which informally guarantees that k is indistinguishable from uniform, provided that $\mathsf{sk_R}$ is unknown and that $(\mathsf{pk_L}, \mathsf{sk_L})$ has been honestly generated.
- *Second possible situation: $\mathsf{s}^{\mathsf{Test}}$ is an initiator and* corrupt *is not used on it.* We rely on [IND-CCA, -], which informally ensures that k is indistinguishable from uniform, provided that $\mathsf{sk_L}$ is unknown and that $(\mathsf{pk_R}, \mathsf{sk_R})$ has been honestly generated. Note that, unlike the previous case, CCA security is required here, since multiple sessions distinct from $\mathsf{s}^{\mathsf{Test}}$ may share the same long-term key $(\mathsf{pk_L}, \mathsf{sk_L})$. The decapsulation oracle is therefore needed to simulate decapsulations for these sessions without knowing $\mathsf{sk_L}$.
- *Third possible situation: $\mathsf{s}^{\mathsf{Test}}$ is a responder and* corrupt *is not used on its peer.* We need to rely on [IND-CK-CCA, -], which informally guarantees that k is indistinguishable from uniform provided that $\mathsf{sk_L}$ is unknown, even if $\mathsf{pk_R}$ is adversarially chosen. The chosen-key property is crucial here, since the ephemeral key $\mathsf{pk_R}$ received by $\mathsf{s}^{\mathsf{Test}}$ may be chosen by the adversary. Moreover, the CCA security is required for the same reason as previous item.
- *Fourth possible situation: $\mathsf{s}^{\mathsf{Test}}$ is a responder and* corrupt *is used on its peer.* This situation can only appear in scenario 3 or 4. Consequently, by definition of these scenarios, there exists an origin session s for $\mathsf{s}^{\mathsf{Test}}$, and eph-key has not been used on s. In particular, the ephemeral key $\mathsf{pk_R}$ received by $\mathsf{s}^{\mathsf{Test}}$ was honestly generated by s. In this case, we rely on the $[\,\text{-}\,,\mathsf{IND\text{-}CCA}]$ property, which informally guarantees that k is indistinguishable from a uniform key as long as $\mathsf{sk_R}$ remains unrevealed and $(\mathsf{pk_L}, \mathsf{sk_L})$ has been honestly generated. The CCA security is required in situations where the ciphertext ct received by s differs from the ciphertext created by $\mathsf{s}^{\mathsf{Test}}$; in such cases, the decapsulation oracle must be used on ct to compute the key k of s.

Key secrecy for Scenarios 5 and 6. We show that these scenarios can occur only with negligible probability – independently of whether the adversary won the Real-or-random game. More precisely, we first consider the execution only up to the moment when $\mathsf{s}^{\mathsf{Test}}$ accepts, which implies that we can assume its peer is not corrupted. We then prove that, except with negligible probability, there exists an origin session for $\mathsf{s}^{\mathsf{Test}}$, thereby contradicting both scenarios 5 and 6. We provide more details below, depending on the role of the test session.

[17] The complete proof is modular in order to simultaneously cover DAKE and DAKE^{σ}. In particular, the attacks to be considered are regrouped into a different set of scenarios than those of Table 4, leveraging the structural similarity between DAKE and DAKE^{σ} and allowing us to precisely isolate where the analyses diverge.

- *If* $\mathsf{s}^{\mathsf{Test}}$ *is an initiator*, we use the security of the MAC MAC to show that $\mathsf{tag}^{\mathsf{R}}$ must have been honestly computed by a responder session s. From the content of the message authenticated by this MAC, we can then deduce that s is in fact an origin session of $\mathsf{s}^{\mathsf{Test}}$. Note that, to invoke the MAC security, we first rely on the $\mathsf{IND\text{-}CCA}$ security of the KEM KEM to ensure that the key $\widetilde{\mathsf{k}}$ generated by $\mathsf{s}^{\mathsf{Test}}$ is indistinguishable from a uniform key. The CCA security is necessary to compute the decapsulations performed by the responder sessions whose actor is $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$.
- *If* $\mathsf{s}^{\mathsf{Test}}$ *is a responder*, we use the security of the MAC to show that $\mathsf{tag}^{\mathsf{I}}$ must have been honestly computed by an initiator session s. As in the previous case, the content of the authenticated message allows us to deduce that s is an origin session of $\mathsf{s}^{\mathsf{Test}}$. Note that to apply the MAC security, we first rely on the $[\mathsf{IND\text{-}CK\text{-}CCA}, \text{-}]$ security of the $\mathsf{2K\text{-}KEM}$ dKEM and then on the security of PRG to ensure that the key $\mathsf{k}^{\mathsf{MAC}}$ created by $\mathsf{s}^{\mathsf{Test}}$ is indistinguishable from uniform. The use of $[\mathsf{IND\text{-}CK\text{-}CCA}, \text{-}]$ is justified in the same way as in the analysis of the third possible situation, in the previous paragraph explaining the proof for Scenarios 1 to 4.

Match Soundness. It is achieved for DAKE, DAKE^{σ}, and DUAKE. Indeed, the first condition is a direct consequence of the correctness of the used primitives. Concerning the second condition, note that:
- Two distinct initiator sessions can only derive the same key pk_{R} with negligible probability. The contrary would directly lead to an attack against the right-key-recovery game $[\cdot, \mathsf{Key\text{-}Recov}]$ of the $\mathsf{2K\text{-}KEM}$.
- Spreadness of the $\mathsf{2K\text{-}KEM}$ implies that two responder sessions will compute the same encapsulation ct with negligible probability.

Inverse-Match Soundness. It is not guaranteed by default in DAKE, DAKE^{σ}, or DUAKE, but it can be enforced in a straightforward manner by deriving the session key through a collision-resistant key derivation function (KDF). Specifically, each protocol sets the session key as $\mathsf{KDF}(\mathsf{k}^{\mathsf{SESSION}}, \tau)$, where $\mathsf{k}^{\mathsf{SESSION}}$ is the initial session key, and τ is the concatenation of the message transcript and the identities of the actor and its peer.

The advantage of the collision-resistance experiment of a KDF against an adversary $\mathcal{A}$ is defined as[18]

$$\Pr[(\mathsf{k},\mathsf{m}) \neq (\mathsf{k}',\mathsf{m}') \wedge \mathsf{KDF}(\mathsf{k},\mathsf{m}) = \mathsf{KDF}(\mathsf{k}',\mathsf{m}') : (\mathsf{k},\mathsf{k}',\mathsf{m},\mathsf{m}') \leftarrow \mathcal{A}(\mathsf{pp})] \,.$$

We chose not to include this KDF natively, both for simplicity of the protocol description and because it is only required for this specific property.

Explicit Authentication. It is fully achieved by DAKE_3 and DAKE_3^{σ}, and partially achieved by our other protocols. More precisely:
- Init-explicit authentication is achieved by DAKE_3, DAKE_3^{σ} and DUAKE_3.
- Resp-explicit authentication holds for DAKE_2, DAKE_3, DAKE_2^{σ}, and DAKE_3^{σ}.

Theorem 2 provides precise security bounds for explicit authentication in DAKE. All remaining bounds, along with their corresponding proofs, are deferred to Appendix F.

The arguments establishing explicit authentication for our protocols closely follow those used to prove key secrecy in attack scenarios 5 and 6 of Table 4.[19] The core idea is that the security of the MAC (and of the signature in the case of DAKE^{σ}) guarantees that the tag (or signature) received by a session s was honestly generated by another session s'. The elements authenticated by this tag (or signature) then allow concluding that s' satisfies all the requirements of the explicit authentication property. In particular, to ensure Init-explicit authentication, the message used to compute $\mathsf{tag}^{\mathsf{I}}$ in all our three-message AKEs is defined to include both the transcript of exchanged messages and the identities of the actor and its peer. If Init-explicit authentication is not required, this message can simply be set to any fixed string.

Theorem 2 (Explicit Authentication for DAKE). *Let $\mathcal{A}$ be a PPT adversary against* $\mathsf{Init\text{-}ExplicitAuth}$ *for* DAKE_3*. Suppose it activates at most N parties and, for each party P, at most l sessions with actor or peer P. Assume that the underlying* $\mathsf{2K\text{-}KEM}$ *is γ-spread and δ-correct. Then, in the standard model, there exist PPT adversaries $\mathcal{B},\mathcal{C},\mathcal{D}$ and $\mathcal{E}$ such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Init\text{-}ExplicitAuth}}_{\mathsf{DAKE}_3}(\mathcal{A}) \leq\ & Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}) + 2Nl^2\left(\delta + 2^{-\gamma}\right) \\ & + 2Nl^2\left(2\,\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\text{-}]}_{\mathsf{dKEM}}(\mathcal{C}) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D}) + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{E})\right) .\end{aligned}$$

[18] Further discussions about this notion are given in Appendix A.

[19] They are described in the proof sketch of Theorem 1 for the case of DAKE_3.

Moreover, $\mathcal{C}$ queries $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$ at most l times, and $\mathcal{E}$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most one time.

Now let $\Pi \in \{\mathsf{DAKE}_2, \mathsf{DAKE}_3\}$ and suppose that $\mathcal{A}$ is a PPT adversary against Resp-ExplicitAuth *for Π. Assume the same conditions on $\mathcal{A}$ as above, and that the standard* KEM *is δ'-correct. Then, in the standard model, there exist PPT adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}$ such that $\mathsf{Adv}^{\mathsf{Resp\text{-}ExplicitAuth}}_{\Pi}(\mathcal{A})$ is upper-bounded by:*

$$\mathsf{Adv}^{\mathsf{MacColl}}_{\mathsf{MAC}}(\mathcal{B}) + N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\widetilde{\mathsf{KEM}}}(\mathcal{C}) + N^2 l\Big(2\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\widetilde{\mathsf{KEM}}}(\mathcal{D}) + l\delta' + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{E})\Big)\,.$$

Moreover, $\mathcal{D}$ queries $\mathcal{O}_{\mathsf{CCA}}$ at most l times and $\mathcal{E}$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most l times.

4 Chosen-key Fujisaki–Okamoto Transform

In this section, we introduce and analyze a new *chosen-key* Fujisaki–Okamoto transform (CK-FO) which upgrades a double-key, double-message IND-CPA PKE into an IND-CCA double-KEM that additionally provides left chosen-key security. Concretely, in the corresponding security game the adversary may freely choose the (possibly malformed) right public key. Section 4.1 defines the class of 2K2M-PKE schemes compatible with our CK-FO, obtained by combining two PKE instances, while Section 4.2 presents the transform and its security proof.

4.1 Combining Two PKE to Obtain a 2K2M-PKE

We describe a generic method for combining two PKE schemes that share common randomness into a 2K2M-PKE. This construction produces shorter ciphertexts than running the original PKEs independently, while yielding exactly the input format required by our CK-FO transform. Informally, we consider two PKE schemes whose ciphertexts share a common component $\mathsf{c_C}$, and our construction keeps just one $\mathsf{c_C}$ instead of keeping it in both ciphertexts.

Concretely, let two PKE schemes $\mathsf{PKE_L} = (\mathsf{Setup}, \mathsf{Keygen_L}, \mathsf{Encr_L}, \mathsf{Decr_L})$ and $\mathsf{PKE_R} = (\mathsf{Setup}, \mathsf{Keygen_R}, \mathsf{Encr_R}, \mathsf{Decr_R})$ share the same setup algorithm and satisfy the following conditions:

- The randomness spaces are structured as $\mathsf{Rand_C} \times \mathsf{Rand_L}$ for $\mathsf{Encr_L}$ and $\mathsf{Rand_C} \times \mathsf{Rand_R}$ for $\mathsf{Encr_R}$, with $\mathsf{Rand_C}$ the common part.
- There exists a deterministic algorithm E_{base} that, given $(\mathsf{pp}, r_\mathsf{C})$ with public parameters pp and $r_\mathsf{C} \in \mathsf{Rand_C}$, outputs $\mathsf{c_C}$ such that both $\mathsf{Encr_L}$ and $\mathsf{Encr_R}$ produce ciphertexts whose first component equals $\mathsf{c_C} = E_{\mathsf{base}}(\mathsf{pp}, r_\mathsf{C})$.

Under these conditions, encryptions using the same r_C share the component $\mathsf{c_C}$, which can be factored out to improve compactness. This yields the combined scheme $\mathsf{DPKE} = (\mathsf{dSetup} = \mathsf{Setup}, \mathsf{dKeygenL} = \mathsf{Keygen_L}, \mathsf{dKeygenR} = \mathsf{Keygen_R}, \mathsf{dEnc}, \mathsf{dDec})$, with algorithms dEnc and dDec given in Figure 5.

It is important to note that IND-CPA security of $\mathsf{PKE_L}$ and $\mathsf{PKE_R}$ does not automatically imply any security for DPKE, since the two encryptions are no longer independent. The security of DPKE must therefore be established separately, on a case-by-case basis. In Section 5, we present $\mathsf{Maul}_{\mathsf{PKE}}$ as an example of this transformation, whose security can be proven under lattice assumptions.

$\mathsf{dEnc}(\mathsf{pp}, \mathsf{pk_L}, \mathsf{pk_R}, \mathsf{m} = (m_\mathsf{L}, m_\mathsf{R}))$	$\mathsf{dDec}(\mathsf{pp}, \mathsf{sk_L}, \mathsf{sk_R}, \mathsf{ct} = (\mathsf{c_L}, \mathsf{c_R}))$
$r_\mathsf{C} \leftarrow\!\!\$\ \mathsf{Rand_C}$	$m_\mathsf{L} = \mathsf{Decr_L}(\mathsf{pp}, \mathsf{sk_L}, \mathsf{c_L})$
$r_\mathsf{L} \leftarrow\!\!\$\ \mathsf{Rand_L}$	$m_\mathsf{R} = \mathsf{Decr_R}(\mathsf{pp}, \mathsf{sk_R}, \mathsf{c_R})$
$r_\mathsf{R} \leftarrow\!\!\$\ \mathsf{Rand_R}$	**if** $m_\mathsf{L} = \perp \vee m_\mathsf{R} = \perp$ **then**
// Here, $\mathsf{c_C} = E_{\mathsf{base}}(\mathsf{pp}, r_\mathsf{C})$	**return** $\perp$
$(\mathsf{c_C}, \mathsf{c_L}) = \mathsf{Encr_L}(\mathsf{pp}, \mathsf{pk_L}, m_\mathsf{L}; (r_\mathsf{C}, r_\mathsf{L}))$	**return** $\mathsf{m} = (m_\mathsf{L}, m_\mathsf{R})$
$(\mathsf{c_C}, \mathsf{c_R}) = \mathsf{Encr_R}(\mathsf{pp}, \mathsf{pk_R}, m_\mathsf{R}; (r_\mathsf{C}, r_\mathsf{R}))$	
return $(\mathsf{c_C}, \mathsf{c_L}, \mathsf{c_R})$	

Fig. 5: Algorithms dEnc and dDec of the transformation described in Section 4.1.

4.2 The CK-FO Transform and Its Security

Our CK-FO transform takes as input a family of hash functions $(\mathsf{H}_{\mathsf{rej}}, \mathsf{H}_{\mathsf{key}}, \mathsf{H}_{\mathsf{rand}}, \mathsf{H}_{\mathsf{OTP}})$ together with a 2K2M-PKE DPKE constructed from two PKE schemes $\mathsf{PKE}_{\mathsf{L}}$ and $\mathsf{PKE}_{\mathsf{R}}$, as detailed in Section 4.1. The transform outputs a 2K-KEM, denoted $\mathsf{dKEM}^{\mathsf{FO}}$, described in Figure 6.

The CK-FO transform upgrades the [IND-CPA, IND-CPA] security of DPKE into the $\big[$IND-CK-CCA, IND-CCA$\big]$ security of the resulting 2K-KEM.

A central difficulty in proving [IND-CK-CCA, -] security is that the adversary may choose the right public key pk_{R}, which need not be well-formed. Since the 2K2M-PKE is only [IND-CPA, -]-secure, we can guarantee the security of c_{R} only when pk_{R} is generated by $\mathsf{Keygen}_{\mathsf{R}}$. Otherwise, c_{R} might leak information about both its encapsulated message m_{R} and the common randomness r_{C}, potentially revealing the message m_{L} encapsulated in c_{L}.

Our transform resolves this by concealing c_{R} using m_{L} (line 6 of $\mathsf{dEncaps}^{\mathsf{FO}}$ in Figure 6).[20] This ensures c_{R} remains hidden until m_{L} is recovered, allowing us to reduce the [IND-CK-CCA, -] security of $\mathsf{dKEM}^{\mathsf{FO}}$ to the IND-CPA security of $\mathsf{PKE}_{\mathsf{L}}$. The proof therefore relies fundamentally on the structure of the input 2K2M-PKE and, in particular, on the security of its left constituent PKE $\mathsf{PKE}_{\mathsf{L}}$.

To obtain security in the QROM rather than only in the ROM, we must carefully select which elements are hashed. In particular, the encapsulation randomness is not derived solely from the messages m_{L} and m_{R}, but also incorporates the two public keys, as shown in the algorithm $\mathsf{dEncaps}^{\mathsf{FO}}$ in Figure 6. This choice is essential for the QROM proof, which uses semi-classical oracles [3], following the approach of [23].

$\mathsf{dSetup}^{\mathsf{FO}}()$

$\mathsf{pp} \leftarrow \mathsf{Setup}()$
return pp

$\mathsf{dKeygen}_{\mathsf{L}}^{\mathsf{FO}}(\mathsf{pp})$

$\mathsf{s}_{\mathsf{L}} \leftarrow\!\$\ \{0,1\}^{\nu}$
$(\mathsf{pk}_{\mathsf{L}}, \mathsf{sk}_{\mathsf{L}}) \leftarrow \mathsf{Keygen}_{\mathsf{L}}(\mathsf{pp})$
$\mathsf{sk}_{\mathsf{L}}^{\mathsf{FO}} = (\mathsf{sk}_{\mathsf{L}}, \mathsf{s}_{\mathsf{L}})$
return $(\mathsf{pk}_{\mathsf{L}}, \mathsf{sk}_{\mathsf{L}}^{\mathsf{FO}})$

$\mathsf{dKeygen}_{\mathsf{R}}^{\mathsf{FO}}(\mathsf{pp})$

$\mathsf{s}_{\mathsf{R}} \leftarrow\!\$\ \{0,1\}^{\nu}$
$(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}}) \leftarrow \mathsf{Keygen}_{\mathsf{R}}(\mathsf{pp})$
$\mathsf{sk}_{\mathsf{R}}^{\mathsf{FO}} = (\mathsf{sk}_{\mathsf{R}}, \mathsf{s}_{\mathsf{R}})$
return $(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}}^{\mathsf{FO}})$

$\mathsf{dEncaps}^{\mathsf{FO}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}})$

$m_{\mathsf{L}} \leftarrow\!\$\ \mathcal{M}_{\mathsf{L}}$
$m_{\mathsf{R}} \leftarrow\!\$\ \mathcal{M}_{\mathsf{R}}$
$(r_{\mathsf{C}}, r_{\mathsf{L}}, r_{\mathsf{R}}) \leftarrow \mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}})$
$(\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}) = \mathsf{Encr}_{\mathsf{L}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L}}, m_{\mathsf{L}}; r_{\mathsf{C}}, r_{\mathsf{L}})$
$(\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{R}}) = \mathsf{Encr}_{\mathsf{R}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{R}}; r_{\mathsf{C}}, r_{\mathsf{R}})$
$c_{\mathsf{R}}^{\mathsf{OTP}} \leftarrow \mathsf{c}_{\mathsf{R}} \oplus \mathsf{H}_{\mathsf{OTP}}(m_{\mathsf{L}})$
$\mathsf{k} \leftarrow \mathsf{H}_{\mathsf{key}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c_{\mathsf{R}}^{\mathsf{OTP}})$
return $(\mathsf{k}, (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c_{\mathsf{R}}^{\mathsf{OTP}}))$

$\mathsf{dDecaps}^{\mathsf{FO}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{L}}^{\mathsf{FO}} = (\mathsf{sk}_{\mathsf{L}}, \mathsf{s}_{\mathsf{L}}), \mathsf{sk}_{\mathsf{R}}^{\mathsf{FO}} = (\mathsf{sk}_{\mathsf{R}}, \mathsf{s}_{\mathsf{R}}), (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c_{\mathsf{R}}^{\mathsf{OTP}}))$

$\mathsf{k}' \leftarrow \mathsf{H}_{\mathsf{rej}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, \mathsf{s}_{\mathsf{L}}, \mathsf{s}_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c_{\mathsf{R}}^{\mathsf{OTP}})$
$m'_{\mathsf{L}} \leftarrow \mathsf{Decr}_{\mathsf{L}}(\mathsf{pp}, \mathsf{sk}_{\mathsf{L}}, (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}))$
if $m'_{\mathsf{L}} = \perp$ **then return** k'
$\mathsf{c}_{\mathsf{R}} \leftarrow c_{\mathsf{R}}^{\mathsf{OTP}} \oplus \mathsf{H}_{\mathsf{OTP}}(m'_{\mathsf{L}})$
$m'_{\mathsf{R}} \leftarrow \mathsf{Decr}_{\mathsf{R}}(\mathsf{pp}, \mathsf{sk}_{\mathsf{R}}, (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{R}}))$
if $m'_{\mathsf{R}} = \perp$ **then return** k'
$(r'_{\mathsf{C}}, r'_{\mathsf{L}}, r'_{\mathsf{R}}) \leftarrow \mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m'_{\mathsf{L}}, m'_{\mathsf{R}})$
if $\mathsf{Encr}_{\mathsf{L}}(\mathsf{pk}_{\mathsf{L}}, m'_{\mathsf{L}}; r'_{\mathsf{C}}, r'_{\mathsf{L}}) \neq (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}})$ **then return** k'
if $\mathsf{Encr}_{\mathsf{R}}(\mathsf{pk}_{\mathsf{R}}, m'_{\mathsf{R}}; r'_{\mathsf{C}}, r'_{\mathsf{R}}) \neq (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{R}})$ **then return** k'
return $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m'_{\mathsf{L}}, m'_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c_{\mathsf{R}}^{\mathsf{OTP}})$

Fig. 6: Our CK-FO Transform. $(\mathsf{dSetup}, \mathsf{dKeygenL}, \mathsf{dKeygenR}, \mathsf{dEnc}, \mathsf{dDec})$ is a 2K2M-PKE built from two PKEs $\mathsf{PKE}_{\mathsf{L}}$ and $\mathsf{PKE}_{\mathsf{R}}$, as described in Section 4.1.

The following theorem summarizes the security guarantees of our CK-FO transform in both the ROM and the QROM. The corresponding proofs are provided in Appendix B.

Theorem 3 (CK-FO transform in ROM and QROM). *Consider random oracles* $\mathsf{H}_{\mathsf{key}}$, $\mathsf{H}_{\mathsf{rej}}$, $\mathsf{H}_{\mathsf{OTP}}$, *and* $\mathsf{H}_{\mathsf{rand}}$, *and let* DPKE *constructed from* $\mathsf{PKE}_{\mathsf{L}}$ *and* $\mathsf{PKE}_{\mathsf{R}}$ *be* δ*-correct. Let* $\mathcal{A}$ *be an adversary against*

[20] Note that the xor is well-defined: we assume that the set of right ciphertexts is contained in the binary space $\{0,1\}^{\tau}$ and choose $\mathsf{H}_{\mathsf{OTP}}$ to have range $\{0,1\}^{\tau}$.

the [IND-CK-CCA, -] *(resp.* [-, IND-CCA]*) security of* $\mathsf{dKEM}^{\mathsf{FO}}$*, making at most* $q_{\mathsf{leak}}, q_{\mathsf{CCA}}, q_{\mathsf{key}}, q_{\mathsf{rej}}, q_{\mathsf{OTP}}$ *and* q_{rand} *queries to the oracles* $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$ *(resp.* $\mathcal{O}_{\mathsf{leak}_\mathsf{L}}$*),* $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ *(resp.* $\mathcal{O}_{\mathsf{CCA}_\mathsf{R}}$*),* $\mathsf{H}_{\mathsf{key}}$*,* $\mathsf{H}_{\mathsf{rej}}$*,* $\mathsf{H}_{\mathsf{OTP}}$ *and* $\mathsf{H}_{\mathsf{rand}}$*. Then there exists an adversary* $\mathcal{B}$ *such that:*

$$\mathsf{Adv}^{[\text{IND-CK-CCA}, -]}_{\mathsf{dKEM}^{\mathsf{FO}}}(\mathcal{A}) \leq 2\,\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{PKE}_\mathsf{L}}(\mathcal{B}) + \frac{q_{\mathsf{rej}}}{2^\nu} + q_{\mathsf{rand}}\delta + \frac{q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}}}{|\mathcal{M}_\mathsf{L}|}\,,$$
$$\mathsf{Adv}^{[-, \text{IND-CCA}]}_{\mathsf{dKEM}^{\mathsf{FO}}}(\mathcal{A}) \leq 2\,\mathsf{Adv}^{[-, \text{IND-CPA}]}_{\mathsf{DPKE}}(\mathcal{B}) + \frac{q_{\mathsf{rej}}}{2^\nu} + q_{\mathsf{rand}}\delta + \frac{q_{\mathsf{key}}}{|\mathcal{M}_\mathsf{R}|}\,.$$

If we operate in the quantum-accessible random oracle model with quantum-accessible oracles $\mathsf{H}_{\mathsf{key}}$*,* $\mathsf{H}_{\mathsf{rej}}$*,* $\mathsf{H}_{\mathsf{OTP}}$*, and* $\mathsf{H}_{\mathsf{rand}}$*, and* q_{key}*,* q_{rej}*,* q_{OTP}*,* q_{rand} *denote the respective query bounds, then there exists an adversary* $\mathcal{B}$ *such that:*

$$\mathsf{Adv}^{[\text{IND-CK-CCA}, -]}_{\mathsf{dKEM}^{\mathsf{FO}}}(\mathcal{A}) \leq \beta + 2\sqrt{(\hat{q}+1)\left(2\,\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{PKE}_\mathsf{L}}(\mathcal{B}) + 4\hat{q}/|\mathcal{M}_\mathsf{L}|\right)},$$
$$\mathsf{Adv}^{[-, \text{IND-CCA}]}_{\mathsf{dKEM}^{\mathsf{FO}}}(\mathcal{A}) \leq \widetilde{\beta} + 2\sqrt{(\tilde{q}+1)\left(2\,\mathsf{Adv}^{[-, \text{IND-CPA}]}_{\mathsf{DPKE}}(\mathcal{B}) + 4\tilde{q}/|\mathcal{M}_\mathsf{R}|\right)},$$

where $\beta = 2\,q_{\mathsf{rej}}\,2^{-\nu/2} + 16\,q_{\mathsf{leak}}\,\delta\,(q_{\mathsf{rand}} + q_{\mathsf{CCA}} + 1)^2$*,* $\widetilde{\beta} = 2\,q_{\mathsf{rej}}\,2^{-\nu/2} + 16\,(q_{\mathsf{leak}}+1)\,\delta\,(q_{\mathsf{rand}} + q_{\mathsf{CCA}} + 1)^2$*,* $\hat{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}} + q_{\mathsf{CCA}}$ *and* $\tilde{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{CCA}}$*.*

In summary, [-, IND-CCA] security reduces to the [-, IND-CPA] security of DPKE, and [IND-CK-CCA, -] security reduces to the IND-CPA security of PKE_L.

5 Compact Lattice-based Double-KEM

In this section we introduce Maul, a compact 2K-KEM inspired by Kyber [48] and achieving $[\text{IND-CK-CCA}, \text{IND-CCA}]$ security. Its security is proven in both the ROM and QROM under the MLWE and Hint-MLWE assumptions. Maul is obtained by applying our CK-FO transform (Section 4) to $\mathsf{Maul}_{\mathsf{PKE}}$, a [IND-CPA, IND-CPA]-secure 2K2M-PKE.

After introducing the necessary preliminaries in Section 5.1, we present the 2K2M-PKE $\mathsf{Maul}_{\mathsf{PKE}}$ and establish its security in Section 5.2. Finally, the concrete parameters of Maul are provided in Section 5.3 and justified in Sections 5.4 and 5.5.

5.1 Preliminaries on Lattices, MLWE, and Hint-MLWE Problems

Linear Algebra and Modular Operations. Let A be a ring. Vectors $\mathbf{u} \in A^n$ are assumed to be column vectors, and we denote by $\mathbf{u}^\intercal$ the transpose of $\mathbf{u}$. For $x \in \mathbb{R}$, we denote by $\lfloor x \rceil$ the nearest integer to x, with ties rounded up. We define the polynomial ring $\mathcal{R} := \mathbb{Z}[X]/(X^n+1)$, where $n \in \mathbb{N}$ is called the *dimension*, to be specified as needed. For a modulus $q \in \mathbb{N}$, we define the quotient ring $\mathcal{R}_q := \mathbb{Z}_q[X]/(X^n+1)$. For any $q > 1$, we identify $\mathcal{R}_2$ with the subset of polynomials in $\mathcal{R}_q$ whose coefficients lie in $\{0,1\}$. For a real matrix $\mathbf{H} \in \mathbb{R}^{d\times\ell}$, we define the spectral norm $s_1(\mathbf{H})$ as $s_1(\mathbf{H}) = \max_{\mathbf{x}\neq 0} \frac{\|\mathbf{H}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$, where $\|\cdot\|_2$ denotes the euclidean norm. For $\mathbf{H} \in \mathcal{R}^{d\times\ell}$, we identify it with an associated negacyclic matrix $\bar{\mathbf{H}} \in \mathbb{Z}^{nd\times n\ell}$, and define the spectral norm of $\mathbf{H}$ as that of $\bar{\mathbf{H}}$.

Throughout this work, unless otherwise stated, all operations are performed modulo a positive integer $q \in \mathbb{N}$, which will be specified in context.

MLWE and Hint-MLWE Problems. We rely on the *module learning with errors* (MLWE) problem [41], as well as a generalized variant known as the Hint-MLWE problem [37], in which the adversary is additionally given partial information (or *hints*) about the MLWE secret and error vectors.

Definition 1 (MLWE). *Let* k, m, q *be positive integers, and let* χ *be a distribution over* $\mathcal{R}^{k+m}$*. The advantage of an adversary* $\mathcal{A}$ *against the module learning with errors problem* $\mathsf{MLWE}^{\chi}_{\mathcal{R},k,m,q}$ *is defined as:* $\left|\Pr[\mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s}+\mathbf{e}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) = 1]\right|$*, where* $\mathbf{A} \leftarrow\!\$\, \mathcal{U}\left(\mathcal{R}_q^{m\times k}\right)$*,* $\left(\begin{smallmatrix}\mathbf{s}\\ \mathbf{e}\end{smallmatrix}\right) \leftarrow\!\$\, \chi$ *and* $\mathbf{u} \leftarrow\!\$\, \mathcal{U}\left(\mathcal{R}_q^m\right)$*.*

In the original Hint-MLWE formulation [37, Definition 7], the adversary is given noisy linear hints of the form $\gamma_i \cdot \mathbf{r} + \mathbf{y}_i$, where $\mathbf{r} = \left(\begin{smallmatrix}\mathbf{s}\\ \mathbf{e}\end{smallmatrix}\right)$ is the concatenated secret-error vector, $\mathbf{y}_i$ are noise terms, and γ_i are public polynomials. This was later generalized in [24] to hints of the form $\mathbf{H}\mathbf{r} + \mathbf{y}$, where $\mathbf{H} \in \mathcal{R}_q^{\ell\times 2k}$ and $\mathbf{A} \in \mathcal{R}_q^{k\times k}$. We consider a slight extension of this generalization, where $\mathbf{A} \in \mathcal{R}_q^{m\times k}$ and $\mathbf{H} \in \mathcal{R}_q^{\ell\times d}$ for some $d \leq k+m$. This allows the hints to depend only on a subset of the secret-error vector.

Definition 2 (Hint-MLWE, adapted from [24]). *Let k, m, ℓ, d, q be positive integers with $d \leq k+m$, and let $J \subseteq \mathbb{Z} \cap [0, k+m-1]$ be an index set of size d. Let χ be a distribution over $\mathcal{R}^{k+m}$, χ_h a distribution over $\mathcal{R}^{\ell}$, and $\mathcal{F}$ a distribution over $\mathcal{R}_q^{\ell \times d}$.*

The advantage of an adversary $\mathcal{A}$ against the hint module learning with errors problem , denoted $\mathsf{Hint\text{-}MLWE}^{\chi,\chi_h,\mathcal{F}}_{\mathcal{R},k,m,q,\ell,d,J}$, is defined as

$$\left| \Pr[\mathcal{A}(\mathbf{A}, \mathbf{As}+\mathbf{e}, \mathbf{H}, \mathbf{Hr}+\mathbf{y}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{b}, \mathbf{H}, \mathbf{Hr}+\mathbf{y}) = 1] \right|,$$

where $(\mathbf{A}, \mathbf{b}, \bar{\mathbf{r}}, \mathbf{H}, \mathbf{y}) \leftarrow \mathcal{U}\left(\mathcal{R}_q^{m \times k}\right) \times \mathcal{U}\left(\mathcal{R}_q^{m}\right) \times \chi \times \mathcal{F} \times \chi_h$, with $\bar{\mathbf{r}} := \left(\begin{smallmatrix}\mathbf{s}\\ \mathbf{e}\end{smallmatrix}\right) \in \mathcal{R}^{k+m}$, and where $\mathbf{r} = (\bar{\mathbf{r}}_i)_{i \in J} \in \mathcal{R}^d$ consists of the components of $\bar{\mathbf{r}}$ indexed by J.

In [37], it is shown that the Hint-MLWE problem is at least as hard as the MLWE problem when the secret and error are sampled from discrete Gaussian distributions, assuming certain constraints on the width parameters. For completeness, we present an adaptation of this reduction that applies to the generalized Hint-MLWE formulation in Definition 2, under the same parameter regime. It is proven in Appendix H.1.

Theorem 4 (Hardness of Hint-MLWE, adapted from [37]).
Let k, m, ℓ, d, q, J be as in Definition 2, and let $\mathcal{F}$ be a distribution over

$$\mathcal{S}_B := \left\{ \mathbf{H} \in \mathcal{R}_q^{\ell \times d} \;\middle|\; s_1(\mathbf{H})^2 \leq B \right\},$$

for $B > 0$. Let $\sigma_1, \sigma_2 > 0$, and define $\sigma > 0$ via $\frac{1}{\sigma^2} = 2\left(\frac{1}{\sigma_1^2} + \frac{B}{\sigma_2^2}\right)$. Let $\chi_h := D_{\mathbb{Z}^{\ell \cdot n}, \sigma_2}$, and let χ be a distribution over $\mathcal{R}^{k+m-d}$. Define χ_σ over $\mathcal{R}^{k+m}$ as the distribution where coordinates in J are drawn from $D_{\mathbb{Z}^{d \cdot n}, \sigma}$ and the rest from χ.

If $\sigma \geq \sqrt{2} \cdot \eta_\varepsilon(\mathbb{Z}^{d \cdot n})$ for some $0 < \varepsilon \leq 1/2$, then there exists an efficient reduction from $\mathsf{MLWE}^{\chi_\sigma}_{\mathcal{R},k,m,q}$ to $\mathsf{Hint\text{-}MLWE}^{\chi_{\sigma_1},\chi_h,\mathcal{F}}_{\mathcal{R},k,m,q,\ell,d,J}$ that reduces the advantage by at most 2ε.

5.2 $\mathsf{Maul}_{\mathsf{PKE}}$: A [IND-CPA, IND-CPA]-Secure 2K2M-PKE

Let $k \in \mathbb{N}$, two distributions $\mathcal{B}, \mathcal{D}$ over $\mathcal{R}$, and compression parameters $d_u, d_v \leq \log_2(q)$. For $x \in \mathbb{Q}$, we define the compression $\mathsf{Comp}_{(q,d_u)}(x) = \lfloor (2^{d_u} x)/q \rceil \bmod 2^{d_u}$, and the decompression $\mathsf{Decomp}_{(q,d_v)}(x) = \lfloor (qx)/2^{d_v} \rceil$.

Consider the Kyber-like PKE DualRegev defined in Figure 7. Its encryption algorithm uses three sources of randomness $(\mathbf{s}_\mathsf{C}, \mathbf{e}_\mathsf{C}, f)$, producing ciphertexts $(\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{scal})$, where c_C depends only on $(\mathbf{s}_\mathsf{C}, \mathbf{e}_\mathsf{C})$. By applying the transformation from Section 4.1, we combine two instances of DualRegev into a 2K2M-PKE that reuses the randomness pair $(\mathbf{s}_\mathsf{C}, \mathbf{e}_\mathsf{C})$. The resulting scheme, $\mathsf{Maul}_{\mathsf{PKE}}$, is detailed in Figure 8.

Theorem 5 analyses the [IND-CPA, -] and [- , IND-CPA] security of $\mathsf{Maul}_{\mathsf{PKE}}$. Its correctness is analyzed in the following lemma, proven in Appendix H.2.

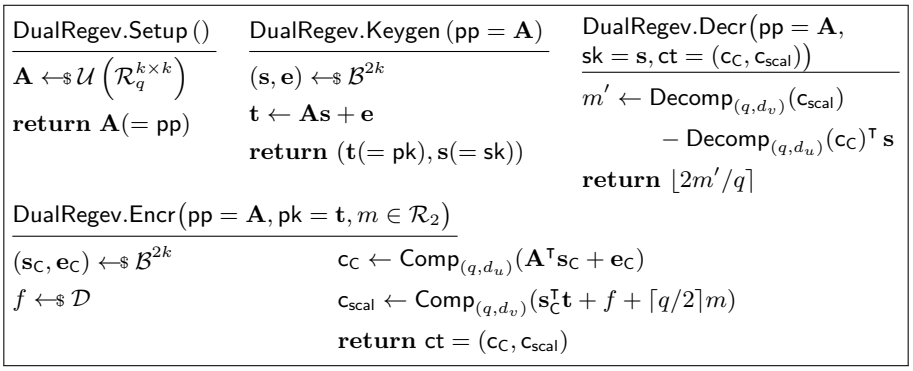

Fig. 7: Algorithms for DualRegev.

$\mathsf{Maul}_{\mathsf{PKE}}.\mathsf{Setup}\,()$

$\mathbf{A} \leftarrow\!\$\; \mathcal{U}\left(\mathcal{R}_q^{k\times k}\right)$

return $\mathbf{A}(=\mathsf{pp})$

$\mathsf{Maul}_{\mathsf{PKE}}.\mathsf{KeygenL}\,(\mathsf{pp}=\mathbf{A})$

$(\mathbf{s}_\mathsf{L}, \mathbf{e}_\mathsf{L}) \leftarrow\!\$\; \mathcal{B}^{2k}$

$\mathbf{t}_\mathsf{L} \leftarrow \mathbf{A}\mathbf{s}_\mathsf{L} + \mathbf{e}_\mathsf{L}$

return $(\mathsf{pk}_\mathsf{L} = \mathbf{t}_\mathsf{L}, \mathsf{sk}_\mathsf{L} = \mathbf{s}_\mathsf{L})$

$\mathsf{Maul}_{\mathsf{PKE}}.\mathsf{KeygenR}\,(\mathsf{pp}=\mathbf{A})$

$(\mathbf{s}_\mathsf{R}, \mathbf{e}_\mathsf{R}) \leftarrow\!\$\; \mathcal{B}^{2k}$

$\mathbf{t}_\mathsf{R} \leftarrow \mathbf{A}\mathbf{s}_\mathsf{R} + \mathbf{e}_\mathsf{R}$

return $(\mathsf{pk}_\mathsf{R} = \mathbf{t}_\mathsf{R}, \mathsf{sk}_\mathsf{R} = \mathbf{s}_\mathsf{R})$

$\mathsf{Maul}_{\mathsf{PKE}}.\mathsf{Encr}\big(\mathsf{pp}=\mathbf{A}, \mathsf{pk}_\mathsf{L}=\mathbf{t}_\mathsf{L}, \mathsf{pk}_\mathsf{R}=\mathbf{t}_\mathsf{R}, (m_\mathsf{L}, m_\mathsf{R}) \in \mathcal{R}_2^2\big)$

$(\mathbf{s}_\mathsf{C}, \mathbf{e}_\mathsf{C}) \leftarrow\!\$\; \mathcal{B}^{2k}$

$(f_\mathsf{L}, f_\mathsf{R}) \leftarrow\!\$\; \mathcal{D}^{2k}$

$\mathsf{c}_\mathsf{C} \leftarrow \mathsf{Comp}_{(q,d_u)}(\mathbf{A}^\intercal\mathbf{s}_\mathsf{C} + \mathbf{e}_\mathsf{C})$

$\mathsf{c}_\mathsf{L} \leftarrow \mathsf{Comp}_{(q,d_v)}(\mathbf{s}_\mathsf{C}^\intercal\mathbf{t}_\mathsf{L} + f_\mathsf{L} + \lceil q/2\rceil m_\mathsf{L})$

$\mathsf{c}_\mathsf{R} \leftarrow \mathsf{Comp}_{(q,d_v)}(\mathbf{s}_\mathsf{C}^\intercal\mathbf{t}_\mathsf{R} + f_\mathsf{R} + \lceil q/2\rceil m_\mathsf{R})$

return $\mathsf{ct} = (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, \mathsf{c}_\mathsf{R})$

$\mathsf{Maul}_{\mathsf{PKE}}.\mathsf{Decr}\big(\mathsf{pp}=\mathbf{A}, \mathsf{sk}_\mathsf{L}=\mathbf{s}_\mathsf{L}, \mathsf{sk}_\mathsf{R}=\mathbf{s}_\mathsf{R}, \mathsf{ct}=(\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, \mathsf{c}_\mathsf{R})\big)$

$m'_\mathsf{L} \leftarrow \mathsf{Decomp}_{(q,d_v)}(\mathsf{c}_\mathsf{L}) - \mathsf{Decomp}_{(q,d_u)}(\mathsf{c}_\mathsf{C})^\intercal\,\mathbf{s}_\mathsf{L}$

$m'_\mathsf{R} \leftarrow \mathsf{Decomp}_{(q,d_v)}(\mathsf{c}_\mathsf{R}) - \mathsf{Decomp}_{(q,d_u)}(\mathsf{c}_\mathsf{C})^\intercal\,\mathbf{s}_\mathsf{R}$

return $(\lfloor 2\,m'_\mathsf{L}/q\rceil, \lfloor 2\,m'_\mathsf{R}/q\rceil)$

Fig. 8: Algorithms for $\mathsf{Maul}_{\mathsf{PKE}}$.

Lemma 1 ($\mathsf{Maul}_{\mathsf{PKE}}$ correctness). *Using the notation from Figure 8, the scheme* $\mathsf{Maul}_{\mathsf{PKE}}$ *is* δ*-correct provided that*

$$\Pr\left[\left\|\begin{pmatrix} f_\mathsf{L} + e_{\mathsf{c}_\mathsf{L}} + \mathbf{s}_\mathsf{C}^\intercal\mathbf{e}_\mathsf{L} - (\mathbf{e}_\mathsf{C} + \mathbf{e}_{\mathsf{c}_\mathsf{C}})^\intercal\mathbf{s}_\mathsf{L} \\ f_\mathsf{R} + e_{\mathsf{c}_\mathsf{R}} + \mathbf{s}_\mathsf{C}^\intercal\mathbf{e}_\mathsf{R} - (\mathbf{e}_\mathsf{C} + \mathbf{e}_{\mathsf{c}_\mathsf{C}})^\intercal\mathbf{s}_\mathsf{R} \end{pmatrix}\right\|_\infty \le \frac{q}{4}\right] > 1-\delta\,,$$

where, $\mathbf{e}_{\mathsf{c}_\mathsf{C}} = \mathsf{Decomp}_{(q,d_u)}(\mathsf{c}_\mathsf{C}) - \mathbf{A}^\intercal\mathbf{s}_\mathsf{C} + \mathbf{e}_\mathsf{C}$*,* $e_{\mathsf{c}_\mathsf{L}} = \mathsf{Decomp}_{(q,d_v)}(\mathsf{c}_\mathsf{L}) - \mathbf{s}_\mathsf{C}^\intercal\mathbf{t}_\mathsf{L} + f_\mathsf{L} + m_\mathsf{L}$*, and* $e_{\mathsf{c}_\mathsf{R}} = \mathsf{Decomp}_{(q,d_v)}(\mathsf{c}_\mathsf{R}) - \mathbf{s}_\mathsf{C}^\intercal\mathbf{t}_\mathsf{R} + f_\mathsf{R} + m_\mathsf{R}$*.*

Theorem 5 ([IND-CPA, IND-CPA] Security of $\mathsf{Maul}_{\mathsf{PKE}}$). *Let consider* $\mathsf{X} \in \{[\mathsf{IND\text{-}CPA}, \text{-}], [\text{-}, \mathsf{IND\text{-}CPA}]\}$ *and* $\mathcal{A}$ *a PPT adversary of the* X *security of* $\mathsf{Maul}_{\mathsf{PKE}}$*. There exist PPT adversaries* $\mathcal{B}, \mathcal{C}$ *with same time complexity as* $\mathcal{A}$ *such that:*

$$\mathsf{Adv}^{\mathsf{X}}_{\mathsf{Maul}_{\mathsf{PKE}}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{MLWE}_{k,k,\mathcal{B}^k,\mathcal{B}^k}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}^{\chi,\chi_h,\mathcal{F}}_{\mathcal{R},k,k+1,q,1,2k,J}}(\mathcal{C}).$$

Where $\mathcal{F} = \mathcal{B}^k \times (-\mathcal{B}^k)$*,* $\chi_h = \mathcal{D}$*,* $\chi = \mathcal{B}^{2k} \times \mathcal{D}$ *and the subset* J *being the* $2k$ *first elements.*

Proof. The complete proof is provided in Appendix H.2. We briefly outline here the argument for [IND-CPA, -] security; the proof for [- , IND-CPA] follows symmetrically. Moreover, it suffices to consider the variant of the scheme in which the functions Compress and Decompress are replaced by the identity.

We aim to show that the left ciphertext $c_{\mathsf{L},b}$ given to the adversary $\mathcal{A}$ is indistinguishable from uniform. Consequently, $\mathcal{A}$ cannot determine which left message was encrypted.

For this, we first introduce a game G_1 in which the vector $\mathbf{t}_\mathsf{L}$ is sampled uniformly at random, and the vectors $\mathbf{e}_\mathsf{L}$ and $\mathbf{s}_\mathsf{L}$ are no longer used. The difference in the adversary's advantage between the two games is then bounded by the MLWE advantage specified in the theorem statement.

Using the notation $\mathbf{B}^\intercal = (\mathbf{A} \mid \mathbf{t}_\mathsf{L})$, we can express the pair $(\mathsf{c}_\mathsf{C}, \tilde{\mathsf{c}_\mathsf{L}})$ as $(\mathsf{c}_\mathsf{C}^\intercal, \tilde{\mathsf{c}_\mathsf{L}})^\intercal = \mathbf{B}\mathbf{s}_\mathsf{C} + (\mathbf{e}_\mathsf{C}^\intercal, f_\mathsf{L})^\intercal$. In G_1, we know that $\mathbf{B}$ is uniform. We would thus like to invoke the MLWE assumption to argue that $(\mathsf{c}_\mathsf{C}^\intercal, \tilde{\mathsf{c}_\mathsf{L}})^\intercal$ is indistinguishable from uniform. However, the secret $\mathbf{s}_\mathsf{C}$ also appears in another term: $\mathsf{c}_\mathsf{R} = \mathbf{s}_\mathsf{C}^\intercal\mathbf{t}_\mathsf{R} + f_\mathsf{R} + \lceil q/2\rceil m_\mathsf{R}$, which prevents a direct application of the MLWE hypothesis. To address this, we instead rely on the Hint-MLWE hypothesis.

5.3 Parameter Selection

In this section, we propose concrete parameters for Maul, our 2K-KEM construction, targeting λ-bit [IND-CK-CCA, IND-CCA] security for $\lambda \in \{128, 192, 256\}$, corresponding respectively to NIST security levels 1, 3, and 5. These parameters are selected with consideration of the security implications of the Hint-MLWE assumption, on which the scheme ultimately relies.

Table 5 summarizes the parameter sets for each target security level. As shown in Appendix H.3, instantiating Maul with Kyber's original parameters yielded a decapsulation failure probability δ above

our design target, mainly due to the large variance of the distribution $\mathcal{D}$. To restore the intended failure probability, we slightly adjusted the parameters. For each set, we report the estimated BKZ block size (bikz) for solving the underlying Hint-MLWE problem, the ring dimension n, key dimension k, modulus q (NTT-friendly), compression parameters (d_u, d_v), and the distributions $(\mathcal{B}, \mathcal{D})$ used in key generation and encapsulation. Here, $\mathcal{B}_\eta$ denotes the centered binomial distribution with parameter η as defined in [52], and $\mathcal{D}_\sigma$ a *Gaussian-like* distribution of standard deviation σ, encompassing centered binomials, sums of uniforms, and discrete Gaussians. We also provide the public key and ciphertext sizes (in bytes). Estimates for bikz and δ were obtained using a modified version of the scripts[21].

Table 5: Parameter sets for Maul, including sizes (in bytes).

	bikz	n	k	q	$\mathcal{B}$	$\mathcal{D}$	(d_u, d_v)	δ	ν	|pk|	|ct|	|pk| + |ct|
Maul512	337	256	2	7681	$\mathcal{B}_4$	$\mathcal{D}_{64}$	$(10, 4)$	2^{-150}	384	826	896	1722
Maul768	557	256	3	7681	$\mathcal{B}_4$	$\mathcal{D}_{64}$	$(11, 6)$	2^{-196}	512	1240	1440	2680
Maul1024	766	256	4	9473	$\mathcal{B}_4$	$\mathcal{D}_{64}$	$(12, 5)$	2^{-257}	640	1691	1856	3547

5.4 Security Estimation

Our security analysis is grounded in the assumed hardness of the MLWE problem, assessed via the *core-SVP* cost model [1]. This model estimates security levels against both classical and quantum attacks by relating them to the required BKZ block size (bikz) needed to solve the underlying MLWE instance. A conservative estimate is then derived by applying the complexity of the best known sieving algorithms—both classical and quantum [6,14]—to this block size, deliberately omitting polynomial factors in the full BKZ complexity. These omitted factors typically account for around 30 bits of security.

Transitioning from Hint-MLWE to standard MLWE for parameter estimation involves subtleties, as the distributions and parameter regimes fall outside Theorem 4 reduction. To address this, we follow the heuristic approach of Dodis et al. [21], which guides our parameter selection and hardness estimation for Hint-MLWE. Specifically, we rely on the following three heuristics:

Heuristic 1: Replacing Gaussian Distributions. Although Theorem 4 formally applies to discrete Gaussians with widths above the smoothing parameter, we assume similar hardness for non-Gaussian distributions of equal variance. This is justified by their *Gaussian-like* behavior for large parameters and consistent performance in Rényi-divergence analyses.

Heuristic 2: Removing the Factor 2. We omit the factor 2 in the expression $1/\sigma^2 = 2 \cdot (1/\sigma_1^2 + s_1(\mathbf{H})^2/\sigma_2^2)$ originally introduced in the Hint-MLWE reduction. As noted in [21], this factor arises from smoothing arguments specific to discrete Gaussians and is irrelevant for non-Gaussian distributions. Under this simplification, the relation between the equivalent Gaussian parameters becomes $1/\sigma^2 = 1/\sigma_1^2 + s_1(\mathbf{H})^2/\sigma_2^2$.

Heuristic 3: Approximate Spectral Norm. The security reduction involves the spectral norm $s_1(\mathbf{H})$ of the hint-related matrix $\mathbf{H} = (\mathbf{e}_\mathsf{L}^\intercal, -\mathbf{s}_\mathsf{L}^\intercal)$. Instead of bounding this norm in the worst-case, we adopt the average-case approximation, yielding that $s_1(\mathbf{H})^2$ is replaced by $n \cdot \sigma_1^2$. This estimation gives us a baseline to assess the hardness of Hint-MLWE, with $\sigma_2 = \sqrt{n}\,\sigma_1^2$, as that of the MLWE problem, with $\sigma = \sigma_1/\sqrt{2}$.

5.5 Concrete Security Loss

We emphasise that potential security loss arising from the reduction from MLWE to Hint-MLWE is not formally captured in the above heuristic approach. In particular, it assumes that the standard deviation of the involved distributions exceeds the smoothing parameter, which ensures a tight reduction. Since the distributions used in Maul fall below this threshold, some degree of security degradation may be expected. Nevertheless, we conjecture that even under these conditions, the security of Hint-MLWE instantiated with parameters σ_1 and $\sigma_2 = \sqrt{n}\,\sigma_1^2$ remains higher than that of MLWE instantiated with parameter $\sigma_1/\sqrt{2}$. To support this conjecture, we conduct empirical analyses and propose a concrete attack against Maul that explicitly exploits the hint. Further details are provided in Appendix H.3.

[21] `https://github.com/pq-crystals/security-estimates`

LWE with Side Information [18]. The most effective known attack against Hint-MLWE for Maul parameters relies on the framework of Ducas et al. [18], which extends the classical primal attack to incorporate auxiliary information. The standard primal attack on MLWE (typically disregarding the module structure) interprets, via a suitable embedding, the LWE instance as a *bounded distance decoding* (BDD) problem, which is then solved using lattice reduction techniques. The framework of [18] extends this approach to non-spherical (ellipsoidal) distributions and settings with side information that alters the target distribution.

Specifically, *approximate hints* arise naturally in the Maul setting when an adversary, given $(\mathbf{e}_{\mathsf{L}}, \mathbf{s}_{\mathsf{L}})$, attempts to recover the encapsulated message m_{R} without access to $(\mathbf{e}_{\mathsf{R}}, \mathbf{s}_{\mathsf{R}})$. [22] These hints take the form of noisy linear relations

$$\mathbf{z} = \mathbf{H}\begin{pmatrix}\mathbf{s}\\ \mathbf{e}\end{pmatrix} + \mathbf{y},$$

which matches the structure of the Hint-MLWE hint in Maul, with $\mathbf{H} = (\mathbf{e}_{\mathsf{L}}^{\mathsf{T}}, -\mathbf{s}_{\mathsf{L}}^{\mathsf{T}})$, $(\mathbf{s}, \mathbf{e}) = (\mathbf{s}_{\mathsf{C}}, \mathbf{e}_{\mathsf{C}})$, and $\mathbf{y} = f_{\mathsf{L}}$.

The attack proceeds by conditioning the joint prior distribution of $(\mathbf{s}, \mathbf{e})$ on the observed hint $\mathbf{z}$. By Lemma 6 of [18], the resulting conditional distribution remains Gaussian, with covariance matrix Σ_{Hints}, but is generally no longer spherical. To compensate for this anisotropy, a linear transformation is applied to whiten the covariance, thereby reducing the instance to an equivalent spherical BDD problem amenable to standard lattice reduction techniques.

Residual security is estimated by assuming that the vectors $\mathbf{s}$, $\mathbf{e}$, and $\mathbf{y}$ follow continuous Gaussian distributions with variances matching those of their actual distributions. This approximation enables the use of the leaky-LWE estimator from [18][23] to evaluate the impact of approximate hints.

Table 6: Estimated security ($\mathsf{PrimalLWE}_{\Sigma_{\mathsf{Hints}}}$) in bikz for Maul parameters, including upper bound ($\mathsf{PrimalLWE}_{\sigma_1}$), and lower bound ($\mathsf{PrimalLWE}_{\sigma_1/\sqrt{2}}$).

	$\mathsf{PrimalLWE}_{\sigma_1}$	$\mathsf{PrimalLWE}_{\Sigma_{\mathsf{Hints}}}$	$\mathsf{PrimalLWE}_{\sigma_1/\sqrt{2}}$
Maul512	373	353	338
Maul768	613	589	560
Maul1024	839	811	772

Table 6 reports the leaky-LWE estimated security (bikz) for Maul, including an upper bound, and a lower bound, under the following models:

- $\mathsf{PrimalLWE}_{\sigma_1}$: the estimated security of a standard (M)LWE instance with secret and error distributions of standard deviation σ_1; this serves as an upper bound on the security of Hint-MLWE with parameters $(\sigma_1, \sigma_2 = \sqrt{n}\,\sigma_1^2)$,
- $\mathsf{PrimalLWE}_{\Sigma_{\mathsf{Hints}}}$: the residual security after incorporating approximate hints to (M)LWE with parameter σ_1, using the leaky-LWE estimator,
- $\mathsf{PrimalLWE}_{\sigma_1/\sqrt{2}}$: the estimated security of a standard (M)LWE instance with standard deviation $\sigma_1/\sqrt{2}$;this models an instance that is at least as hard as the corresponding Hint-MLWE instance under the Heuristics 2-3.

Figure 9 shows that approximate hints weaken security, though less than the Theorem 4 reduction suggests, indicating the correctness of the Heuristics 2-3 and also that more competitive Hint-MLWE parameters may be viable. All the scripts used for security estimate and size comparison are available at `https://github.com/tlegavre/dake_estimator`.

[22] This is the only attack scenario in which the adversary encounters a Hint-MLWE instance; in the symmetric left-side case, the one-time pad introduced by the CK-FO transform prevents such an attack.

[23] `https://github.com/lducas/leaky-LWE-Estimator`

[24] For $k = 2$, $n = 256$, and $q = 7681$, as in the Maul512 parameter set.

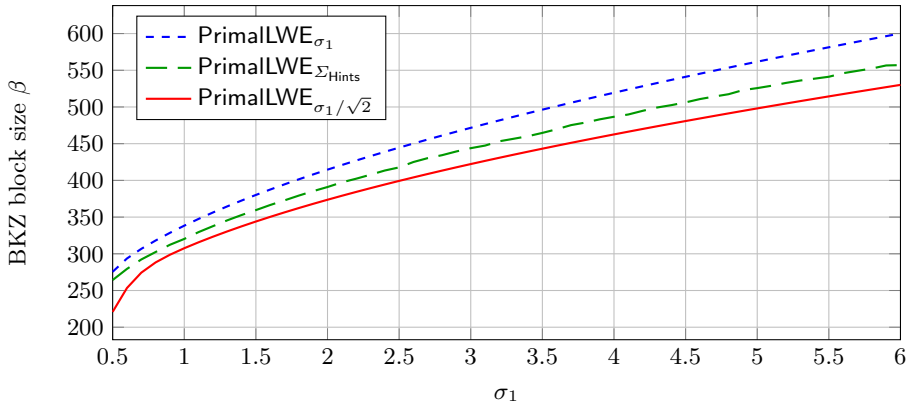

Fig. 9: Estimated security[24] (PrimalLWE$_{\Sigma_{\text{Hints}}}$) in bikz as a function of σ_1, including upper bound (PrimalLWE$_{\sigma_1}$), and lower bound (PrimalLWE$_{\sigma_1/\sqrt{2}}$).

References

1. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. In: Holz, T., Savage, S. (eds.) USENIX Security 2016: 25th USENIX Security Symposium. pp. 327–343. USENIX Association, Austin, TX, USA (Aug 10–12, 2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim
2. Alwen, J., Coretti, S., Dodis, Y.: The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019, Part I. Lecture Notes in Computer Science, vol. 11476, pp. 129–158. Springer, Cham, Switzerland, Darmstadt, Germany (May 19–23, 2019). https://doi.org/10.1007/978-3-030-17653-2_5
3. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology – CRYPTO 2019, Part II. Lecture Notes in Computer Science, vol. 11693, pp. 269–295. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26951-7_10
4. Apple Security Engineering and Architecture (SEAR): iMessage with PQ3: The new state of the art in quantum-secure messaging at scale (February 2024), https://security.apple.com/blog/imessage-pq3
5. Barnes, R., Bhargavan, K., Lipp, B., Wood, C.A.: Hybrid Public Key Encryption. RFC 9180 (Feb 2022). https://doi.org/10.17487/RFC9180, https://www.rfc-editor.org/info/rfc9180
6. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: Krauthgamer, R. (ed.) 27th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 10–24. ACM-SIAM, Arlington, VA, USA (Jan 10–12, 2016). https://doi.org/10.1137/1.9781611974331.ch2
7. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) Advances in Cryptology – EUROCRYPT 2000. Lecture Notes in Computer Science, vol. 1807, pp. 139–155. Springer Berlin Heidelberg, Germany, Bruges, Belgium (May 14–18, 2000). https://doi.org/10.1007/3-540-45539-6_11
8. Bindel, N., Hamburg, M., Hövelmanns, K., Hülsing, A., Persichetti, E.: Tighter proofs of CCA security in the quantum random oracle model. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019: 17th Theory of Cryptography Conference, Part II. Lecture Notes in Computer Science, vol. 11892, pp. 61–90. Springer, Cham, Switzerland, Nuremberg, Germany (Dec 1–5, 2019). https://doi.org/10.1007/978-3-030-36033-7_3
9. Blake-Wilson, S., Menezes, A.: Unknown key-share attacks on the station-to-station (STS) protocol. In: Imai, H., Zheng, Y. (eds.) PKC'99: 2nd International Workshop on Theory and Practice in Public Key Cryptography. Lecture Notes in Computer Science, vol. 1560, pp. 154–170. Springer Berlin Heidelberg, Germany, Kamakura, Japan (Mar 1–3, 1999). https://doi.org/10.1007/3-540-49162-7_12
10. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy. pp. 353–367. IEEE Computer Society Press, London, United Kingdom (Apr 24–26, 2018). https://doi.org/10.1109/EuroSP.2018.00032
11. Boyd, C., Cliff, Y., González Nieto, J., Paterson, K.G.: Efficient one-round key exchange in the standard model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 08: 13th Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 5107, pp. 69–83. Springer Berlin Heidelberg, Germany, Wollongong, Australia (Jul 7–9, 2008). https://doi.org/10.1007/978-3-540-70500-0_6
12. Boyd, C., de Kock, B., Millerjord, L.: Modular design of KEM-based authenticated key exchange. In: Simpson, L., Baee, M.A.R. (eds.) ACISP 23: 28th Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 13915, pp. 553–579. Springer, Cham, Switzerland, Brisbane, QLD, Australia (Jul 5–7, 2023). https://doi.org/10.1007/978-3-031-35486-1_24

13. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) Advances in Cryptology – EUROCRYPT 2001. Lecture Notes in Computer Science, vol. 2045, pp. 453–474. Springer Berlin Heidelberg, Germany, Innsbruck, Austria (May 6–10, 2001). https://doi.org/10.1007/3-540-44987-6_28
14. Chailloux, A., Loyer, J.: Lattice sieving via quantum random walks. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021, Part IV. Lecture Notes in Computer Science, vol. 13093, pp. 63–91. Springer, Cham, Switzerland, Singapore (Dec 6–10, 2021). https://doi.org/10.1007/978-3-030-92068-5_3
15. Cremers, C., Düzlü, S., Fiedler, R., Fischlin, M., Janson, C.: BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In: 2021 IEEE Symposium on Security and Privacy. pp. 1696–1714. IEEE Computer Society Press, San Francisco, CA, USA (May 24–27, 2021). https://doi.org/10.1109/SP40001.2021.00093
16. Cremers, C.J.F.: Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS authenticated key exchange protocol. In: Abdalla, M., Pointcheval, D., Fouque, P.A., Vergnaud, D. (eds.) ACNS 2009: 7th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 5536, pp. 20–33. Springer Berlin Heidelberg, Germany, Paris-Rocquencourt, France (Jun 2–5, 2009). https://doi.org/10.1007/978-3-642-01957-9_2
17. Cremers, C.J.F., Feltz, M.: Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012: 17th European Symposium on Research in Computer Security. Lecture Notes in Computer Science, vol. 7459, pp. 734–751. Springer Berlin Heidelberg, Germany, Pisa, Italy (Sep 10–12, 2012). https://doi.org/10.1007/978-3-642-33167-1_42
18. Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: Attacks and concrete security estimation. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 329–358. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56880-1_12
19. D'Anvers, J.P., Batsleer, S.: Multitarget decryption failure attacks and their application to Saber and Kyber. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 13177, pp. 3–33. Springer, Cham, Switzerland, Virtual Event (Mar 8–11, 2022). https://doi.org/10.1007/978-3-030-97121-2_1
20. Dodis, Y., Fiore, D.: Unilaterally-authenticated key exchange. In: Kiayias, A. (ed.) FC 2017: 21st International Conference on Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 10322, pp. 542–560. Springer, Cham, Switzerland, Sliema, Malta (Apr 3–7, 2017). https://doi.org/10.1007/978-3-319-70972-7_31
21. Dodis, Y., Jost, D., Katsumata, S., Prest, T., Schmidt, R.: Triple ratchet: A bandwidth efficient hybrid-secure signal protocol. In: Fehr, S., Fouque, P.A. (eds.) Advances in Cryptology – EUROCRYPT 2025, Part VIII. Lecture Notes in Computer Science, vol. 15608, pp. 302–331. Springer, Cham, Switzerland, Madrid, Spain (May 4–8, 2025). https://doi.org/10.1007/978-3-031-91101-9_11
22. Dowling, B., Hale, B.: Authenticated continuous key agreement: Active MitM detection and prevention. Cryptology ePrint Archive, Report 2023/228 (2023), https://eprint.iacr.org/2023/228
23. Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G.: Faster lattice-based KEMs via a generic Fujisaki-Okamoto transform using prefix hashing. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021: 28th Conference on Computer and Communications Security. pp. 2722–2737. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). https://doi.org/10.1145/3460120.3484819
24. Esgin, M.F., Espitau, T., Niot, G., Prest, T., Sakzad, A., Steinfeld, R.: Plover: Masking-friendly hash-and-sign lattice signatures. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024, Part VII. Lecture Notes in Computer Science, vol. 14657, pp. 316–345. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58754-2_12
25. Fiore, D., Gennaro, R., Smart, N.P.: Constructing certificateless encryption and ID-based encryption from ID-based key agreement. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) PAIRING 2010: 4th International Conference on Pairing-based Cryptography. Lecture Notes in Computer Science, vol. 6487, pp. 167–186. Springer Berlin Heidelberg, Germany, Yamanaka Hot Spring, Japan (Dec 13–15, 2010). https://doi.org/10.1007/978-3-642-17455-1_11
26. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-Fourier lattice-based compact signatures over NTRU. Submission to the NIST's post-quantum cryptography standardization process (2020)
27. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography. Lecture Notes in Computer Science, vol. 7293, pp. 467–484. Springer Berlin Heidelberg, Germany, Darmstadt, Germany (May 21–23, 2012). https://doi.org/10.1007/978-3-642-30057-8_28
28. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. Designs, Codes and Cryptography **76**(3), 469–504 (2015). https://doi.org/10.1007/s10623-014-9972-2

29. Gaborit, P., Aguilar-Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Persichetti, E., Zémor, G., Bos, J., Dion, A., Lacan, J., Robert, J.M., Véron, P., Barreto, P.L., Ghosh, S., Gueron, S., Güneysu, T., Misoczki, R., Richter-Brokmann, J., Sendrier, N., Tillich, J.P., Vasseur, V.: Hamming Quasi-Cyclic (HQC). Submission to the NIST's post-quantum cryptography standardization process (2025)
30. Goldberg, I., Stebila, D., Ustaoglu, B.: Anonymity and one-way authentication in key exchange protocols. Designs, Codes and Cryptography **67**(2), 245–269 (2013). https://doi.org/10.1007/s10623-011-9604-z
31. Hashimoto, K., Katsumata, S., Niot, G., Wiggers, T.: Revisiting PQ WireGuard: A comprehensive security analysis with a new design using Reinforced KEMs p. To Appear (2026), https://eprint.iacr.org/2025/1758
32. Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II. Lecture Notes in Computer Science, vol. 12111, pp. 389–422. Springer, Cham, Switzerland, Edinburgh, UK (May 4–7, 2020). https://doi.org/10.1007/978-3-030-45388-6_14
33. Hülsing, A., Ning, K.C., Schwabe, P., Weber, F.J., Zimmermann, P.R.: Post-quantum WireGuard. In: 2021 IEEE Symposium on Security and Privacy. pp. 304–321. IEEE Computer Society Press, San Francisco, CA, USA (May 24–27, 2021). https://doi.org/10.1109/SP40001.2021.00030
34. Jacobs, F.: Invited talk: Designing imessage pq3: Quantum-secure messaging at scale. Talk given at RWC 2024 (2024), video at https://www.youtube.com/watch?v=RVbHElGe518
35. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. Cryptology ePrint Archive, Report 2011/219 (2011), https://eprint.iacr.org/2011/219
36. Katsumata, S., Kwiatkowski, K., Pintore, F., Prest, T.: Scalable ciphertext compression techniques for post-quantum KEMs and their applications. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020, Part I. Lecture Notes in Computer Science, vol. 12491, pp. 289–320. Springer, Cham, Switzerland, Daejeon, South Korea (Dec 7–11, 2020). https://doi.org/10.1007/978-3-030-64837-4_10
37. Kim, D., Lee, D., Seo, J., Song, Y.: Toward practical lattice-based proof of knowledge from hint-MLWE. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part V. Lecture Notes in Computer Science, vol. 14085, pp. 549–580. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38554-4_18
38. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) Advances in Cryptology – CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621, pp. 546–566. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2005). https://doi.org/10.1007/11535218_33
39. Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: A systematic analysis. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 429–448. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40041-4_24
40. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007: 1st International Conference on Provable Security. Lecture Notes in Computer Science, vol. 4784, pp. 1–16. Springer Berlin Heidelberg, Germany, Wollongong, Australia (Nov 1–2, 2007). https://doi.org/10.1007/978-3-540-75670-5_1
41. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Des. Codes Cryptogr. **75**(3), 565–599 (2015). https://doi.org/10.1007/S10623-014-9938-4, https://doi.org/10.1007/s10623-014-9938-4
42. Lyu, Y., Liu, S.: Two-message authenticated key exchange from public-key encryption. In: Tsudik, G., Conti, M., Liang, K., Smaragdakis, G. (eds.) ESORICS 2023: 28th European Symposium on Research in Computer Security, Part I. Lecture Notes in Computer Science, vol. 14344, pp. 414–434. Springer, Cham, Switzerland, The Hague, The Netherlands (Sep 25–29, 2023). https://doi.org/10.1007/978-3-031-50594-2_21
43. Naehrig, M., Alkim, E., Bos, J., Ducas, L., Easterbrook, K., LaMacchia, B., Longa, P., Mironov, I., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: FrodoKEM. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions
44. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Rabin, T. (ed.) Advances in Cryptology – CRYPTO 2010. Lecture Notes in Computer Science, vol. 6223, pp. 80–97. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2010). https://doi.org/10.1007/978-3-642-14623-7_5
45. Prest, T.: Sharper bounds in lattice-based cryptography using the Rényi divergence. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 347–374. Springer, Cham, Switzerland, Hong Kong, China (Dec 3–7, 2017). https://doi.org/10.1007/978-3-319-70694-8_13
46. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Aug 2018). https://doi.org/10.17487/RFC8446, https://www.rfc-editor.org/info/rfc8446
47. Rogaway, P.: Formalizing human ignorance. In: Nguyen, P.Q. (ed.) Progress in Cryptology - VIETCRYPT 06: 1st International Conference on Cryptology in Vietnam. Lecture Notes in Computer Science, vol. 4341, pp. 211–228. Springer Berlin Heidelberg, Germany, Hanoi, Vietnam (Sep 25–28, 2006). https://doi.org/10.1007/11958239_14

48. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Seiler, G., Stehlé, D., Ding, J.: CRYSTALS-KYBER. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
49. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020: 27th Conference on Computer and Communications Security. pp. 1461–1480. ACM Press, Virtual Event, USA (Nov 9–13, 2020). https://doi.org/10.1145/3372297.3423350
50. Schwabe, P., Stebila, D., Wiggers, T.: More efficient post-quantum KEMTLS with pre-distributed public keys. In: Bertino, E., Shulman, H., Waidner, M. (eds.) ESORICS 2021: 26th European Symposium on Research in Computer Security, Part I. Lecture Notes in Computer Science, vol. 12972, pp. 3–22. Springer, Cham, Switzerland, Darmstadt, Germany (Oct 4–8, 2021). https://doi.org/10.1007/978-3-030-88418-5_1
51. Shoup, V.: On formal models for secure key exchange. Cryptology ePrint Archive, Report 1999/012 (1999), https://eprint.iacr.org/1999/012
52. of Standards, N.I., Technology: Fips 203: Module-lattice-based key-encapsulation mechanism standard, https://csrc.nist.gov/pubs/fips/203/final
53. of Standards, N.I., Technology: Fips 204: Module-lattice-based digital signature standard, https://csrc.nist.gov/pubs/fips/204/final
54. of Standards, N.I., Technology: Fips 205: Stateless hash-based digital signature standard, https://csrc.nist.gov/pubs/fips/205/final
55. Wiggers, T., Celi, S., Schwabe, P., Stebila, D., Sullivan, N.: KEM-based Authentication for TLS 1.3. Internet-Draft draft-celi-wiggers-tls-authkem-05, Internet Engineering Task Force (Apr 2025), https://datatracker.ietf.org/doc/draft-celi-wiggers-tls-authkem/05/, work in Progress
56. Wiggers, T., Celi, S., Schwabe, P., Stebila, D., Sullivan, N.: KEM-based Authentication for TLS 1.3. Internet-Draft draft-celi-wiggers-tls-authkem-05, Internet Engineering Task Force (Apr 2025), https://datatracker.ietf.org/doc/draft-celi-wiggers-tls-authkem/05/, work in Progress
57. Xue, H., Lu, X., Li, B., Liang, B., He, J.: Understanding and constructing AKE via double-key key encapsulation mechanism. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018, Part II. Lecture Notes in Computer Science, vol. 11273, pp. 158–189. Springer, Cham, Switzerland, Brisbane, Queensland, Australia (Dec 2–6, 2018). https://doi.org/10.1007/978-3-030-03329-3_6

A Additional Preliminaries

Additional Notations We denote by ϵ the empty string in $\{0,1\}^*$. For any non-zero integer $i \in \mathbb{N}$, we define $\mathsf{Range}(i) := [0, i-1] \cap \mathbb{Z}$.

Public Key Encryption. A public key encryption (PKE) scheme $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ consists of four probabilistic polynomial-time (PPT) algorithms. The $\mathsf{Setup}()$ algorithm generates public parameters pp. Using these parameters, $\mathsf{Keygen}(\mathsf{pp})$ produces a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$. The $\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m)$ algorithm encrypts a message m from the message space $\mathcal{M}$ into a ciphertext ct, while $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$ decrypts the ciphertext ct back into the message m. The IND-CPA security of a PKE scheme is defined through the following game, where the adversary's advantage is given by $|\Pr[b = b'] - 1/2|$.

IND-CPA

$\mathsf{pp} \leftarrow \mathsf{Setup}()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(\mathsf{pp})$
$(\mathsf{state}, m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk})$
$\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m_0)$
$\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m_1)$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow \mathcal{A}_2(\mathsf{state}, \mathsf{ct}_b)$
return $b = b'$

Key Encapsulation Mechanism. A key encapsulation mechanism (KEM) scheme $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Encaps}, \mathsf{Decaps})$ consists of four probabilistic polynomial-time (PPT) algorithms. The $\mathsf{Setup}()$ algorithm generates public parameters pp. Using these parameters, $\mathsf{Keygen}(\mathsf{pp})$ produces a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$. The $\mathsf{Encaps}(\mathsf{pp}, \mathsf{pk})$ algorithm creates a couple $(\mathsf{ct}, \mathsf{k})$: k is a key belonging to a key space Keys and ct is its encapsulation. $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$ decrypts ct back into key k. The IND-CCA security of a KEM scheme is defined through the following game, where the adversary's advantage is given by $|\Pr[b = b'] - 1/2|$.

IND-CCA($\mathcal{A}$)	$\mathcal{O}_{\mathsf{CCA}}(\mathsf{ct}')$
$b \leftarrow\!\$\ \{0,1\}$	**if** $\mathsf{ct}' \neq \mathsf{ct}$ **then**
$\mathsf{pp} \leftarrow \mathsf{Setup}()$	$\mathsf{k}' \leftarrow \mathsf{Decaps}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct}')$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(\mathsf{pp})$	**return** k'
$\mathsf{k}_0 \leftarrow\!\$\ \mathsf{Keys}$	**return** $\perp$
$(\mathsf{ct}, \mathsf{k}_1) \leftarrow \mathsf{Encaps}(\mathsf{pp}, \mathsf{pk})$	
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}}(\mathsf{pp}, \mathsf{pk}, \mathsf{ct}, \mathsf{k}_b)$	
return $b = b'$	

For $\delta \geq 0$, a KEM scheme is said δ-correct if for $\mathsf{pp} \leftarrow \mathsf{Setup}()$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(\mathsf{pp})$, $(\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Encaps}(\mathsf{pp}, \mathsf{pk})$, the probability to have $\mathsf{Decaps}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct}) = \mathsf{k}$ is more than $1 - \delta$.

We also consider the *key-recovery game* Key-Recov. First, we suppose the existence of a function RecovPKL, such that, for any $\mathsf{pp} \leftarrow \mathsf{Setup}()$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}()$, we have $\mathsf{pk} = \mathsf{RecovPK}(\mathsf{pp}, \mathsf{sk})$. The advantage of the game Key-Recov is defined as:

$$\Pr\left[\mathsf{pk} = \mathsf{RecovPK}(\mathsf{pp}, \mathsf{sk}') : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(\mathsf{pp}), \\ \mathsf{sk} \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}) \end{array}\right].$$

Signature. A signature scheme $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignVf})$ consists of four probabilistic polynomial-time (PPT) algorithms. The $\mathsf{Setup}()$ algorithm generates public parameters pp. Using these parameters, $\mathsf{Keygen}(\mathsf{pp})$ produces a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$. The $\mathsf{Sign}(\mathsf{sk}, \mathsf{m})$ create a signature σ of a message m belonging to the message space $\mathcal{M}$. $\mathsf{SignVf}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}, \sigma)$ returns 1 (the signature is valid) or 0 (the signature is invalid).

The *strong unforgeability under chosen message attack* (sUF-CMA) security of a signature scheme is defined through the game sUF-CMA of Figure 10, where the adversary's advantage is given by the probability $\Pr[\mathsf{sUF\text{-}CMA}(\mathcal{A}) = 1]$.

sUF-CMA($\mathcal{A}$)	MBS($\mathcal{A}$)
$Q \leftarrow \emptyset$	$\mathsf{pp} \leftarrow \mathsf{Setup}()$
$\mathsf{pp} \leftarrow \mathsf{Setup}()$	$(\mathsf{m}_1, \mathsf{m}_2, \sigma, \mathsf{pk}) \leftarrow \mathcal{A}(\mathsf{pp})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(\mathsf{pp})$	**if** $\mathsf{m}_1 \neq \mathsf{m}_2$
$(\mathsf{m}', \sigma') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{pk})$	$\wedge\ \mathsf{SignVf}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}_1, \sigma) = 1$
if $(\mathsf{m}', \sigma') \notin Q$ **then**	$\wedge\ \mathsf{SignVf}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}_2, \sigma) = 1$ **then**
return $\mathsf{SignVf}(\mathsf{pp}, \mathsf{pk}, \mathsf{m}, \sigma)$	**return** 1
return 0	**return** 0
$\mathcal{O}_{\mathsf{Sign}}(m)$	M-S-UEO($\mathcal{A}$)
$\mathsf{m} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m})$	$\mathsf{pp} \leftarrow \mathsf{Setup}()$
$Q \leftarrow Q \cup \{(\mathsf{m}, \sigma)\}$	$(\mathsf{m}_1, \mathsf{m}_2, \sigma, \mathsf{pk}_1, \mathsf{pk}_2) \leftarrow \mathcal{A}(\mathsf{pp})$
return σ	**if** $\mathsf{pk}_1 \neq \mathsf{pk}_2$
	$\wedge\ \mathsf{SignVf}(\mathsf{pp}, \mathsf{pk}_1, \mathsf{m}_1, \sigma) = 1$
	$\wedge\ \mathsf{SignVf}(\mathsf{pp}, \mathsf{pk}_2, \mathsf{m}_2, \sigma) = 1$ **then**
	return 1
	return 0

Fig. 10: Security games of a signature scheme.

The key-recovery game Key-Recov of a signature scheme is defined identically to the recovery game of a KEM. The *message-bound signature* (MBS) security of a signature scheme, explained in [15], is defined through the game MBS of Figure 10, where the adversary's advantage is given by the probability $\Pr[\mathsf{MBS}(\mathcal{A}) = 1]$. The *malicious-strong universal exclusive ownership* (M-S-UEO) security of a signature

scheme, explained in [15], is defined through the game M-S-UEO of Figure 10, where the adversary's advantage is given by the probability $\Pr[\mathsf{M\text{-}S\text{-}UEO}(\mathcal{A}) = 1]$.

Message Authentication Code. A message authentication code (MAC) scheme $(\mathsf{Mac}, \mathsf{MacVf})$ consists of two PPT algorithms. The $\mathsf{Mac}(\mathsf{k}, m)$ algorithm takes a key from a key space $\mathsf{SetKeysMAC}$ and a message $m \in \{0,1\}^*$ to generate a tag tag. The algorithm $\mathsf{MacVf}(\mathsf{k}, \mathsf{tag}, m)$ takes as input a key, a tag, and a message, returning a boolean value. For all messages m and keys k, we require that $\mathsf{MacVf}(\mathsf{k}, \mathsf{Mac}(\mathsf{k}, m), m)$ always evaluates to true.

The strong unforgeability security of a MAC scheme against an adversary $\mathcal{A}$ is evaluated by the probability that $\mathcal{A}$ wins the game $\mathsf{sUnforg}$, defined in Figure 11.

$\mathsf{sUnforg}(\mathcal{A})$	$\mathcal{O}_{\mathsf{Mac}}(\mathsf{m})$
$Q \leftarrow \emptyset$	$\mathsf{tag} \leftarrow \mathsf{MAC}(\mathsf{k}, \mathsf{m})$
$\mathsf{k} \leftarrow\!\!\$\ \mathsf{Keys}$	$Q \leftarrow Q \cup \{(\mathsf{m}, \mathsf{tag})\}$
$(\mathsf{m}', \mathsf{tag}') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Mac}}}()$	**return** tag
if $(\mathsf{m}', \mathsf{tag}') \notin Q$ **then**	
return $\mathsf{MacVf}(\mathsf{k}, \mathsf{tag}', \mathsf{m}')$	
return 0	

Fig. 11: Strong forgeability security game of a MAC scheme.

We also consider another security notion, *message-key bound* security—inspired by the signature security notions of [15]—that can be ensured when $\mathsf{MAC}(\mathsf{k}, \mathsf{m})$ is given by $\mathsf{H}((\mathsf{k}, \mathsf{m}))$ for any collision-resistant hash function H.

The advantage of an adversary $\mathcal{A}$ against this game is defined as:

$$\Pr\left[\begin{array}{l}(\mathsf{k}_1, \mathsf{m}_1) \neq (\mathsf{k}_2, \mathsf{m}_2) \\ \wedge \mathsf{MacVf}\,(\mathsf{k}_1, \mathsf{tag}, \mathsf{m}_1) = 1 \\ \wedge \mathsf{MacVf}\,(\mathsf{k}_2, \mathsf{tag}, \mathsf{m}_2) = 1\end{array} : (\mathsf{m}_1, \mathsf{m}_2, \mathsf{k}_1, \mathsf{k}_2, \mathsf{tag}) \leftarrow \mathcal{A}()\right].$$

Note that, using an argument similar to the one given in [47] for KDFs, a MAC cannot claim $\mathsf{MacColl}$ security against all efficient adversaries, since the mere existence of a collision (even if the colliding values are not yet known) implies the existence of an efficient adversary.

To capture a stronger notion secure against all efficient adversaries, one may employ a prefixed variant of this property. This game relies on an external source $\mathsf{AuxInput}$ and the advantage of an adversary $\mathcal{A}$ against it is given by:

$$\Pr\left[\begin{array}{l}(\mathsf{k}_1, \mathsf{m}_1) \neq (\mathsf{k}_2, \mathsf{m}_2) \\ \wedge \mathsf{MacVf}\,(\mathsf{k}_1, \mathsf{tag}, (\mathsf{in}, \mathsf{m}_1)) = 1 \\ \wedge \mathsf{MacVf}\,(\mathsf{k}_2, \mathsf{tag}, (\mathsf{in}, \mathsf{m}_2)) = 1\end{array} : \begin{array}{l}\mathsf{in} \leftarrow \mathsf{AuxInput} \\ (\mathsf{m}_1, \mathsf{m}_2, \mathsf{k}_1, \mathsf{k}_2, \mathsf{tag}) \leftarrow \mathcal{A}(\mathsf{in})\end{array}\right].$$

To apply the prefixed variant to DAKE and its variants, the external source can be taken as the public parameters of the AKE. In this case, the public parameters must be prefixed to the message used in the computation of $\mathsf{m}^{\mathsf{R}}_{\mathsf{tag}}$.

Pseudo-Random Generator. A pseudorandom generator (PRG) is a function $\mathsf{PRG} : \mathcal{K}_{\mathsf{PRG}} \times \mathsf{In}_{\mathsf{PRG}} \rightarrow \mathsf{Out}_{\mathsf{PRG}}$, where the security against an adversary $\mathcal{A}$ is quantified by the advantage:

$$\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}(\mathcal{A}) = \Big|\Pr\left[b = b' \mid b' \leftarrow \mathcal{A}(\mathsf{out}_b); \mathsf{out}_1 = \mathsf{PRG}(\mathsf{k}); \right. \\ \left. (\mathsf{k}, \mathsf{out}_0) \leftarrow\!\!\$\ \mathcal{K}_{\mathsf{PRG}} \times \mathsf{Out}_{\mathsf{PRG}};\ b \leftarrow\!\!\$\ \{0,1\}\right] - 1/2\Big|.$$

Key Derivation Function. A key derivation function (KDF) is a function $\mathsf{KDF} : \mathcal{K}_{\mathsf{KDF}} \times \{0,1\}^* \to \mathsf{Out}_{\mathsf{KDF}}$. Its security is defined through the following game, where the adversary's advantage is given by $|\Pr[b = b'] - 1/2|$.

IND-KDF
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1()$
$(\mathsf{k}, \mathsf{out}_0) \leftarrow\!\!\$\ \mathcal{K}_{\mathsf{KDF}} \times \mathsf{Out}_{\mathsf{KDF}}$
$\mathsf{out}_1 = \mathsf{KDF}(\mathsf{k}, m)$
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow \mathcal{A}_2(\mathsf{state}, \mathsf{out}_b)$
return $b = b'$

We also consider the collision-resistance security of a KDF. The advantage of an adversary $\mathcal{A}$ against the collision-resistance experiment of KDF is defined as

$$\Pr[(\mathsf{k},\mathsf{m}) \neq (\mathsf{k}',\mathsf{m}') \wedge \mathsf{KDF}(\mathsf{k},\mathsf{m}) = \mathsf{KDF}(\mathsf{k}',\mathsf{m}') : (\mathsf{k},\mathsf{k}',\mathsf{m},\mathsf{m}') \leftarrow \mathcal{A}()] \,.$$

Note that, as with a MAC and for the same reason, a KDF cannot be secure against all efficient adversaries, since the mere existence of a collision already implies the existence of such an adversary.

Similarly, to capture a stronger notion that remains secure against all efficient adversaries, one may employ a prefixed variant of this property. This game relies on an external source $\mathsf{AuxInput}$ and its advantage against an adversary $\mathcal{A}$ is given by

$$\Pr\left[(\mathsf{k},\mathsf{m}) \neq (\mathsf{k}',\mathsf{m}') \wedge \mathsf{KDF}(\mathsf{k},(\mathsf{in},\mathsf{m})) = \mathsf{KDF}(\mathsf{k}',(\mathsf{in},\mathsf{m}')) : \begin{array}{l}\mathsf{in} \leftarrow \mathsf{AuxInput}(),\\ (\mathsf{k},\mathsf{k}',\mathsf{m},\mathsf{m}') \leftarrow \mathcal{A}()\end{array}\right].$$

When applying the prefixed variant to DAKE and its variants, the external source may be taken as the public parameters of the AKE. In this case, ensuring inverse-match soundness requires prefixing the public parameters to the message transcript being added.

Hash Function. A hash function is a function $\mathsf{H} : \{0,1\}^* \to \mathsf{SetHashs}$. The advantage of an adversary $\mathcal{A}$ against the collision-resistance experiment $\mathsf{HashColl}$ of H is defined as

$$\Pr[\mathsf{m} \neq \mathsf{m}' \wedge \mathsf{H}(\mathsf{m}) = \mathsf{H}(\mathsf{m}') : (\mathsf{m},\mathsf{m}') \leftarrow \mathcal{A}()].$$

An analogous observation to the one above on collisions in KDF functions applies to collisions in hash functions, and a prefixed variant can likewise be defined.

B Proof of the CK-FO Transform

We first recall and introduce some notations that will be commonly used in the proofs for the $\mathsf{CK\text{-}FO}$ (Section 4.2) in both the ROM and QROM.

The PKE schemes PKE_L and PKE_R are subject to the conditions described in Section 4.1. In particular, they share the same setup algorithm and satisfy the following two properties:

- The randomness spaces are structured as $\mathsf{Rand}_\mathsf{C} \times \mathsf{Rand}_\mathsf{L}$ for Encr_L and $\mathsf{Rand}_\mathsf{C} \times \mathsf{Rand}_\mathsf{R}$ for Encr_R, where Rand_C denotes the common part. We define $\mathsf{Rand}_{\mathsf{CLR}} = \mathsf{Rand}_\mathsf{C} \times \mathsf{Rand}_\mathsf{L} \times \mathsf{Rand}_\mathsf{R}$.
- There exists a deterministic algorithm E_{base} that, given $(\mathsf{pp}, r_\mathsf{C})$ with $r_\mathsf{C} \in \mathsf{Rand}_\mathsf{C}$, outputs c_C such that both Encr_L and Encr_R produce ciphertexts whose first component equals $\mathsf{c}_\mathsf{C} = E_{\mathsf{base}}(\mathsf{pp}, r_\mathsf{C})$. We denote by $\mathsf{Ciphers}_\mathsf{C}$ the set of all such values c_C. Moreover, we denote by $\mathsf{Ciphers}_\mathsf{L}$ and $\mathsf{Ciphers}_\mathsf{R}$ the sets such that the ciphertext spaces of Encr_L and Encr_R are respectively structured as $\mathsf{Ciphers}_\mathsf{C} \times \mathsf{Ciphers}_\mathsf{L}$ and $\mathsf{Ciphers}_\mathsf{C} \times \mathsf{Ciphers}_\mathsf{R}$.

We denote $\mathsf{Enc}_{\mathsf{OTP}}$ and $\mathsf{Dec}_{\mathsf{OTP}}$ as the following algorithms:

$\mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})$	$\mathsf{Dec}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{sk}_\mathsf{L}, \mathsf{sk}_\mathsf{R}, (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}}))$
$(r_\mathsf{L}, r_\mathsf{R}, r_\mathsf{C}) \leftarrow \mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})$	$m_\mathsf{L} \leftarrow \mathsf{Decr}_\mathsf{L}(\mathsf{pp}, \mathsf{sk}_\mathsf{L}, (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}))$
$(\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}) \leftarrow \mathsf{Encr}_\mathsf{L}(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, m_\mathsf{L}; r_\mathsf{C}, r_\mathsf{L})$	**if** $m_\mathsf{L} = \perp$ **then return** $(\perp, \perp)$
$(\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{R}) \leftarrow \mathsf{Encr}_\mathsf{R}(\mathsf{pp}, \mathsf{pk}_\mathsf{R}, m_\mathsf{R}; r_\mathsf{C}, r_\mathsf{R})$	$\mathsf{c}_\mathsf{R} = c_\mathsf{R}^{\mathsf{OTP}} \oplus \mathsf{H}_{\mathsf{OTP}}(m_\mathsf{L})$
$c_\mathsf{R}^{\mathsf{OTP}} = \mathsf{c}_\mathsf{R} \oplus \mathsf{H}_{\mathsf{OTP}}(m_\mathsf{L})$	$m_\mathsf{R} \leftarrow \mathsf{Decr}_\mathsf{R}(\mathsf{pp}, \mathsf{sk}_\mathsf{R}, (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{R}))$
return $(\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$	**if** $m_\mathsf{R} = \perp$ **then return** $(\perp, \perp)$
	return $(m_\mathsf{L}, m_\mathsf{R})$

Moreover, if we explicitly indicate some elements $(r_{\mathsf{C}}, r_{\mathsf{L}}, r_{\mathsf{R}})$ as randomness, such as in

$$\mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}}; r_{\mathsf{C}}, r_{\mathsf{L}}, r_{\mathsf{R}}),$$

the elements $(r_{\mathsf{C}}, r_{\mathsf{L}}, r_{\mathsf{R}})$ will be used instead of $\mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}})$.

Lastly, we will also suppose that $\mathsf{Ciphers}_{\mathsf{R}}$ is included in a binary set $\{0,1\}^{\tau}$ and choose $\mathsf{H}_{\mathsf{OTP}}$ to have range $\{0,1\}^{\tau}$.

B.1 Proof in the Random Oracle Model

[IND-CK-CCA, -] and [IND-CCA, -]

$\mathsf{pp} \leftarrow \mathsf{Setup}()$
$(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{L},\star} = (\mathsf{sk}_{\mathsf{L},\star}, \mathsf{s}_{\mathsf{L},\star})) \leftarrow \mathsf{dKeygen}^{\mathsf{FO}}_{\mathsf{L}}(\mathsf{pp})$
$\mathsf{LeakedR} = \emptyset$
$\mathsf{Oracles} = \{\mathcal{O}_{\mathsf{CCA}_{\mathsf{L}}}, \mathcal{O}_{\mathsf{leak}_{\mathsf{R}}}, \mathsf{H}_{\mathsf{rej}}, \mathsf{H}_{\mathsf{rand}}, \mathsf{H}_{\mathsf{key}}, \mathsf{H}_{\mathsf{OTP}}\}$
$(\mathsf{state}, \mathsf{pk}_{\mathsf{R},\star}) \leftarrow \mathcal{A}_1^{\mathsf{Oracles}}(\mathsf{pk}_{\mathsf{L},\star})$
$b \leftarrow\!\$\ \{0,1\}$
$(\mathsf{ct}_\star, \mathsf{k}_0) \leftarrow \mathsf{dEncaps}^{\mathsf{FO}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star})$
We parse $\mathsf{ct}_\star$ as $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star})$
We denote by $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$ and by
$(r_{\mathsf{C},\star}, r_{\mathsf{L},\star}, r_{\mathsf{R},\star})$ the messages and
randomnesses used to produce
this encapsulation.
$\mathsf{k}_1 \leftarrow\!\$\ \mathsf{Keys}$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Oracles}}(\mathsf{state}, \mathsf{ct}_\star, \mathsf{k}_b)$
return $b = b'$

$\mathcal{O}_{\mathsf{leak}_{\mathsf{R}}}()$

// with LeakedR taken from the security game
$r \leftarrow\!\$\ \mathsf{RandKeygenR}$
$(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{R}} = (\mathsf{sk}_{\mathsf{R}}, \mathsf{s}_{\mathsf{R}})) \leftarrow \mathsf{dKeygen}^{\mathsf{FO}}_{\mathsf{R}}(r)$
$\mathsf{LeakedR} \leftarrow \mathsf{LeakedR} \cup \{(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{R}}, r)\}$
$\mathsf{LeakedpkR} \leftarrow \mathsf{LeakedpkR} \cup \{\mathsf{pk}_{\mathsf{R}}\}$
return $(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{R}}, r)$

$\mathsf{H}_{\mathsf{OTP}}, \mathsf{H}_{\mathsf{rej}}, \mathsf{H}_{\mathsf{rand}}, \underline{\mathsf{H}}_{\mathsf{Rej}}$

Random oracles

$\mathcal{O}_{\mathsf{CCA}_{\mathsf{L}}}(\mathsf{pk}_{\mathsf{R}}, \mathsf{ct} = (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}}))$

// with $\mathsf{ct}_\star$, LeakedR and $\mathsf{sk}_{\mathsf{L},\star}$
// taken from the security game
if $(\exists \mathsf{sk}^{\mathsf{FO}}_{\mathsf{R}} = (\mathsf{sk}_{\mathsf{R}}, \mathsf{s}_{\mathsf{R}}), r : (\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{R}}, r) \in \mathsf{LeakedR})$
  $\wedge\ (\mathsf{ct} \neq \mathsf{ct}_\star \vee \mathsf{pk}_{\mathsf{R}} \neq \mathsf{pk}_{\mathsf{R},\star})$ **then**
  $\widetilde{\mathsf{k}} \leftarrow \mathsf{H}_{\mathsf{rej}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, \mathsf{s}_{\mathsf{L}}, \mathsf{s}_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ / G_0
  $\widetilde{\mathsf{k}} \leftarrow \underline{\mathsf{H}}_{\mathsf{Rej}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, \mathsf{s}_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ / G_1 – G_3
  return $\widetilde{\mathsf{k}}$ / G_2 – G_3
$(m_{\mathsf{L}}, m_{\mathsf{R}}) \leftarrow \mathsf{Dec}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{sk}_{\mathsf{L}}, \mathsf{sk}_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ / G_0 – G_1
if $(m_{\mathsf{L}}, m_{\mathsf{R}}) = (\perp, \perp)$ **then** / G_0 – G_1
  return $\widetilde{\mathsf{k}}$ / G_0 – G_1
if $\mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}})$
  $\neq (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ **then** / G_0 – G_1
  return $\widetilde{\mathsf{k}}$ / G_0 – G_1
return $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ / G_0 – G_1

$\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$

if $\exists \mathsf{k} : (\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}}, \mathsf{k}) \in \mathcal{L}_{\mathsf{H}_{\mathsf{key}}}$ **then**
  return k
$\mathsf{k} \leftarrow\!\$\ \mathsf{Keys}$
if $(\exists \mathsf{sk}^{\mathsf{FO}}_{\mathsf{R}} = (\mathsf{sk}_{\mathsf{R}}, \mathsf{s}_{\mathsf{R}}), r : (\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{R}}, r) \in \mathsf{LeakedR})$
  $\wedge\ \mathsf{pk}_{\mathsf{L}} = \mathsf{pk}_{\mathsf{L},\star}$ **then** / G_2 – G_3
$(c'_{\mathsf{C}}, c'_{\mathsf{L}}, c'_{\mathsf{R}}) = \mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}})$ / G_2 – G_3
$(m'_{\mathsf{L}}, m'_{\mathsf{R}}) = \mathsf{Dec}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{sk}_{\mathsf{L}}, \mathsf{sk}_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ / G_2
if $(c'_{\mathsf{C}}, c'_{\mathsf{L}}, c'_{\mathsf{R}}, m'_{\mathsf{L}}, m'_{\mathsf{R}}) = (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}})$ **then** / G_2
  $\mathsf{k} \leftarrow \underline{\mathsf{H}}_{\mathsf{Rej}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ / G_2
if $(c'_{\mathsf{C}}, c'_{\mathsf{L}}, c'_{\mathsf{R}}) = (\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ **then** / G_3
  $\mathsf{k} \leftarrow \underline{\mathsf{H}}_{\mathsf{Rej}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ / G_3
$\mathcal{L}_{\mathsf{H}_{\mathsf{key}}} = \{(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}}, \mathsf{k})\} \cup \mathcal{L}_{\mathsf{H}_{\mathsf{key}}}$
return k

Fig. 12: Games G_0 to G_3 for the proof of [IND-CK-CCA, -] security in the ROM (Theorem 3).

Proof. Security games for [IND-CK-CCA, -] security.
They are detailed below, the games G_0 to G_3 being in addition illustrated in Figure 12 (page 29).

Game G_0. The initial [IND-CK-CCA, -] game. We clarify some of the notations used in this game. $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{L},\star} = (\mathsf{sk}_{\mathsf{L},\star}, \mathsf{s}_{\mathsf{L},\star}))$ are the left challenge public and secret keys. $\mathsf{pk}_{\mathsf{R},\star}$ is the right challenge public key, and, only in the [IND-CK-CCA, -] game, $\mathsf{sk}^{\mathsf{FO}}_{\mathsf{R},\star} = (\mathsf{sk}_{\mathsf{R},\star}, \mathsf{s}_{\mathsf{R},\star})$ is the right challenge secret key. The ciphertext and key computed with $\mathsf{dEncaps}^{\mathsf{FO}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star})$ by the challenger form the pair $(\mathsf{ct}_\star = (c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star}), \mathsf{k}_0)$, whereas k_1 is the key uniformly sampled from Keys by the challenger. We note $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$ the messages sampled to compute the challenge ciphertexts and $(r_{\mathsf{C},\star}, r_{\mathsf{L},\star}, r_{\mathsf{R},\star}) = \mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$. b is the challenge bit, and b' is the response provided by the adversary. We also recall that $\mathsf{LeakedR}$ is the set of triplets $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}^{\mathsf{FO}}_\mathsf{R} = (\mathsf{sk}_\mathsf{R}, \mathsf{s}_\mathsf{R}), r)$ outputted by the oracle $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$.

Game G_1. The oracle $\mathsf{H}_{\mathsf{rej}}$ used by $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ is replaced by a new internal oracle $\underline{\mathsf{H}}_{\mathsf{Rej}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, \mathsf{s}_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ that no longer takes the secret s_L as input. Moreover, $\underline{\mathsf{H}}_{\mathsf{Rej}}$ is not accessible by the adversary.[25]

As s_L is sampled uniformly from $\{0,1\}^\nu$ and is not given to the adversary, we have

$$\left|\mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_0}(\mathcal{A})\right| \leq \frac{q_{\mathsf{rej}}}{2^\nu}.$$

Game G_2. In G_2, we modify the construction of $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$. When $\mathsf{pk}_\mathsf{L} = \mathsf{pk}_{\mathsf{L},\star}$, pk_R belongs to a triplet $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}^{\mathsf{FO}}_\mathsf{R} = (\mathsf{sk}_\mathsf{R}, \mathsf{s}_\mathsf{R}), r) \in \mathsf{LeakedR}$, and $(\mathsf{pk}_\mathsf{L}, \mathsf{sk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ verifies the predicate P defined by

$$\begin{aligned}&\left((\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R}) = \mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})\right)\\ \wedge &\left((m_\mathsf{L}, m_\mathsf{R}) = \mathsf{Dec}_{\mathsf{OTP}}\left(\mathsf{pp}, \mathsf{sk}_{\mathsf{L},\star}, \mathsf{sk}_\mathsf{R}, (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})\right)\right),\end{aligned}$$

we define $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ as $\underline{\mathsf{H}}_{\mathsf{Rej}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, \mathsf{s}_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$. Otherwise, we define $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ as equal to $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ as in G_1. Thanks to this new definition of $\mathsf{H}_{\mathsf{key}}$, we can rewrite the oracle $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ using only $\mathsf{H}_{\mathsf{rej}}$.

Note that for such tuples, the function that adds s_R while removing $(m_\mathsf{L}, m_\mathsf{R})$:

$$(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R}) \mapsto (\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, \mathsf{s}_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$$

is injective. Moreover, the value of $\underline{\mathsf{H}}_{\mathsf{Rej}}$ on such tuples $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ was never used in the previous game. As a result, the advantage does not change between G_1 and G_2.

Game G_3. In G_3, the predicate $\mathsf{P}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ described in G_2 is simplified to only be:

$$(\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R}) = \mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}).$$

The only situation where an adversary can distinguish G_2 from G_3 is if $\mathcal{A}$ queries $\mathsf{H}_{\mathsf{key}}$ on an input $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ for a pk_R that corresponds to a triplet $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}^{\mathsf{FO}}_\mathsf{R}, r) \in \mathsf{LeakedR}$, and where:

$$\begin{aligned}&\left((\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R}) = \mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})\right)\\ \wedge &\left((m_\mathsf{L}, m_\mathsf{R}) \neq \mathsf{Dec}_{\mathsf{OTP}}\left(\mathsf{pp}, \mathsf{sk}_{\mathsf{L},\star}, \mathsf{sk}_\mathsf{R}, (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})\right)\right).\end{aligned}$$

Because DPKE is δ-correct, we see that the probability that such a query is made is less than $q_{\mathsf{rand}}\delta$. We thus have $\left|\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})\right| \leq q_{\mathsf{rand}}\delta$.

Moreover, we can see that, by definition, G_3 no longer uses the secret key $\mathsf{sk}^{\mathsf{FO}}_{\mathsf{L},\star} = (\mathsf{sk}_{\mathsf{L},\star}, \mathsf{s}_{\mathsf{L},\star})$ and can thus be completely simulated without its knowledge.

Game G_4. We denote by $\mathsf{EventAskedLeftMess}$ the event that, at any moment in G_3, the adversary asks $m_{\mathsf{L},\star}$ to $\mathsf{H}_{\mathsf{OTP}}$ or asks an input containing left message $m_{\mathsf{L},\star}$ to $\mathsf{H}_{\mathsf{rand}}$ or $\mathsf{H}_{\mathsf{key}}$. We define G_4 as G_3 except that:

- $c^{\mathsf{OTP}}_{\mathsf{R},\star}$ is uniformly sampled in $\mathsf{Ciphers}_\mathsf{R}$ instead of being computed as $c_{\mathsf{R},\star} \oplus \mathsf{H}_{\mathsf{OTP}}(m_{\mathsf{L},\star})$.
- The randomness $(r_{\mathsf{L},\star}, r_{\mathsf{R},\star}, r_{\mathsf{C},\star})$, previously computed as $\mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$, is now independently and uniformly sampled.
- The challenge key, previously computed as $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_{\mathsf{R},\star}, c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c^{\mathsf{OTP}}_{\mathsf{R},\star})$, is now independently and uniformly sampled.
- The game returns a random bit if $\mathsf{EventAskedLeftMess}$ happens.

We now study the difference of advantage between G_3 and G_4. Note that while $\mathsf{EventAskedLeftMess}$ does not happen, the distribution of $c^{\mathsf{OTP}}_\mathsf{R}$, $(r_\mathsf{L}, r_\mathsf{R}, r_\mathsf{C})$ and k_1 is indistinguishable from independent uniform distributions by $\mathcal{A}$. Thus, we have

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A})\right| \leq \Pr[\mathsf{EventAskedLeftMess}]. \tag{1}$$

[25] We underline all oracles that are not (directly) accessible to the adversary.

For the same reason, the event $\mathsf{EventAskedLeftMess}$ has the same probability to appear in G_3 than in G_4. Thus, we study the probability of $\mathsf{EventAskedLeftMess}$ happening in G_4.

To this end, we build an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the $\mathsf{IND\text{-}CPA}$ security of $\mathsf{PKE}_\mathsf{L} = (\mathsf{Keygen}_\mathsf{L}, \mathsf{Encr}_\mathsf{L}, \mathsf{Decr}_\mathsf{L})$, using in black-box the adversary $\mathcal{A}$ of G_4.

$\mathcal{B}_1$ is given $(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star})$ and returns two messages $(m_{\mathsf{L},\mathsf{Fixed}}, m_{\mathsf{L},\star})$ to the challenger where $m_{\mathsf{L},\star}$ is uniformly sampled and $m_{\mathsf{L},\mathsf{Fixed}}$ is any fixed left message. It then receives a challenge ciphertext $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star})$ that is either an encryption of $m_{\mathsf{L},\mathsf{Fixed}}$ (if some bit u sampled by the $\mathsf{IND\text{-}CPA}$ challenger is 0) or $m_{\mathsf{L},\star}$ (if u is 1).

$\mathcal{B}_2$ uses $(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, c_{\mathsf{C},\star}, c_{\mathsf{L},\star})$ to make a simulation of G_4 to the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with left challenge public key $\mathsf{pk}_{\mathsf{L},\star}$. First, $\mathcal{B}$ gives $(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star})$ to $\mathcal{A}_1$ that returns the right challenge public key $\mathsf{pk}_{\mathsf{R},\star}$. Then, $\mathcal{B}$ uniformly samples $c_\mathsf{R}^{\mathsf{OTP}} \leftarrow\!\!\$\ \mathsf{Ciphers}_\mathsf{R}$ and simulates the rest of G_4 with challenge ciphertext $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_\mathsf{R}^{\mathsf{OTP}})$.

Moreover, each time an input is requested to $\mathsf{H}_{\mathsf{OTP}}$, $\mathsf{H}_{\mathsf{rand}}$ or $\mathsf{H}_{\mathsf{key}}$, $\mathcal{B}_2$ verifies if $m_{\mathsf{L},\star}$ is part of the input. If it is true, it returns 1 (*i.e.* $\mathcal{B}_2$ thinks that $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star})$ is an encryption of $m_{\mathsf{L},\star}$). If this never happens, at the end of the simulation, it returns 0 (*i.e.*, $\mathcal{B}_2$ thinks that $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star})$ is an encryption of $m_{\mathsf{L},\mathsf{Fixed}}$).

We will show the following inequality:

$$\Pr[\mathsf{EventAskedLeftMess}] \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}_\mathsf{L}}(\mathcal{B}) + \frac{q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}}}{|\mathcal{M}_\mathsf{L}|}. \tag{2}$$

It is trivially verified if

$$\Pr[\mathsf{EventAskedLeftMess}] \leq \frac{(q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}})}{|\mathcal{M}_\mathsf{L}|}.$$

We now suppose that

$$\Pr[\mathsf{EventAskedLeftMess}] > \frac{(q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}})}{|\mathcal{M}_\mathsf{L}|}.$$

A direct analysis of $\mathcal{B}$ shows that:

- If the $\mathsf{IND\text{-}CPA}$ challenger sampled $u = 0$, any value seen by $\mathcal{A}$ is independent of $m_{\mathsf{L},\star}$, and thus, $\Pr[\mathcal{B} \text{ returns } 1 \mid u = 0] \leq \frac{(q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}})}{|\mathcal{M}_\mathsf{L}|}$.
- If the $\mathsf{IND\text{-}CPA}$ challenger sampled $u = 1$, $\mathcal{B}$ perfectly simulates G_4 to $\mathcal{A}$ and $\mathcal{B}_2$ returns 1 exactly when, in this simulation of G_4, $\mathsf{EventAskedLeftMess}$ happens. Thus $\Pr[\mathcal{B} \text{ returns } 1 \mid u = 1] = \Pr[\mathsf{EventAskedLeftMess}]$.

It follows that:

$$\begin{aligned} 2\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}_\mathsf{L}}(\mathcal{B}) &= |\Pr[\mathcal{B} \text{ returns } 1 \mid u = 1] - \Pr[\mathcal{B} \text{ returns } 1 \mid u = 0]| \\ &> \Pr[\mathsf{EventAskedLeftMess}] - \frac{(q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}})}{|\mathcal{M}_\mathsf{L}|}, \end{aligned}$$

which indeed implies Equation (2). Using Equations (1) and (2), we find:

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A})\right| \leq 2\,\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}_\mathsf{L}}(\mathcal{B}) + \frac{(q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}})}{|\mathcal{M}_\mathsf{L}|}.$$

Finally, we see that in G_4, k_0 and k_1 are both uniformly sampled and independent of any other elements given to $\mathcal{A}$. Thus, $\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) = 0$. This completes the proof.

Security games for [- , IND-CCA] security.

Game G_0. The initial game [- , IND-CCA]. We use similar notations as for the proof of the [IND-CK-CCA, -] security. $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star})$ are the left challenge public and secret keys, computed with randomness $\mathsf{r}_{\mathsf{L},\star}$, while $(\mathsf{pk}_{\mathsf{R},\star}, \mathsf{sk}_{\mathsf{R},\star})$ are the right challenge public and secret keys. The ciphertext and key computed with $\mathsf{dEncaps}^{\mathsf{FO}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star})$ by the challenger form the pair $(\mathsf{ct}_\star = (c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star}), \mathsf{k}_0)$, whereas k_1 is the key uniformly sampled from Keys by the challenger. We note $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$ the messages sampled to compute the challenge ciphertexts and $(r_{\mathsf{C},\star}, r_{\mathsf{L},\star}, r_{\mathsf{R},\star}) = \mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$. b is the challenge bit, and b' is the response provided by the adversary. We also recall that $\mathsf{LeakedR}$ is the set of triplets $(\mathsf{pk}_\mathsf{L}, \mathsf{sk}_\mathsf{L}^{\mathsf{FO}} = (\mathsf{sk}_\mathsf{L}, \mathsf{s}_\mathsf{L}), r)$ outputted by the oracle $\mathcal{O}_{\mathsf{leak}_\mathsf{L}}$.

We can construct $\mathsf{G}_1, \ldots, \mathsf{G}_3$ by applying the same modifications as those used in [IND-CK-CCA, -] security, with only two differences:

– We invert the left and right elements,
– In $\mathsf{G}_3, \mathsf{G}_4$ and G_5, we do not only apply the changes when pk_L belongs to a triplet $(\mathsf{pk}_\mathsf{L}, \mathsf{sk}_\mathsf{L}^{\mathsf{FO}} = (\mathsf{sk}_\mathsf{L}, \mathsf{s}_\mathsf{L}), r) \in \mathsf{LeakedL}$, but also include $\mathsf{pk}_\mathsf{L} = \mathsf{pk}_{\mathsf{L},\star}$, even if $\mathsf{pk}_{\mathsf{L},\star}$ does not belong to such a triplet.

Using the same reasoning as with the [IND-CK-CCA, -] security, we obtain:

$$\left|\mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})\right| \leq \frac{q_{\mathsf{rej}}}{2^\nu} + q_{\mathsf{rand}}\delta.$$

Game G_4. We denote by EventAskHashKey the event that, at any moment in G_3, the adversary asks $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_{\mathsf{R},\star}, c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star})$ to $\mathsf{H}_{\mathsf{key}}$.

We define G_4 as G_3 except that the game returns a random bit if the event EventAskHashKey happens. By definition, we have:

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A})\right| \leq \Pr[\mathsf{EventAskHashKey}]. \tag{3}$$

In order to bound $\Pr[\mathsf{EventAskHashKey}]$, we build an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the [- , IND-CPA] security of the 2K2M-PKE DPKE, using in black-box the adversary $\mathcal{A}$ of G_3.

First, $\mathcal{B}_1$ obtains public parameters, randomness and keys $(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star}, \mathsf{r}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star})$, and then gives $(m_{\mathsf{R},\mathsf{Fixed}}, m_\mathsf{R}, m_\mathsf{L})$ to the challenger, m_R and m_L being uniformly sampled and $m_{\mathsf{R},\mathsf{Fixed}}$ being any fixed right message. It then receives from the challenger a challenge ciphertext $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star})$ which is either an encryption of the two messages $(m_{\mathsf{L},\star}, m_{\mathsf{R},\mathsf{Fixed}})$ (if some bit u sampled by the [- , IND-CPA] challenger is 0) or an encryption of the two messages $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$ (if u is 1).

$\mathcal{B}_2$ uses these elements to make a perfect simulation of G_3 to $\mathcal{A}$ with challenge ciphertext $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c^{\mathsf{OTP}}_{\mathsf{R},\star} = c_{\mathsf{R},\star} \oplus \mathsf{H}_{\mathsf{OTP}}(m_{\mathsf{L},\star}))$. This is possible because knowledge of $\mathsf{sk}_{\mathsf{R},\star}$ is not required to simulate G_3.

Moreover, each time $\mathsf{H}_{\mathsf{key}}$ is queried by $\mathcal{A}$ on an input of the form $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_\mathsf{R}, c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star})$, $\mathcal{B}_2$ checks if $m_\mathsf{R} = m_{\mathsf{R},\star}$. If it is the case, it stops and returns 1 (*i.e.* $\mathcal{B}_2$ thinks that $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star})$ is an encryption of $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$). If this never happens during the simulation, $\mathcal{B}_2$ returns 0 (*i.e.* $\mathcal{B}_2$ thinks that $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star})$ is an encryption of $(m_{\mathsf{L},\star}, m_{\mathsf{R},\mathsf{Fixed}})$).

We will show the following inequality:

$$\Pr[\mathsf{EventAskHashKey}] \leq 2\,\mathsf{Adv}^{[\,\text{-}\,,\mathsf{IND\text{-}CPA}]}_{\mathsf{DPKE}}(\mathcal{B}) + \frac{q_{\mathsf{key}}}{|\mathcal{M}_\mathsf{R}|}. \tag{4}$$

It is trivially verified if

$$\Pr[\mathsf{EventAskHashKey}] \leq \frac{q_{\mathsf{key}}}{|\mathcal{M}_\mathsf{R}|}.$$

We now suppose that

$$\Pr[\mathsf{EventAskHashKey}] > \frac{q_{\mathsf{key}}}{|\mathcal{M}_\mathsf{R}|}.$$

A direct analysis of $\mathcal{B}$ shows that:
– If the [- , IND-CPA] challenger sampled $u = 0$, any value seen by $\mathcal{A}$ is independent of $m_{\mathsf{R},\star}$, and thus, $\Pr[\mathcal{B} \text{ returns } 1 \mid u = 0] \leq \frac{q_{\mathsf{key}}}{|\mathcal{M}_\mathsf{R}|}$.
– If the [- , IND-CPA] challenger sampled $u = 1$, $\mathcal{B}$ makes a perfect simulation of G_3 to $\mathcal{A}$ and $\mathcal{B}_2$ returns 1 exactly when EventAskHashKey happens in this simulation of G_3. Thus $\Pr[\mathcal{B} \text{ returns } 1 \mid u = 1] = \Pr[\mathsf{EventAskHashKey}]$.

Thus, we have:

$$\begin{aligned} 2\,\mathsf{Adv}^{[\,\text{-}\,,\mathsf{IND\text{-}CPA}]}_{\mathsf{DPKE}}(\mathcal{B}) &= \left|\Pr[\mathcal{B} \text{ returns } 1 \mid u = 1] - \Pr[\mathcal{B} \text{ returns } 1 \mid u = 0]\right| \\ &> \Pr[\mathsf{EventAskHashKey}] - \frac{q_{\mathsf{key}}}{|\mathcal{M}_\mathsf{R}|}, \end{aligned}$$

which indeed implies Equation (4).

Using Equations (3) and (4), we obtain

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A})\right| \leq 2\,\mathsf{Adv}^{[\,\text{-}\,,\mathsf{IND\text{-}CPA}]}_{\mathsf{DPKE}}(\mathcal{B}) + \frac{q_{\mathsf{key}}}{|\mathcal{M}_\mathsf{R}|}. \tag{5}$$

Finally, we see that in G_4, whenever EventAskHashKey does not happen, k_0 and k_1 are both uniformly sampled and independent of any other element given to $\mathcal{A}$. Thus, $\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) = 0$ and we can conclude.

B.2 Proof in the Quantum-Accessible Random Oracle Model

The proof of this theorem uses the One-Way to Hide Lemma with Semi-Classical Oracles introduced in [3] and follows the same ideas than the proof of [23, Theorem 3.2].

We begin by introducing some necessary results in two subsections before doing the proof, using the quantum notations defined in [23, Section 6].

Generic Distinguishing Problem. For $\lambda \in [0,1]$, we denote by B_λ the Bernoulli distribution with parameter λ.

Lemma 2 (Generic Distinguishing Problem with Bounded Probabilities [32]). *Let X be a finite set, and let $\lambda \in [0,1]$. For any (unbounded, quantum) algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ issuing at most $\mathsf{q_O}$ quantum queries to the oracle $\mathcal{O}$ given by the games $\mathsf{GDPB}_{\lambda,0}$ and $\mathsf{GDPB}_{\lambda,1}$ defined in Figure 13, we have:*

$$\left|\Pr\left[\mathsf{GDPB}^{\mathcal{A}}_{\lambda,0} \Rightarrow 1\right] - \Pr\left[\mathsf{GDPB}^{\mathcal{A}}_{\lambda,1} \Rightarrow 1\right]\right| \leq 8\,\lambda\,(\mathsf{q_O}+1)^2.$$

Game $\mathsf{GDPB}^{\mathcal{A}}_{\lambda,b}$

$(\lambda_x)_{x\in X} \leftarrow \mathcal{A}_1$
if $\exists x \in X$ such that $\lambda_x > \lambda$ **then return** 0
if $b = 1$ **then**
  for $x \in X$: $\mathcal{O}(x) \leftarrow B_{\lambda_x}$
else
  for $x \in X$: $\mathcal{O}(x) \leftarrow 0$
$b' \leftarrow \mathcal{A}_2^{|\mathcal{O}\rangle}$
return b'

Fig. 13: Generic distinguishing games $\mathsf{GDPB}_{\lambda,b}$ with bounded maximal Bernoulli parameter $\lambda \in [0,1]$.

Semi-Classical Oracles and One-Way to Hide Lemma. We follow the presentation of semi-classical oracles made in [23], using the quantum notations defined in [23, Section 6]. In particular, for a finite set X, we define $\mathcal{H}_X$ the Hilbert space with canonical orthonormal basis $\{|x\rangle, x \in X\}$. We define the X register as the space of $\sum_{x\in X} \alpha_x |x\rangle$ such that $\sum_{x\in X} |\alpha_x|^2 = 1$.

From any finite sets X, Y and subset $S \subset X$, one can define a *semi-classical* oracle $\mathcal{O}_S^{\mathsf{SC}}$ that we describe now. Intuitively, the oracle $\mathcal{O}_S^{\mathsf{SC}}$ collapses a state $\sum_{(x,y)\in X\times Y} \alpha_{x,y} |x\rangle \otimes |y\rangle \in \mathcal{H}_X \otimes \mathcal{H}_Y$ to a state whose X-register contains only elements of either S or $X \setminus S$. More precisely, $\mathcal{O}_S^{\mathsf{SC}}$ takes as input a quantum state $|\psi, 0\rangle$ such that $|\psi\rangle \in_X \otimes \mathcal{H}_Y$. $\mathcal{O}_S^{\mathsf{SC}}$ first measures the X-register with respect to the projectors $M_1 := \sum_{x\in S} |x\rangle\langle x|$ and $M_0 := \sum_{x\notin S} |x\rangle\langle x|$ and then initializes the last register to $|b\rangle$ for the measured bit b. Consequently, $|\psi, 0\rangle$ collapses to either a state $|\psi', 1\rangle$ such that the X-register of $|\psi'\rangle$ only contains elements of S, or a state $|\psi', 0\rangle$ such that the X-register of $|\psi'\rangle$ only contains elements of $X \setminus S$.

To a quantum-accessible oracle $\mathcal{O}$ and a subset $S \subset X$, one can furthermore associate oracle $\mathcal{O} \setminus S$ which first queries $\mathcal{O}_S^{\mathsf{SC}}$ and then $\mathcal{O}$. Let FIND denote the event that $\mathcal{O}_S^{\mathsf{SC}}$ ever returns a state $|\psi', 1\rangle$. Unless FIND occurs, the outcome of $\mathcal{O} \setminus S$ is independent of the values $\mathcal{O}(x)$ for all $x \in S$, which is why $\mathcal{O} \setminus S$ is also called a "punctured" oracle.

Lemma 3 (One Way To Hiding Lemma of [23, Lemmas 5.2 and 5.3]). *Let $S \subset X$ be random and $\mathcal{O}_1, \mathcal{O}_2 \in Y^X$ be random functions such that $\mathcal{O}_1(x) = \mathcal{O}_2(x)$ for all $x \in X \setminus S$. Let also z be a random bitstring, where $(\mathcal{S}, \mathcal{O}_1, \mathcal{O}_2, z)$ may have an arbitrary joint distribution. Let $q \in \mathbb{N}$ and $\mathcal{A}$ a quantum algorithm with binary output and making less than q queries to a random oracle $\mathcal{O}_1$. Moreover, for $i \in \{1,2\}$, let*

$$p_i := \Pr_{\mathcal{O}_i,z}\left[1 \leftarrow \mathcal{A}^{|\mathcal{O}_i\rangle}(z)\right],$$
$$p_{\mathsf{Find}} = \Pr_{\mathcal{O}_2,\mathcal{S},z}\left[b \leftarrow \mathcal{A}^{|\mathcal{O}_2\setminus\mathcal{S}\rangle}(z) : \mathsf{FIND}\right].^{26}$$

Then:

$$|p_1 - p_2| \leq 2\sqrt{(q+1)\, p_{\mathsf{FIND}}}.$$

Lemma 4 (Corollary 1 of [3]). *Using the notations of Lemma 3, suppose that $\mathcal{S}$ and z are independent and that $\mathcal{C}$ is a q-query algorithm. Let $p_{\mathsf{max}} := \max_{x \in X} \Pr[x \in \mathcal{S}]$. Then, we have $\Pr_{\mathcal{O}_2,\mathcal{S},z}\left[\mathsf{FIND} : \mathcal{A}^{|\mathcal{O}_2 \setminus \mathcal{S}\rangle}\right] \leq 4q \cdot p_{\mathsf{max}}$.*

Lemma 5 (Using a hash function as a PRF in QROM, Adaptation of Corollary 1 of [8]). *Let $\mathsf{H} : (K \times X) \to Y$ be a quantum-accessible random oracle. This function may be used as a quantum-accessible PRF $F_k(x) := \mathsf{H}(k, x)$ with a key $k \hookleftarrow\$\ K$. Suppose a PRF-adversary $\mathcal{A}$ makes q queries to H and any number of queries to F_k. Then $\mathsf{Adv}^{\mathsf{IND\text{-}PRF}}(\mathcal{A}) \leq 2q/\sqrt{|K|}$, where:*

$$\mathsf{Adv}^{\mathsf{IND\text{-}PRF}}(\mathcal{A}) = \Pr\left[b = b' \mid b' \leftarrow \mathcal{A}^{|\mathcal{O}_b\rangle}(); \right. \\ \left. \mathcal{O}_0 = \mathsf{H}(k, \cdot), (\mathsf{k}, \mathcal{O}_1) \hookleftarrow\$\ K \times Y^X;\ b \hookleftarrow\$\ \{0,1\}\right].$$

QROM Proof of [IND-CK-CCA, -] Security

Proof. The detailed descriptions of the games are given below, with an overview of Games G_0 to G_5 provided in Figure 14 (page 35).

Game G_0. This is the initial game [IND-CK-CCA, -].

We clarify some of the notations used in this game. $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star})$ are the left challenge public and secret keys, while $\mathsf{pk}_{\mathsf{R},\star}$ is the right challenge public key. The ciphertext and key from $\mathsf{dEncaps}^{\mathsf{FO}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star})$, computed by the challenger, form the pair $(\mathsf{ct}_\star = (c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star}), \mathsf{k}_0)$, whereas k_1 is the key uniformly sampled from Keys by the challenger. We note $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$ the messages sampled to compute the challenge ciphertexts and $(r_{\mathsf{C},\star}, r_{\mathsf{L},\star}, r_{\mathsf{R},\star}) = \mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$. b is the challenge bit and b' is the response provided by the adversary. We also recall that $\mathsf{LeakedR}$ is the set of triplets $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R}^{\mathsf{FO}} = (\mathsf{sk}_\mathsf{R}, \mathsf{s}_\mathsf{R}), r)$ outputted by the oracle $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$.

Game G_1. As we know there are at most q_{leak} requests to $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$ so we can precompute at the beginning of the game the elements $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}_\mathsf{R}^{\mathsf{FO}}, r)$ that can be outputted by $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$. We denote them by $(\mathsf{pk}_{\mathsf{R},i}, \mathsf{sk}_{\mathsf{R},i}^{\mathsf{FO}} = (\mathsf{sk}_{\mathsf{R},i}, \mathsf{s}_{\mathsf{R},i}), r_i)$ for $0 \leq i \leq q_{\mathsf{leak}} - 1$. Moreover, we modify the construction of $\mathsf{H}_{\mathsf{rand}}$. When the input is of the form $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, m_\mathsf{L}, m_\mathsf{R})$, the computation is made by an internal oracle $\underline{\mathsf{H}}^i_{\mathsf{Rand}}(m_\mathsf{L}, m_\mathsf{R})$ with range $\mathsf{Rand}_\mathsf{L} \times \mathsf{Rand}_\mathsf{R} \times \mathsf{Rand}_\mathsf{C}$, not directly accessible by the adversary.[27] The advantage remains unchanged.

Game G_2. The oracle $\mathsf{H}_{\mathsf{rej}}$ used by $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ is replaced by a new oracle $\underline{\mathsf{H}}_{\mathsf{Rej}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_\mathsf{R}, \mathsf{s}_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$ that no longer takes the secret s_L as input. Moreover, $\underline{\mathsf{H}}_{\mathsf{Rej}}$ is not accessible by the adversary. Using Lemma 5, we obtain:

$$\left|\mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})\right| \leq 2q_{\mathsf{rej}} 2^{-\nu/2}.$$

Game G_3. We replace the random oracle $\mathsf{H}_{\mathsf{rand}}$ by an oracle which samples only "good" randomness for any couple of public keys $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i})$, for $0 \leq i \leq q_{\mathsf{leak}} - 1$, meaning that no decryption failures can occur when using $\mathsf{Dec}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{sk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{R},i}, \cdot)$ on proper ciphertexts of the form $\mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, \cdot)$.

More precisely, for each $0 \leq i \leq q_{\mathsf{leak}} - 1$ and each $m_\mathsf{L}, m_\mathsf{R}$ we define $\mathcal{R}^i_{\mathsf{bad}}(m_\mathsf{L}, m_\mathsf{R})$ as the set:

$$\Big\{\mathsf{r} = (r_\mathsf{L}, r_\mathsf{R}, r_\mathsf{C}) \in \mathsf{Rand}_\mathsf{L} \times \mathsf{Rand}_\mathsf{R} \times \mathsf{Rand}_\mathsf{C} : \\ \mathsf{Dec}_{\mathsf{OTP}}\Big(\mathsf{pp}, \mathsf{sk}_\mathsf{L}, \mathsf{sk}_{\mathsf{R},i}, \mathsf{Enc}_{\mathsf{OTP}}\big(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, m_\mathsf{L}, m_\mathsf{R}; \mathsf{r}\big)\Big) \neq (m_\mathsf{L}, m_\mathsf{R})\Big\}.$$

In G_3, for each $0 \leq i \leq q_{\mathsf{leak}} - 1$, we replace the internal random oracle $\underline{\mathsf{H}}^i_{\mathsf{Rand}}$ by an oracle that, on query $(m_\mathsf{L}, m_\mathsf{R})$, uniformly samples an output in $\mathsf{Rand}_\mathsf{L} \times \mathsf{Rand}_\mathsf{R} \times \mathsf{Rand}_\mathsf{C} \setminus \mathcal{R}^i_{\mathsf{bad}}(m_\mathsf{L}, m_\mathsf{R})$.

Using Lemma 2 with exactly the same reasoning as in Game 4 of the proof of [23, Theorem 3.2], we deduce that:

$$\left|\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})\right| \leq 8q_{\mathsf{leak}}\delta(q_{\mathsf{rand}} + q_{\mathsf{CCA}} + 1)^2.$$

Game G_4. In G_4, we modify the construction of $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$ when $\mathsf{pk}_\mathsf{L} = \mathsf{pk}_{\mathsf{L},\star}$ and $\mathsf{pk}_\mathsf{R} = \mathsf{pk}_{\mathsf{R},i}$ for some $0 \leq i \leq q_{\mathsf{leak}} - 1$. More precisely, for any tuple $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$

[26] Hence, p_{Find} denotes the probability that the event FIND occurs during the execution of $\mathcal{A}^{|\mathcal{O}_2 \setminus \mathcal{S}\rangle}(z)$, where $\mathcal{S}$, z, and $\mathcal{O}_2$ are sampled according to their respective distributions.

[27] We underline all oracles that are not directly accessible by the adversary.

[IND-CK-CCA, -] and [IND-CCA, -]

$\mathsf{pp} \leftarrow \mathsf{Setup}()$
$(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{L},\star} = (\mathsf{sk}_{\mathsf{L},\star}, \mathsf{s}_{\mathsf{L},\star})) \leftarrow \mathsf{dKeygen}^{\mathsf{FO}}_{\mathsf{L}}(\mathsf{pp})$
for $i \in \mathsf{Range}(\text{eph-key})$: / G_1 – G_5
  $r_i \leftarrow\!\$\ \mathsf{RandKeygenR}$ / G_1 – G_5
  $(\mathsf{pk}_{\mathsf{R},i}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{R},i} = (\mathsf{sk}_{\mathsf{R},i}, \mathsf{s}_{\mathsf{R},i})) \leftarrow \mathsf{dKeygen}^{\mathsf{FO}}_{\mathsf{R}}(r)$ / G_1 – G_5
$\mathsf{cpt}_{\mathsf{leak}} \leftarrow 0$ / G_1 – G_5
$\mathsf{LeakedR} = \emptyset$ / G_0
$\mathsf{Oracles} = \{\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}, \mathcal{O}_{\mathsf{leak}_\mathsf{R}}, |\mathsf{H}_{\mathsf{rej}}\rangle, |\mathsf{H}_{\mathsf{rand}}\rangle, |\mathsf{H}_{\mathsf{key}}\rangle, |\mathsf{H}_{\mathsf{OTP}}\rangle\}$
$(\mathsf{state}, \mathsf{pk}_{\mathsf{R},\star}) \leftarrow \mathcal{A}_1^{\mathsf{Oracles}}(\mathsf{pk}_{\mathsf{L},\star})$
$b \leftarrow\!\$\ \{0,1\}$
$(\mathsf{ct}_\star, \mathsf{k}_0) \leftarrow \mathsf{dEncaps}^{\mathsf{FO}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star})$
We parse $\mathsf{ct}_\star$ as $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star})$
We denote by $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$ and by
$(r_{\mathsf{C},\star}, r_{\mathsf{L},\star}, r_{\mathsf{R},\star})$ the messages and
randomnesses used to produce
this encapsulation.
$\mathsf{k}_1 \leftarrow\!\$\ \mathsf{Keys}$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Oracles}}(\mathsf{state}, \mathsf{ct}_\star, \mathsf{k}_b)$
return $b = b'$

$\mathcal{O}_{\mathsf{leak}_\mathsf{R}}()$

$r \leftarrow\!\$\ \mathsf{RandKeygenR}$ / G_0
$(\mathsf{pk}_\mathsf{R}, \mathsf{sk}^{\mathsf{FO}}_\mathsf{R} = (\mathsf{sk}_\mathsf{R}, \mathsf{s}_\mathsf{R})) \leftarrow \mathsf{dKeygen}^{\mathsf{FO}}_{\mathsf{R}}(r)$ / G_0
$\mathsf{LeakedR} \leftarrow \mathsf{LeakedR} \cup \{(\mathsf{pk}_\mathsf{R}, \mathsf{sk}^{\mathsf{FO}}_\mathsf{R}, r)\}$ / G_0
$\mathsf{LeakedpkR} \leftarrow \mathsf{LeakedpkR} \cup \{\mathsf{pk}_\mathsf{R}\}$ / G_0
return $(\mathsf{pk}_\mathsf{R}, \mathsf{sk}^{\mathsf{FO}}_\mathsf{R}, r)$ / G_0
$\mathsf{cpt}_{\mathsf{leak}} \leftarrow \mathsf{cpt}_{\mathsf{leak}} + 1$ / G_1 – G_5
return $(\mathsf{pk}_{\mathsf{R},\mathsf{cpt}_{\mathsf{leak}}-1}, \mathsf{sk}_{\mathsf{R},\mathsf{cpt}_{\mathsf{leak}}-1}, r_{\mathsf{cpt}_{\mathsf{leak}}-1})$ / G_1 – G_5

$\mathsf{H}_{\mathsf{OTP}}, \mathsf{H}_{\mathsf{rej}}, \underline{\mathsf{H}}_{\mathsf{Rand}}$

Random oracles

$\mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})$

Computed as a random oracle / G_0
if $\mathsf{pk}_\mathsf{L} = \mathsf{pk}_{\mathsf{L},\star} \wedge \exists i : \mathsf{pk}_\mathsf{R} = \mathsf{pk}_{\mathsf{R},i}$ **then** / G_1 – G_5
  return $\underline{\mathsf{H}}^i_{\mathsf{Rand}}(m_\mathsf{L}, m_\mathsf{R})$ / G_1 – G_5
else return $\underline{\mathsf{H}}_{\mathsf{Rand}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})$ / G_1 – G_5

$\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}(\mathsf{pk}_\mathsf{R}, \mathsf{ct} = (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R}))$

// with $\mathsf{ct}_\star$, LeakedR and $\mathsf{sk}_{\mathsf{L},\star}$
// taken from the security game
if $(\exists \mathsf{sk}^{\mathsf{FO}}_\mathsf{R} = (\mathsf{sk}_\mathsf{R}, \mathsf{s}_\mathsf{R}), r : (\mathsf{pk}_\mathsf{R}, \mathsf{sk}^{\mathsf{FO}}_\mathsf{R}, r) \in \mathsf{LeakedR})$
  $\wedge (\mathsf{ct} \neq \mathsf{ct}_\star \vee \mathsf{pk}_\mathsf{R} \neq \mathsf{pk}_{\mathsf{R},\star})$ **then**
  $\widetilde{\mathsf{k}} \leftarrow \mathsf{H}_{\mathsf{rej}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, \mathsf{s}_\mathsf{L}, \mathsf{s}_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ / G_0 – G_1
  $\widetilde{\mathsf{k}} \leftarrow \underline{\mathsf{H}}_{\mathsf{Rej}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, \mathsf{s}_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ / G_2 – G_5
  return $\widetilde{\mathsf{k}}$ / G_4 – G_5
  $(m_\mathsf{L}, m_\mathsf{R}) \leftarrow \mathsf{Dec}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{sk}_\mathsf{L}, \mathsf{sk}_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ / G_0 – G_3
  if $(m_\mathsf{L}, m_\mathsf{R}) = (\perp, \perp)$ **then** / G_0 – G_3
    return $\widetilde{\mathsf{k}}$ / G_0 – G_3
  if $\mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})$
    $\neq (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ **then** / G_0 – G_3
    return $\widetilde{\mathsf{k}}$ / G_0 – G_3
  return $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ / G_0 – G_3

$\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$

if $\exists \mathsf{k} : (\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R}, \mathsf{k}) \in \mathcal{L}_{\mathsf{H}_{\mathsf{key}}}$ **then**
  return k
$\mathsf{k} \leftarrow\!\$\ \mathsf{Keys}$
if $(\mathsf{pk}_\mathsf{L} = \mathsf{pk}_{\mathsf{L},\star}) \wedge (\exists i : \mathsf{pk}_\mathsf{R} = \mathsf{pk}_{\mathsf{R},i})$ **then** / G_4 – G_5
  $(c'_\mathsf{C}, c'_\mathsf{L}, \widetilde{c}'_\mathsf{R}) = \mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, \mathsf{pk}_{\mathsf{R},i}, m_\mathsf{L}, m_\mathsf{R})$ / G_4 – G_5
  if $(c'_\mathsf{C}, c'_\mathsf{L}, \widetilde{c}'_\mathsf{R}) = (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ **then** / G_4 – G_5
    $\mathsf{k} = \underline{\mathsf{H}}_{\mathsf{Rej}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, \mathsf{s}_{\mathsf{R},i}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$ / G_4 – G_5
$\mathcal{L}_{\mathsf{H}_{\mathsf{key}}} = \{(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R}, \mathsf{k})\} \cup \mathcal{L}_{\mathsf{H}_{\mathsf{key}}}$
return k

$\underline{\mathsf{H}}^i_{\mathsf{Rand}}(m_\mathsf{L}, m_\mathsf{R})$ for $0 \leq i \leq \text{eph-key} - 1$

Computed as a random oracle / G_1 , G_5
if input already queried **then** / G_3 – G_4
  return the same result / G_3 – G_4
else then / G_3 – G_4
  $x \leftarrow\!\$\ \mathsf{Rand}_{\mathsf{CLR}} \setminus \mathcal{R}^i_{\mathsf{bad}}(m_\mathsf{L}, m_\mathsf{R})$ / G_3 – G_4
  record and output x / G_3 – G_4

Fig. 14: Games G_0 to G_5 for the proof of [IND-CK-CCA, -] security in the QROM (Theorem 3).

such that $\mathsf{Enc}_{\mathsf{OTP}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, m_\mathsf{L}, m_\mathsf{R}) = (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$, we set $\mathsf{H}_{\mathsf{key}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_{\mathsf{R},i}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R}) = \underline{\mathsf{H}}_{\mathsf{Rej}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, \mathsf{s}_{\mathsf{R},i}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c^{\mathsf{OTP}}_\mathsf{R})$.

Thanks to this new definition of $\underline{\mathsf{H}}_{\mathsf{key}}$, we can rewrite the oracle $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ using only $\underline{\mathsf{H}}_{\mathsf{Rej}}$.

Due to the modification brought by G_3, for any tuple $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$ such that $(\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}}) = \mathsf{Enc}_{\mathsf{OTP}}\big(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, \mathsf{pk}_{\mathsf{R},i}, m_\mathsf{L}, m_\mathsf{R}\big)$, we have $(m_\mathsf{L}, m_\mathsf{R}) = \mathsf{Dec}_{\mathsf{OTP}}\left(\mathsf{pp}, \mathsf{sk}_\mathsf{L}, \mathsf{sk}_{\mathsf{R},i}, (\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})\right)$. As a result, the function transforming tuples $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$ to $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},i}, \mathsf{s}_{\mathsf{R},i}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$ is injective. Moreover, $\underline{\mathsf{H}}_{\mathsf{Rej}}$ was never used on such tuples. As a result, the advantage does not change between G_3 and G_4.

Game G_5. We make the reverse operation of game G_3 by returning to the initial oracle $\mathsf{H}_{\mathsf{rand}}$ prior of game G_5. Using the same argument as in G_3, we obtain:

$$\left|\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_5}(\mathcal{A})\right| \leq 8q_{\mathsf{leak}}\delta(q_{\mathsf{rand}} + q_{\mathsf{CCA}} + 1)^2 \,.$$

Game G_6. The left challenge message $m_{\mathsf{L},\star}$ is used by three random oracles $\mathsf{H}_{\mathsf{rand}}$, $\mathsf{H}_{\mathsf{OTP}}$ and $\mathsf{H}_{\mathsf{key}}$. In order to use the *one-way to hiding* Lemma 3, we modify them in order that the message $m_{\mathsf{L},\star}$ will be queried to only one random oracle.

We introduce an internal random oracle $\underline{\mathcal{H}}_{\mathsf{OWTH}}$ that, on query $(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$ (same input set than $\mathsf{H}_{\mathsf{key}}$), outputs a triplet $((r'_\mathsf{L}, r'_\mathsf{R}, r'_\mathsf{C}), c'_\mathsf{C}, \mathsf{k}')$ where:

- $(r'_\mathsf{L}, r'_\mathsf{R}, r'_\mathsf{C})$ only depends on $(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})$ (modifying the other arguments $(\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$ does not change its value).
- c'_C only depends on m_L (modifying the other arguments $(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$ does not change its value).

Then, in G_6, we use $\underline{\mathcal{H}}_{\mathsf{OWTH}}$ to simulate the three oracles $\mathsf{H}_{\mathsf{rand}}, \mathsf{H}_{\mathsf{OTP}}$ and $\mathsf{H}_{\mathsf{key}}$. For example, if the adversary queries $\mathsf{H}_{\mathsf{rand}}$ on input $(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R})$, we add any three values $\mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}}$ to them, query $((r'_\mathsf{L}, r'_\mathsf{R}, r'_\mathsf{C}), c'_\mathsf{C}, \mathsf{k}') = \underline{\mathcal{H}}_{\mathsf{OWTH}}(\mathsf{pk}_\mathsf{L}, \mathsf{pk}_\mathsf{R}, m_\mathsf{L}, m_\mathsf{R}, \mathsf{c}_\mathsf{C}, \mathsf{c}_\mathsf{L}, c_\mathsf{R}^{\mathsf{OTP}})$ and answer $(r'_\mathsf{L}, r'_\mathsf{R}, r'_\mathsf{C})$.

The distribution of any element is unchanged, thus the advantage remains unchanged.

Game G_7. Informally, in G_7, we decouple the oracle value returned by $\underline{\mathcal{H}}_{\mathsf{OWTH}}$ when queried with a message containing the challenge plaintext $m_{\mathsf{L},\star}$ from the values that are used to compute the challenge ciphertext, the challenge keys and the OTP added to c_R to obtain $c_\mathsf{R}^{\mathsf{OTP}}$.

More formally, let us consider the set $\mathcal{X} = \mathsf{PK}_\mathsf{L} \times \mathsf{PK}_\mathsf{R} \times \mathcal{M}_\mathsf{L} \times \mathcal{M}_\mathsf{R} \times \mathsf{Ciphers}_\mathsf{C} \times \mathsf{Ciphers}_\mathsf{L} \times \mathsf{Ciphers}_\mathsf{R}$ and $\mathcal{S}$ the subset of elements of $\mathcal{X}$ where the left message is equal to the challenge left message $m_{\mathsf{L},\star}$. In G_7, we reprogram $\underline{\mathcal{H}}_{\mathsf{OWTH}}$ on any value of $\mathcal{S}$ with uniform values. As a result, in G_7, $(r_{\mathsf{C},\star}, r_{\mathsf{L},\star}, r_{\mathsf{R},\star})$, k_1 and $c_{\mathsf{R},\star}^{\mathsf{OTP}}$ follow a uniform distribution independent of the outputs of $\underline{\mathcal{H}}_{\mathsf{OWTH}}$, and thus independent of any value accessible by $\mathcal{A}$. Using [23, Lemma 5.2], we have:

$$\left|\mathsf{Adv}^{\mathsf{G}_6}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_7}(\mathcal{A})\right| \leq 2\sqrt{(\hat{q}+1)\cdot \Pr\left[b \leftarrow \mathcal{C}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}(m_{\mathsf{L},\star}) : \mathsf{FIND}\right]}\,, \tag{6}$$

where:

- $\hat{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}} + q_{\mathsf{CCA}}$.
- $\mathcal{C}$ runs the game G_7 with adversary $\mathcal{A}$ and the punctured oracle $|\underline{\mathcal{H}}_{\mathsf{OWTH}} \setminus \mathcal{S}\rangle$ instead of $|\underline{\mathcal{H}}_{\mathsf{OWTH}}\rangle$. Moreover, $m_{\mathsf{L},\star}$ is considered as an input of $\mathcal{C}$, whose distribution is uniform over $\mathcal{M}_\mathsf{L}$.

Because the distribution of $\mathcal{S}$ is not independent from $m_{\mathsf{L},\star}$, we cannot use Lemma 4 to bound $\Pr\left[b \leftarrow \mathcal{C}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}(m_{\mathsf{L},\star}) : \mathsf{FIND}\right]$.

In order to bound $\Pr\Big[b \leftarrow \mathcal{C}^{\mathcal{O}_\mathcal{S}^{\mathsf{SC}}}(z) : \mathsf{FIND}\Big]$, we introduce game G_8 whose only difference with G_7 is that $c_{\mathsf{L},\star}$ is computed as $\mathsf{Encr}_\mathsf{L}(\mathsf{pk}_{\mathsf{L},\star}, m_{\mathsf{L},\mathsf{Fixed}}, r_{\mathsf{L},\star})$ instead of $\mathsf{Encr}_\mathsf{L}(\mathsf{pk}_{\mathsf{L},\star}, m_{\mathsf{L},\star}, r_{\mathsf{L},\star})$, where $m_{\mathsf{L},\mathsf{Fixed}}$ is any fixed left message.

We note $\mathcal{D}$ the algorithm with no input, that runs the game G_8 with adversary $\mathcal{A}$ and with the punctured oracle $|\underline{\mathcal{H}}_{\mathsf{OWTH}} \setminus \mathcal{S}\rangle$. There is clearly independence between the (empty) input of $\mathcal{D}$ and the set $\mathcal{S}$, so we can use Lemma 4 and obtain:

$$\Pr\Big[\mathcal{D}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}() : \mathsf{FIND}\Big] \leq 4\hat{q}|\mathcal{S}|/|\mathcal{X}| = 4\hat{q}/|\mathcal{M}_\mathsf{L}|\,, \tag{7}$$

where $\hat{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}} + q_{\mathsf{CCA}}$.

Now, we study the difference of advantages between G_7 and G_8. We will bound it by the advantage of an adversary $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ against the IND-CPA game of PKE_L.

First, $\mathcal{E}_1$ obtains public parameters pp, public key $\mathsf{pk}_{\mathsf{L},\star}$ and then gives two messages $(m_{\mathsf{L},\mathsf{Fixed}}, m_{\mathsf{L},\star})$ to the challenger, $m_{\mathsf{L},\star}$ being uniformly sampled and $m_{\mathsf{L},\mathsf{Fixed}}$ being any fixed left message. We denote by $\tilde{b}$ the bit sampled by the challenger of the game IND-CPA of PKE_L before encrypting either $m_{\mathsf{L},\mathsf{Fixed}}$ if $\tilde{b} = 0$ or $m_{\mathsf{L},\star}$ if $\tilde{b} = 1$.

Then, $\mathcal{E}_2$ receives $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star})$ and needs to decide if it is an encryption of $m_{\mathsf{L},\mathsf{Fixed}}$ or $m_{\mathsf{L},\star}$. $\mathcal{E}_2$ uses $(\mathsf{pp}, \mathsf{pk}_\mathsf{L}, c_{\mathsf{C},\star}, c_{\mathsf{L},\star})$ to simulate G_7 to $\mathcal{A}$, this includes the simulation of each random oracle. However, $\mathcal{E}_2$

uses $|\underline{\mathcal{H}}_{\mathsf{OWTH}} \setminus \mathcal{S}\rangle$ instead of $|\underline{\mathcal{H}}_{\mathsf{OWTH}}\rangle$. Finally, $\mathcal{E}_2$ outputs 1 if FIND happens during the simulation, and 0 else.

By definition of $\mathcal{E}$, G_7 and G_8 we see that, until FIND happens:
- $\mathcal{E}$ makes a perfect simulation of $\mathcal{C}$ (with the punctured oracle) if $\tilde{b} = 1$.
- $\mathcal{E}$ makes a perfect simulation of $\mathcal{D}$ (with the punctured oracle) if $\tilde{b} = 0$.

Thus, we have

$$\Pr\left[\mathcal{C}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}(m_{\mathsf{L},\star}) : \mathsf{FIND}\right] \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}(\mathcal{E}) + \Pr\left[\mathcal{D}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}() : \mathsf{FIND}\right]. \tag{8}$$

Using Equations (8) and (7), we obtain:

$$\Pr\left[\mathcal{C}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}(m_{\mathsf{L},\star}) : \mathsf{FIND}\right] \leq 2\,\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}(\mathcal{B}) + 4\hat{q}/|\mathcal{M}_{\mathsf{L}}|\,, \tag{9}$$

where $\hat{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}} + q_{\mathsf{CCA}}$.

Finally, we note that in G_8, both k_0 and k_1 follow independent uniform distributions, and thus, $\mathsf{Adv}^{\mathsf{G}_7}(\mathcal{A}) = 0$. Using this as well as Equations (6) and (9), we have:

$$\mathsf{Adv}^{\mathsf{G}_7}(\mathcal{A}) \leq 2\sqrt{(\hat{q}+1)\left(2\,\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}_{\mathsf{L}}}(\mathcal{B}) + 4\hat{q}/|\mathcal{M}_{\mathsf{L}}|\right)}\,,$$

where $\hat{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{OTP}} + q_{\mathsf{CCA}}$ and this concludes the proof.

Proof of the QROM case, [- , IND-CCA] security

Proof. **Game G_0.** This is the initial [- , IND-CCA] game. We use similar notations as for the proof of the [IND-CCA, -] security. $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star})$ are the left challenge public and secret keys computed with randomness $\mathsf{r}_{\mathsf{L},\star}$, while $(\mathsf{pk}_{\mathsf{R},\star}, \mathsf{sk}_{\mathsf{R},\star})$ are the right ones. The ciphertext and key from $\mathsf{dEncaps}^{\mathsf{FO}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star})$, computed by the challenger, form the pair $(\mathsf{ct}_\star = (c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star}), \mathsf{k}_0)$, whereas k_1 is the key uniformly sampled from Keys by the challenger. We note $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$ the messages sampled to compute the challenge ciphertexts and $(r_{\mathsf{C},\star}, r_{\mathsf{L},\star}, r_{\mathsf{R},\star}) = \mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$. b is the challenge bit and b' is the response provided by the adversary. We also recall that $\mathsf{LeakedR}$ is the set of triplets $(\mathsf{pk}_{\mathsf{L}}, \mathsf{sk}^{\mathsf{FO}}_{\mathsf{L}} = (\mathsf{sk}_{\mathsf{L}}, \mathsf{s}_{\mathsf{L}}), r)$ outputted by the oracle $\mathcal{O}_{\mathsf{leak}_{\mathsf{L}}}$.

We introduce games $\mathsf{G}_1, \cdots, \mathsf{G}_5$ that apply the same modifications as the ones with same name used for [IND-CK-CCA, -] security, with only two differences:

- We invert the left and right elements.
- In $\mathsf{G}_3, \mathsf{G}_4$ and G_5, we do not only apply the changes for any couple of public keys $(\mathsf{pk}_{\mathsf{L},i}, \mathsf{pk}_{\mathsf{R},\star})$ for $0 \leq i \leq q_{\mathsf{leak}} - 1$, but also include the couple $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star})$,

and right elements.

Using the same reasoning as with [IND-CK-CCA, -] security, we obtain that

$$\left|\mathsf{Adv}^{\mathsf{G}_0}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_5}(\mathcal{A})\right| \leq 2q_{\mathsf{rej}}2^{-\nu/2} + 16\,(q_{\mathsf{leak}}+1)\,\delta(q_{\mathsf{rand}} + q_{\mathsf{CCA}} + 1)^2.$$

Game G_6. The right challenge message $m_{\mathsf{R},\star}$ is used by the two random oracles $\mathsf{H}_{\mathsf{rand}}$ and $\mathsf{H}_{\mathsf{key}}$. In order to use the One-Way to Hiding Lemma 3, we modify them so that the message $m_{\mathsf{R},\star}$ will be asked to only one random oracle.

We introduce an internal random oracle $\underline{\mathcal{H}}_{\mathsf{OWTH}}$ that, on input $(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ (same input set as $\mathsf{H}_{\mathsf{key}}$), outputs a triplet $((r'_{\mathsf{L}}, r'_{\mathsf{R}}, r'_{\mathsf{C}}), \mathsf{k}')$ where $(r'_{\mathsf{C}}, r'_{\mathsf{L}}, r'_{\mathsf{R}})$ only depends on $(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}})$ (changing the other arguments $(\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ does not change its value).

Then, in G_6, we use $\underline{\mathcal{H}}_{\mathsf{OWTH}}$ to simulate the oracles $\mathsf{H}_{\mathsf{rand}}$ and $\mathsf{H}_{\mathsf{key}}$. For example, if the adversary makes a query to $\mathsf{H}_{\mathsf{rand}}$ with input $(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}})$, we add any three values $\mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}}$ to them, query $((r'_{\mathsf{C}}, r'_{\mathsf{L}}, r'_{\mathsf{R}}), c'_{\mathsf{C}}, \mathsf{k}') = \underline{\mathcal{H}}_{\mathsf{OWTH}}(\mathsf{pk}_{\mathsf{L}}, \mathsf{pk}_{\mathsf{R}}, m_{\mathsf{L}}, m_{\mathsf{R}}, \mathsf{c}_{\mathsf{C}}, \mathsf{c}_{\mathsf{L}}, c^{\mathsf{OTP}}_{\mathsf{R}})$ and output $(r'_{\mathsf{C}}, r'_{\mathsf{L}}, r'_{\mathsf{R}})$.

The distribution of any element is unchanged, thus the advantage remains unchanged.

Game G_7. In this game, we decouple the values of k_0 and $(r_{\mathsf{C},\star}, r_{\mathsf{L},\star}, r_{\mathsf{R},\star})$, usually computed as $\mathsf{k}_0 = \mathsf{H}_{\mathsf{key}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_{\mathsf{R},\star}, c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c^{\mathsf{OTP}}_{\mathsf{R},\star})$ and $\mathsf{H}_{\mathsf{rand}}(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$, from the oracles $\mathsf{H}_{\mathsf{key}}$ and $\mathsf{H}_{\mathsf{rand}}$.

Let us consider the set $\mathcal{X} = \mathsf{PK}_{\mathsf{L}} \times \mathsf{PK}_{\mathsf{R}} \times \mathcal{M}_{\mathsf{L}} \times \mathcal{M}_{\mathsf{R}} \times \mathsf{Ciphers}_{\mathsf{C}} \times \mathsf{Ciphers}_{\mathsf{L}} \times \mathsf{Ciphers}_{\mathsf{R}}$ and $\mathcal{S}$ the subset of elements of $\mathcal{X}$ where the right message is equal to the challenge right message $m_{\mathsf{R},\star}$. In G_7, we reprogram $\underline{\mathcal{H}}_{\mathsf{OWTH}}$ on any value of $\mathcal{S}$ with uniform values.

As a result, in G_7, $(r_{\mathsf{C},\star}, r_{\mathsf{L},\star}, r_{\mathsf{R},\star})$, k_0 and $c^{\mathsf{OTP}}_{\mathsf{R},\star}$ have a uniform distribution, independent of the outputs of $\underline{\mathcal{H}}_{\mathsf{OWTH}}$, and thus independent of any value accessible by $\mathcal{A}$. Using [23, Lemma 5.2], we have:

$$\Big|\mathsf{Adv}^{\mathsf{G}_6}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_7}(\mathcal{A})\Big| \leq 2\sqrt{(\hat{q}+1)\Pr\big[b \leftarrow \mathcal{C}^{|\mathsf{H}_{\mathsf{key}}\setminus\mathcal{S}\rangle}(m_{\mathsf{L},\star}) : \mathsf{FIND}\big]}, \tag{10}$$

where:
- $\hat{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{CCA}}$.
- $\mathcal{C}$ runs the game G_7 with adversary $\mathcal{A}$ and punctured oracle $|\underline{\mathcal{H}}_{\mathsf{OWTH}} \setminus S\rangle$ instead of $|\underline{\mathcal{H}}_{\mathsf{OWTH}}\rangle$. Moreover, $m_{\mathsf{R},\star}$ is uniformly sampled in $\mathcal{M}_{\mathsf{R}}$.

Note that the distribution of $\mathcal{S}$ is dependent of the input $m_{\mathsf{R},\star}$. Thus, we cannot use Lemma 4 to bound $\Pr\big[b \leftarrow \mathcal{C}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}(m_{\mathsf{R},\star}) : \mathsf{FIND}\big]$.

In order to bound $\Pr\big[b \leftarrow \mathcal{C}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}(z) : \mathsf{FIND}\big]$, we introduce a new game G_8 whose only difference with G_7 is that $c_{\mathsf{R},\star}$ is computed as $\mathsf{Encr}_{\mathsf{R}}(\mathsf{pk}_{\mathsf{L},\star}, m_{\mathsf{R},\mathsf{Fixed}}, r_{\mathsf{L},\star})$ instead of $\mathsf{Encr}_{\mathsf{R}}(\mathsf{pk}_{\mathsf{L},\star}, m_{\mathsf{R},\star}, r_{\mathsf{L},\star})$.

We note $\mathcal{D}$ the algorithm with no input, that runs the game G_8 with adversary $\mathcal{A}$ and with the punctured oracle $|\underline{\mathcal{H}}_{\mathsf{OWTH}} \setminus \mathcal{S}\rangle$.

There is clearly independence between the (empty) input of $\mathcal{D}$ and the set $\mathcal{S}$, thus we can use the Lemma 4 to obtain:

$$\Pr\Big[\mathcal{D}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}() : \mathsf{FIND}\Big] \leq 4\hat{q}|S|/|X| = 4\hat{q}/|\mathcal{M}_{\mathsf{R}}|, \tag{11}$$

where $\hat{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{CCA}}$.

Now, we study the difference of advantages between G_7 and G_8. We will bound it by the advantage of an adversary $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ against the $[\,\text{-}\,, \mathsf{IND\text{-}CPA}]$ game of DPKE.

First, $\mathcal{E}_1$ obtains public parameters and keys $\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}$, then gives $(m_{\mathsf{R},\mathsf{Fixed}}, m_{\mathsf{R},\star}, m_{\mathsf{L},\star})$ to the challenger, $m_{\mathsf{R},\star}$ and $m_{\mathsf{L},\star}$ being uniformly sampled and $m_{\mathsf{R},\mathsf{Fixed}}$ being any fixed right message. We denote by $\tilde{b}$ the bit sampled by the challenger of the $[\,\text{-}\,, \mathsf{IND\text{-}CPA}]$ game of DPKE to choose to encrypt $(m_{\mathsf{L},\star}, m_{\mathsf{R},\mathsf{Fixed}})$ (if $\tilde{b} = 0$) or $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$ (if $\tilde{b} = 1$).

Then, $\mathcal{E}_2$ receives $(c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star})$ and needs to decide if it is an encryption of $(m_{\mathsf{L},\star}, m_{\mathsf{R},\mathsf{Fixed}})$ or $(m_{\mathsf{L},\star}, m_{\mathsf{R},\star})$. $\mathcal{E}_2$ uses $(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star}, c_{\mathsf{C},\star}, c_{\mathsf{L},\star}, c_{\mathsf{R},\star})$ to make a simulation of G_7 to $\mathcal{A}$, this simulation including the simulation of each the random oracles. However, $\mathcal{E}_2$ uses $|\underline{\mathcal{H}}_{\mathsf{OWTH}} \setminus \mathcal{S}\rangle$ instead of $|\underline{\mathcal{H}}_{\mathsf{OWTH}}\rangle$.

Finally, $\mathcal{E}_2$ outputs 1 if FIND happens during the simulation, and 0 else.

By definition of $\mathcal{E}$, G_7 and G_8 we see that, until FIND happens (if it happens):
- $\mathcal{E}$ make a perfect simulation of $\mathcal{C}$ (with the punctured oracle) if $\tilde{b} = 1$.
- $\mathcal{E}$ make a perfect simulation of $\mathcal{D}$ (with the punctured oracle) if $\tilde{b} = 0$.

We thus have:

$$\begin{aligned}\Pr\Big[\mathcal{C}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}(m_{\mathsf{L},\star}) : \mathsf{FIND}\Big] \leq\ & 2\,\mathsf{Adv}^{[\,\text{-}\,,\mathsf{IND\text{-}CPA}]}(\mathcal{E}) \\ & + \Pr\Big[\mathcal{D}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}() : \mathsf{FIND}\Big].\end{aligned} \tag{12}$$

Using Equations (12) and (11), we obtain:

$$\Pr\Big[\mathcal{C}^{|\underline{\mathcal{H}}_{\mathsf{OWTH}}\setminus\mathcal{S}\rangle}(m_{\mathsf{L},\star}) : \mathsf{FIND}\Big] \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}(\mathcal{B}) + 4\hat{q}/|\mathcal{M}_{\mathsf{R}}|\,, \tag{13}$$

where $\hat{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{CCA}}$.

Finally, we note that in G_8, both k_0 and k_1 have an independent uniform distribution, and thus, $\mathsf{Adv}^{\mathsf{G}_7}(\mathcal{A}) = 0$.

Using this as well as Equations (10) and (13), we have:

$$\mathsf{Adv}^{\mathsf{G}_7}(\mathcal{A}) \leq 2\sqrt{(\hat{q}+1)\Big(2\,\mathsf{Adv}^{[\,\text{-}\,,\mathsf{IND\text{-}CPA}]}(\mathcal{B}) + 4\hat{q}/|\mathcal{M}_{\mathsf{R}}|\Big)}.$$

where $\hat{q} = q_{\mathsf{rand}} + q_{\mathsf{key}} + q_{\mathsf{CCA}}$. This concludes the proof.

C The eCK Model and its Variants

Sections C.1 and C.2 introduce a generalization of the eCK model [40], which is then used in Section C.3 to redefine the $\mathsf{eCK}^{\mathsf{w}}$ and $\mathsf{eCK\text{-}PFS}$ models from [17]. Finally, Section C.4 presents and justifies the unilateral variants of $\mathsf{eCK}^{\mathsf{w}}$ and $\mathsf{eCK\text{-}PFS}$.

C.1 Modelization of Protocol, Participants and Adversarial Queries

Preliminary Notations about Finite Sequences. For any set U, denote by $\mathsf{FiniteSequences}(U)$ the set of finite sequences of elements from U, which includes the empty sequence ϵ_{seq}. The length of a finite sequence s—its number of elements—is denoted by $|s|$. For a set U, a finite sequence $s = (s_i)_{i \leq n} \in \mathsf{FiniteSequences}(U)$, and an element $u \in U$, define $\mathsf{AddToSequence}(s, u)$ as the sequence $(u_i)_{i \leq n+1}$ where $s_{n+1} = u$.

Parties and Sessions. In the eCK model, parties represent the participants of the protocol. Each party P has a role $\mathsf{P}_{\mathsf{Role}} \in \{\mathsf{Init}, \mathsf{Resp}\}$ and possesses a pair of static public and secret keys $(\mathsf{P}_{\mathsf{spk}}, \mathsf{P}_{\mathsf{ssk}})$. These keys are generated by the algorithm $\mathsf{Keygen}_{\mathsf{static}}(\mathsf{pp}, \mathsf{Role})$ upon the party's creation, where the static public parameters pp are initially created by an algorithm $\mathsf{Setup}_{\mathsf{static}}$. Each party also has a unique identity, ensuring that no two parties have the same identity.

Each party can be associated to multiple instances of a protocol. Formally, a session s is defined as a tuple

$$\begin{aligned}&\big(\mathsf{InternId}, \mathsf{Actor}, \mathsf{Peer}, \mathsf{SessionKey}, \mathsf{Sent}, \mathsf{Rec}, \mathsf{Eph}, \mathsf{ProtocolState}, \mathsf{RunState}\big)\\ &\in \mathbb{N} \times \mathsf{SetParties}^2 \times (\mathsf{SetSessionKeys} \sqcup \{\bot\}) \times (\mathsf{FiniteSequences}(\{0,1\}^*))^3\\ &\times \{0,1\}^* \times \{\mathsf{running}, \mathsf{completed}, \mathsf{fail}\},\end{aligned}$$

where:

- $\mathsf{InternId}$ is an integer that uniquely identifies the session among other sessions, it is not related to the *sid* used in BPR model [7].
- Actor is a party called the actor of the session, it represents the party that is running the instance of the protocol.
- Peer is a party called the peer of the session whom the actor thinks, sometimes incorrectly, it is interacting with.
- $\mathsf{SessionKey}$ is the session key computed by the actor. It is either undefined and equal to $\bot$, or defined and belongs in the set $\mathsf{SetSessionKeys}$.
- Sent, Rec respectively denote the finite sequence of timely ordered messages sent and received by the Actor in this session.
- Eph denotes the sequence of elements that may be leaked to the adversary via the $\mathsf{eph\text{-}key}$ query defined below.[28] For instance, Eph could contain all randomness used by the protocol that is not securely erased.
- $\mathsf{ProtocolState}$ is an internal state needed to run the protocol. It contains any elements that can be needed for further computations. The following subsection provides a detailed explanation of how the protocol is modeled.
- $\mathsf{RunState}$ is the running state of the session. It indicates if the session is running ($\mathsf{running}$), complete ($\mathsf{completed}$) or if some error occurred (fail).

We will note:

$$\big(\mathsf{s}_{\mathsf{InternId}}, \mathsf{s}_{\mathsf{Actor}}, \mathsf{s}_{\mathsf{Peer}}, \mathsf{s}_{\mathsf{Sent}}, \mathsf{s}_{\mathsf{Rec}}, \mathsf{s}_{\mathsf{SessionKey}}, \mathsf{s}_{\mathsf{Eph}}, \mathsf{s}_{\mathsf{ProtocolState}}, \mathsf{s}_{\mathsf{RunState}}\big)$$

the attributes of a session s.

We call initiator (*resp.* responder) a party whose role is initiator (*resp.* responder). We call initiator session (*resp.* responder session) a session whose actor is an initiator (*resp.* responder).

As this will be shown by the definition of adversarial queries—more precisely, the send query—it is important to note that the elements received by a session will be decided by the adversary, whereas the elements created by a session will be computed according to the protocol's definition.

[28] Our definition is more general than the original eCK model (and its variants $\mathsf{eCK^w}$ and eCK-PFS), since it allows the protocol designer to specify which elements may be leaked through $\mathsf{eph\text{-}key}$. In contrast, the original model fixes this query to reveal exactly the ephemeral secret keys (i.e., the randomness coins) of session s. The choice of which elements can be leaked is significant and may have non-trivial consequences, as illustrated in [16].

Protocol. A protocol is modeled as the tuple $\Pi = (\pi, \mathsf{Setup}_{\mathsf{static}}, \mathsf{Keygen}_{\mathsf{static}})$, where $\mathsf{Setup}_{\mathsf{static}}$ and $\mathsf{Keygen}_{\mathsf{static}}$ are two probabilistic algorithms we have previously described, and π is a probabilistic algorithm that represents the computations performed by a party during a protocol execution. The algorithm π takes the following input:

$$(\mathsf{ProtocolState}, \mathsf{m}, \mathsf{actor}, \mathsf{peerId}, \mathsf{peerSpk}, \mathsf{Eph}) \in (\{0,1\}^*)^2 \times \mathsf{SetParties} \times (\{0,1\}^*)^2 \times \mathsf{FiniteSequences}\,(\{0,1\}^*)$$

and returns a tuple:

$$\left(\mathsf{RunState}', \mathsf{ProtocolState}', \mathsf{m}', \mathsf{k}^{\mathsf{SESSION}}, \mathsf{Eph}'\right) \in \{\mathsf{running}, \mathsf{completed}, \mathsf{fail}\} \times \{0,1\}^* \times \{0,1\}^* \times (\mathsf{SetSessionKeys} \sqcup \{\perp\}) \times \mathsf{FiniteSequences}\,(\{0,1\}^*)\,,$$

where:

- $\mathsf{ProtocolState}$ and $\mathsf{ProtocolState}'$ represent the internal state of the protocol.
- $\mathsf{RunState}$ indicates the resulting running state, *i.e.*, whether after this computation, the session will continue running, be complete, or be subject to an error[29].
- actor is the party which performs the computations—the actor. In particular, its static public and secret keys and its identity can be used in the computation.
- peerID and $\mathsf{peerSpk}$ represent the identity and static public key of the peer, *i.e.*, of the party the actor presumes it is interacting with.
- m is the received message.
- m' is the subsequent message to be sent.
- $\mathsf{k}^{\mathsf{SESSION}}$ is the session key computed by the protocol. We suppose that $\mathsf{k}^{\mathsf{SESSION}} \neq \perp$ if, and only if, $\mathsf{RunState}' = \mathsf{completed}$.
- Eph' denotes the updated sequence of elements that may be leaked to the adversary through the $\mathsf{eph\text{-}key}$ query. For instance, Eph could contain all randomness used by the protocol that is not securely erased.

Adversarial Capabilities. The adversary can control the actions of any party through multiple queries. Additionally, it can obtain certain information and perform a real-or-random test with the session key of a session of its choice—this session is thus referred to as the test session.

Let $\mathsf{cpt}_{\mathsf{party}}$ and $\mathsf{cpt}_{\mathsf{session}}$ be global counters that track the number of created parties and session—initialized to 0. Let b_{qtest} and $\mathsf{s}^{\mathsf{Test}}$ be variables both initialized to $\perp$. Here are the adversarial capabilities:

1. $\mathsf{activate\text{-}party}(\mathsf{Id}, \mathsf{Role})$, with $\mathsf{Id} \in \mathsf{SetIds}, \mathsf{Role} \in \{\mathsf{Init}, \mathsf{Resp}\}$, models the creation of a party. More precisely, if $\mathsf{activate\text{-}party}$ has already been used to create a party with identifier Id, it returns $\perp$. Else, it computes $(\mathsf{spk}, \mathsf{ssk}) \leftarrow \mathsf{Keygen}_{\mathsf{static}}(\mathsf{pp})$, creates party $\mathsf{P} = (\mathsf{cpt}_{\mathsf{party}}, \mathsf{Id}, \mathsf{Role}, \mathsf{spk}, \mathsf{ssk})$ and $\mathsf{cpt}_{\mathsf{party}}$ is incremented by one. Finally, it returns $(\mathsf{s}_{\mathsf{InternId}}(= \mathsf{cpt}_{\mathsf{session}} - 1), \mathsf{spk})$ and spk. Party P is said to be activated.
2. $\mathsf{activate\text{-}session}(\mathsf{Id}, \mathsf{Id}')$ models the creation of a session between the activated parties with identities Id—the actor—and Id'—the peer. More precisely, if one of these parties does not exist, it returns $\perp$. Else, it creates $\mathsf{s} = (\mathsf{cpt}_{\mathsf{session}}, \mathsf{P}[1], \mathsf{P}[2], \epsilon, \epsilon_{\mathsf{seq}}, \epsilon_{\mathsf{seq}}, \epsilon_{\mathsf{seq}}, \epsilon, \mathsf{running})$ where $\mathsf{P}[1], \mathsf{P}[2]$ are the parties with respective identities $\mathsf{Id}, \mathsf{Id}'$ and $\mathsf{cpt}_{\mathsf{session}}$ is incremented by one. Finally, it returns $\mathsf{s}_{\mathsf{InternId}}(= \mathsf{cpt}_{\mathsf{session}} - 1)$. Session s is said to be activated.
3. $\mathsf{send}(\mathsf{InternId}, \mathsf{m})$ models the sending of message m to an activated session with internal identifier $\mathsf{InternId}$. More precisely, if such session does not exist, it returns $\perp$.
   - If $\mathsf{RunState} \in \{\mathsf{fail}, \mathsf{completed}\}$, nothing is done, and it returns $\perp$.
   - Otherwise, $\mathsf{s}_{\mathsf{Rec}}$ is set to $\mathsf{AddToSequence}\,(\mathsf{m}, \mathsf{Rec})$, and: it computes:

$$\left(\mathsf{RunState}', \mathsf{ProtocolState}', \mathsf{m}', \mathsf{k}^{\mathsf{SESSION}}, \mathsf{Eph}'\right) = \pi(\mathsf{s}_{\mathsf{ProtocolState}}, \mathsf{m}, \mathsf{s}_{\mathsf{Actor}}, (\mathsf{s}_{\mathsf{Actor}})_{\mathsf{Id}}, (\mathsf{s}_{\mathsf{Peer}})_{\mathsf{spk}}, \mathsf{s}_{\mathsf{Eph}}).$$

   Then:
   - If $\mathsf{ProtocolState}' = \mathsf{fail}$, $\mathsf{s}_{\mathsf{RunState}}$ is set to fail and send returns fail.

[29] Note that we do not include a running state as input, because π will only be used for sessions whose running state is $\mathsf{running}$. However, the output must indicate the resulting running state.

- If $\mathsf{ProtocolState}' = \mathsf{running}$, $\mathsf{s}_{\mathsf{ProtocolState}}$ is set to state', $\mathsf{s}_{\mathsf{Eph}}$ is set to Eph', $\mathsf{s}_{\mathsf{Sent}}$ is set to $\mathsf{AddToSequence}(\mathsf{s}_{\mathsf{Sent}}, \mathsf{m}')$ and send returns m'.
- If $\mathsf{ProtocolState}' = \mathsf{completed}$, RunState is set to completed, Eph is set to Eph', SessionKey is set to $\mathsf{k}^{\mathsf{SESSION}}$ and send returns $(\mathsf{completed}, \mathsf{m}')$.

Note that in an AKE, the initiator does not need an input message to compute its first message: this is modeled by using send with the empty message ϵ.

4. corrupt(P). This query reveals the long-term secret keys of party P by returning $\mathsf{P}_{\mathsf{ssk}}$.
5. eph-key(s). If $\mathsf{s}_{\mathsf{RunState}} = \mathsf{fail}$, the query returns $\perp$. Otherwise, it outputs the sequence Eph of elements designated as leakable to the adversary (*e.g.*, all randomness used by the protocol that is not securely erased).
6. session-key(s). This query returns $\perp$ if the session s is not complete—*i.e.* $\mathsf{s}_{\mathsf{RunState}} \neq \mathsf{completed}$. Else, it returns its session key $\mathsf{s}_{\mathsf{SessionKey}}$.
7. test-session(s). This query returns $\perp$ if the session s is not complete—*i.e.* $\mathsf{s}_{\mathsf{RunState}} \neq \mathsf{completed}$—or if $\mathsf{s}^{\mathsf{Test}} \neq \perp$. Else, $\mathsf{s}^{\mathsf{Test}}$ is set to s and b_{qtest} is initialized to a uniform bit $b_{\mathsf{qtest}} \leftarrow\!\!\$\ \{0,1\}$. Then:
   - If $b_{\mathsf{qtest}} = 0$, $\mathsf{s}^{\mathsf{Test}}_{\mathsf{SessionKey}}$ is returned.
   - If $b_{\mathsf{qtest}} = 1$, a key $\mathsf{k} \leftarrow\!\!\$\ \mathsf{SetSessionKeys}$ is computed and returned.

Matching Sessions and Origin Session. The notion of matching sessions specifies when two sessions are supposed to be intended communication partners. Here we formalize the matching session definition from the eCK model [40] which is based on matching conversations.

Two completed sessions s and s′ are said to be matching if their respective actors have distinct roles and if, in addition: $\mathsf{s}_{\mathsf{Actor}} = \mathsf{s}'_{\mathsf{Peer}} \wedge \mathsf{s}_{\mathsf{Peer}} = \mathsf{s}'_{\mathsf{Actor}} \wedge \mathsf{s}_{\mathsf{Sent}} = \mathsf{s}'_{\mathsf{Rec}} \wedge \mathsf{s}_{\mathsf{Rec}} = \mathsf{s}'_{\mathsf{Sent}}$.

To relate a message received (and accepted) by some session to the session it originates from (if the later exists), [17] introduced the concept of origin sessions, a possibly not complete session s′ is an origin session for a completed session s when $\mathsf{s}'_{\mathsf{Sent}} = \mathsf{s}_{\mathsf{Rec}}$. If an origin session s′ for some session s exists, then the messages received by session s have not been modified or injected (as in Krawczyk's PFS attack [38]) by the adversary.

C.2 Correctness, Key Secrecy and Freshness predicates

Definition 3 (Correctness of the protocol). *Let $\delta > 0$. The protocol is considered δ-correct if the probability that two sessions executing the protocol together—without any modification of the messages by an adversary—either do not complete or fail to agree on the same session key is upper bounded by δ.*

In order to measure the Key Secrecy of the protocol, we will consider a real-or-random game where the adversary tries to distinguish whether the session key given by the query test-session is $\mathsf{s}^{\mathsf{Test}}_{\mathsf{SessionKey}}$ or has been uniformly sampled.

Note that an adversary capable of performing the above queries can effortlessly reveal the session key of all sessions with session-key, including $\mathsf{s}^{\mathsf{Test}}_{\mathsf{SessionKey}}$, thus easily winning this real-or-random game.

To obtain a sound notion of security, the powers of the adversary need to be limited. These limitations are defined by a predicate F, called freshness predicate, that takes as input the transcript of queries made by the adversary[30] and returns true or false. What defines a variant of eCK is the choice of freshness predicates. When, in an instance of the game, the limitations are not respected, we say that the instance of the game is invalid.

Note that for F and F' two freshness predicates, we can naturally define freshness predicates $F \wedge F'$, $F \vee F'$ and $\neg F$ (also denoted $\overline{F}$).

Let $\Pi = (\pi, \mathsf{Keygen}_{\mathsf{static}})$ a protocol, F a freshness predicate and $\mathcal{A}$ an adversary that can make queries to send, corrupt, eph-key, session-key, test-session and returns a bit $b \in \{0,1\}$. The key secrecy game F-KeySec for this protocol and adversary is defined in Figure 15. Its advantage is defined by:

$$\begin{aligned}\mathsf{Adv}^{F\text{-}\mathsf{KeySec}}_{\Pi}(\mathcal{A}) &= \left| \Pr[F\text{-}\mathsf{KeySec}(\mathcal{A}) = 1] - \frac{1}{2} \right| \\ &= \frac{1}{2} \Big| \Pr[1 \leftarrow \mathcal{A} \wedge F(\mathsf{tr}) \mid b_{\mathsf{qtest}} \text{ defined and } b_{\mathsf{qtest}} = 0] \\ &\qquad - \Pr[1 \leftarrow \mathcal{A} \wedge F(\mathsf{tr}) \mid b_{\mathsf{qtest}} \text{ defined and } b_{\mathsf{qtest}} = 1] \Big|.\end{aligned}$$

[30] We include the arguments and outputs of the queries

F-KeySec($\mathcal{A}$)

$\mathcal{O} = \{\mathsf{activate\text{-}party}, \mathsf{activate\text{-}session}, \mathsf{send}, \mathsf{corrupt}, \mathsf{eph\text{-}key}, \mathsf{session\text{-}key}, \mathsf{test\text{-}session}\}$
$\mathsf{E}_{\mathsf{ValidInstance}} \leftarrow \mathbf{true}$
$b \leftarrow \mathcal{A}^{\mathcal{O}}()$
$\mathsf{tr} \leftarrow$ transcript of requests made by $\mathcal{A}$
if $\mathsf{s}^{\mathsf{Test}} = \perp$ **then** $\mathsf{E}_{\mathsf{ValidInstance}} \leftarrow \mathbf{false}$
  // $\mathsf{s}^{\mathsf{Test}}$ is the test session, set by the query test-session,
  // $\mathsf{s}^{\mathsf{Test}} = \perp$ happens when test-session has not be used to a completed session
if $F(\mathsf{tr}) = \mathbf{false}$ **then** $\mathsf{E}_{\mathsf{ValidInstance}} \leftarrow \mathbf{false}$
if $\mathsf{E}_{\mathsf{ValidInstance}} = \mathbf{false}$ **then**
  $u \leftarrow\$ \{0,1\}$
  return u
return $b = b_{\mathsf{qtest}}$
  // b_{qtest} is the bit sampled by the query to test-session

Fig. 15: Security game F-KeySec($\mathcal{A}$).

where b_{qtest} is defined if and only $\mathsf{s}^{\mathsf{Test}} \neq \perp$.

A game instance where $\mathsf{E}_{\mathsf{ValidInstance}}$ is true will be referred to as the valid instance, while all other instances will be termed invalid instances. A valid instance is thus an instance such that the adversary used test-session on a completed session and that $F(\mathsf{tr})$ is true. Note that for an invalid instance, a random bit is returned instead of the adversary's result.

C.3 Definition of eCK^{w} and eCK-PFS Models

Simple Freshness Predicates. We define multiple freshness predicates that will be used to define the freshness predicates of eCK^{w}, eCK-PFS and their unilateral adaptations u-eCK^{w}, u-eCK-PFS.

Before that, we introduce the following notations.

For a session s, $\mathsf{s}_{\mathsf{ActorId}}$ (*resp.* $\mathsf{s}_{\mathsf{PeerId}}$) denotes the identity of the actor (*resp.* peer) of the session s and $\mathsf{s}_{\mathsf{ActorRole}}$ (*resp.* $\mathsf{s}_{\mathsf{PeerRole}}$) denotes its role. For a session s, $\mathsf{s}_{\mathsf{Init}}$ (*resp.* $\mathsf{s}_{\mathsf{Resp}}$) denotes $\mathsf{s}_{\mathsf{Actor}}$ if $\mathsf{s}_{\mathsf{ActorRole}} = \mathsf{Init}$ (*resp.* $\mathsf{s}_{\mathsf{ActorRole}} = \mathsf{Resp}$) and $\mathsf{s}_{\mathsf{Peer}}$ otherwise.

For $\mathsf{X} \in \mathsf{A}, \mathsf{P}, \mathsf{I}, \mathsf{R}$, we define $\mathsf{party}(\mathsf{X})$ as $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$, $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Init}}$, and $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Resp}}$, respectively—where $\mathsf{s}^{\mathsf{Test}}$ denotes the test session. For $\mathsf{Y} \in \mathsf{Match}, \mathsf{Orig}$, we let $\mathsf{Set}(\mathsf{Y})$ denote the set of sessions that are, respectively, matching or origin sessions of $\mathsf{s}^{\mathsf{Test}}$. Using these notations, we define the following freshness predicates:

- Y is true if, and only if, $\mathsf{Set}(\mathsf{Y})$ is non-empty.
- CorX is true if, and only if, $\mathsf{party}(\mathsf{X})$ is corrupted.
- $\mathsf{EmptySsk}(\mathsf{X})$ is true if, and only if, the static secret-key of $\mathsf{party}(\mathsf{X})$ is empty.
- $\mathsf{AisInit}$ is true if, and only if, the role of $\mathsf{party}(A)$ is Init.
- $\mathsf{AisResp}$ is true if, and only if, the role of $\mathsf{party}(A)$ is Resp.
- SkT is true if, and only if, the query $\mathsf{session\text{-}key}(\mathsf{s}^{\mathsf{Test}})$ is asked by the adversary.
- SkM is true if, and only if, there exists a session s matching the test session such that $\mathsf{session\text{-}key}(\mathsf{s})$ is asked by the adversary.
- $\mathsf{LeakRandT}$ is true if, and only if, the query $\mathsf{eph\text{-}key}\left(\mathsf{s}^{\mathsf{Test}}\right)$ is asked by the adversary.
- $\mathsf{LeakRandO}$ is true if, and only if, the query $\mathsf{eph\text{-}key}(\mathsf{s})$ is asked by the adversary to an origin session s of $\mathsf{s}^{\mathsf{Test}}$.
- $\mathsf{corXbeforeT}$ (*resp.* $\mathsf{corXafterT}$) is true if, and only if, $\mathsf{party}(\mathsf{X})$ is corrupted before (*resp.* corrupted after) $\mathsf{s}^{\mathsf{Test}}$ is complete.

Definitions of eCK^{w} and eCK-PFS. Using the freshness predicate defined in the previous section, we set:

$$\mathsf{CorLeak} = \left(\overline{\mathsf{CorP}} \vee \overline{\mathsf{LeakRandO}}\right) \wedge \left(\overline{\mathsf{CorA}} \vee \overline{\mathsf{LeakRandT}}\right) ,$$

and

$$\mathsf{eCK}^{\mathsf{w}} = \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} \wedge \mathsf{CorLeak} \wedge \left(\overline{\mathsf{Orig}} \Rightarrow \overline{\mathsf{CorP}}\right) ,$$
$$\mathsf{eCK\text{-}PFS} = \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} \wedge \mathsf{CorLeak} \wedge \left(\overline{\mathsf{Orig}} \Rightarrow \overline{\mathsf{corPbeforeT}}\right) .$$

These definitions correspond to the attack scenarios described in Section 3, Table 4. Indeed, if we define the five following predicates:

$$\begin{aligned}
\mathsf{F}_1 &= \overline{\mathsf{CorA}} \wedge \overline{\mathsf{CorP}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,,\\
\mathsf{F}_2 &= \mathsf{CorA} \wedge \overline{\mathsf{CorP}} \wedge \overline{\mathsf{LeakRandT}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,,\\
\mathsf{F}_3 &= \overline{\mathsf{CorA}} \wedge \mathsf{CorP} \wedge \mathsf{Orig} \wedge \overline{\mathsf{LeakRandO}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,,\\
\mathsf{F}_4 &= \mathsf{CorA} \wedge \mathsf{CorP} \wedge \overline{\mathsf{LeakRandT}} \wedge \mathsf{Orig} \wedge \overline{\mathsf{LeakRandO}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,,\\
\mathsf{F}_5 &= \mathsf{CorA} \wedge \mathsf{corPafterT} \wedge \overline{\mathsf{LeakRandT}} \wedge \overline{\mathsf{Orig}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,,\\
\mathsf{F}_6 &= \overline{\mathsf{CorA}} \wedge \mathsf{corPafterT} \wedge \overline{\mathsf{Orig}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,.
\end{aligned} \tag{14}$$

We can see that they are mutually disjoint and that:

$$\mathsf{eCK}^{\mathsf{w}} = \bigvee_{i=1}^{4} \mathsf{F}_i\,, \qquad \mathsf{eCK\text{-}PFS} = \bigvee_{i=1}^{6} \mathsf{F}_i\,, \tag{15}$$

which implies:

$$\mathsf{Adv}_{\Pi}^{\mathsf{eCK^w\text{-}KeySec}} \leq \sum_{i=1^4} \mathsf{Adv}_{\Pi}^{\mathsf{F}_i\text{-}\mathsf{KeySec}}\,, \qquad \mathsf{Adv}_{\Pi}^{\mathsf{eCK\text{-}PFS\text{-}KeySec}} \leq \sum_{i=1^6} \mathsf{Adv}_{\Pi}^{\mathsf{F}_i\text{-}\mathsf{KeySec}}\,.$$

C.4 Unilateral Versions of eCK^{w} (u-eCK^{w}) and eCK-PFS (u-eCK-PFS)

A protocol is called *unilateral* when the parties of a given role have no static secrets. More formally, for $\mathsf{Role} \in \{\mathsf{Init}, \mathsf{Resp}\}$, we say that a protocol is Role-unilateral if, by denoting Role^* the other role and $\epsilon \in \{0,1\}^*$ the empty string, we have,

$$\begin{aligned}
&\Pr[\mathsf{ssk} \neq \epsilon \mid \mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{static}}(), (\mathsf{spk}, \mathsf{ssk}) \leftarrow \mathsf{Keygen}_{\mathsf{static}}(\mathsf{pp}, \mathsf{Role})] = 1,\\
&\Pr[\mathsf{ssk} = \epsilon \mid \mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{static}}(), (\mathsf{spk}, \mathsf{ssk}) \leftarrow \mathsf{Keygen}_{\mathsf{static}}(\mathsf{pp}, \mathsf{Role}^*)] = 1.
\end{aligned}$$

A protocol is unilateral if there is a role Role such that it is Role-unilateral.

The models $\mathsf{eCK}^{\mathsf{w}}, \mathsf{eCK\text{-}PFS}$ we defined are not suited for unilateral protocols. Indeed, in a Role-unilateral protocol, the adversary can impersonate parties with role Role^* as they do not have secret static keys. Therefore, it is necessary to impose additional restrictions on the $\mathsf{eCK}^{\mathsf{w}}$ and eCK-PFS models to eliminate unavoidable attacks arising from their unilateral nature. This modification results in the development of the refined models u-$\mathsf{eCK}^{\mathsf{w}}$ and u-eCK-PFS.

More precisely, we define variants of $\mathsf{eCK}^{\mathsf{w}}$ and eCK-PFS as u-$\mathsf{eCK}^{\mathsf{w}} = \mathsf{eCK}^{\mathsf{w}} \wedge \mathsf{F}_{\mathsf{unilateral}}$ and u-eCK-PFS = eCK-PFS $\wedge\, \mathsf{F}_{\mathsf{unilateral}}$, where the predicate $\mathsf{F}_{\mathsf{unilateral}}$ is given by:

$$\big(\mathsf{EmptySsk(P)} \Rightarrow (\mathsf{PartialOrig} \wedge \overline{\mathsf{LeakRandO}})\big) \wedge (\mathsf{EmptySsk(A)} \Rightarrow \overline{\mathsf{LeakRandT}}).$$

Here, PartialOrig is true if, and only if, there exists a partial origin session of $\mathsf{s}^{\mathsf{Test}}$. The notion of partial origin is a variant of the notion or origin session: a possibly not complete session s' is a partial origin session for a possibly not complete session s if one of the messages received by s is equal to one of the messages sent by s'.

We note that $\mathsf{F}_{\mathsf{unilateral}}$ is false if, and only if, at least one of the unavoidable attacks can occur. Specifically:

- If EmptySsk(P) is true but PartialOrig is false, there is nothing to prevent the adversary from impersonating a session where P is the actor, allowing the adversary to conduct the protocol with $\mathsf{s}^{\mathsf{Test}}$.
- If EmptySsk(P) and PartialOrig are true but $\overline{\mathsf{LeakRandO}}$ is false, the adversary knows the randomness of the first origin session s' of $\mathsf{s}^{\mathsf{Test}}$. This randomness is the only secret protecting s' because its actor, the peer of $\mathsf{s}^{\mathsf{Test}}$, does not have a static secret.
- If EmptySsk(A) is true but $\overline{\mathsf{LeakRandT}}$ is false, the adversary knows the randomness of the test session $\mathsf{s}^{\mathsf{Test}}$, which is the sole secret protecting this session since its actor does not have a static secret.

u-eCK^{w} and u-eCK-PFS as the Disjunction of Simpler Cases. The following proposition shows how to simplify the analysis of Init-unilateral protocols for which the responder sends only one message per session by dividing the analysis in smaller cases.

Proposition 1. *Fix any* Init*-unilateral protocol such that any responder sends only one message per session. We consider the following predicates:*

$$\begin{aligned}
\mathsf{F}_1^U &= \mathsf{AisInit} \wedge \overline{\mathsf{CorI}} \wedge \overline{\mathsf{CorR}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} \wedge \overline{\mathsf{LeakRandO}} \wedge \mathsf{Orig},\\
\mathsf{F}_2^U &= \mathsf{AisResp} \wedge \overline{\mathsf{CorI}} \wedge \overline{\mathsf{CorR}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} \wedge \overline{\mathsf{LeakRandT}},\\
\mathsf{F}_3^U &= \mathsf{AisInit} \wedge \mathsf{CorI} \wedge \overline{\mathsf{CorR}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} \wedge \overline{\mathsf{LeakRandT}} \wedge \overline{\mathsf{LeakRandO}} \wedge \mathsf{Orig},\\
\mathsf{F}_4^U &= \mathsf{AisResp} \wedge \mathsf{CorI} \wedge \overline{\mathsf{CorR}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} \wedge \overline{\mathsf{LeakRandT}} \wedge \overline{\mathsf{LeakRandO}} \wedge \mathsf{Orig},\\
\mathsf{F}_5^U &= \mathsf{AisResp} \wedge \mathsf{corIafterT} \wedge \overline{\mathsf{CorR}} \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} \wedge \overline{\mathsf{LeakRandT}} \wedge \overline{\mathsf{Orig}}.
\end{aligned}$$

They are disjoint and, for $X \in \{\mathsf{u\text{-}eCK^{w}}, \mathsf{u\text{-}eCK\text{-}PFS}\}$ *and* $\mathcal{A}$ *an adversary against the* X-KeySec *security, there exists an adversary* $\mathcal{B}$ *running with same time as* $\mathcal{A}$ *and using at most the same number of requests to each of the available oracles (*activate-party, activate-session, eph-key, corrupt, test-session *and random oracle) such that:*

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{u\text{-}eCK^{w}\text{-}KeySec}}(\mathcal{A}) &\leq \sum_{i=1}^{4} \mathsf{Adv}^{\mathsf{F}_i^U}(\mathcal{A}),\\
\mathsf{Adv}^{\mathsf{u\text{-}eCK\text{-}PFS\text{-}KeySec}}(\mathcal{A}) &\leq \sum_{i=1}^{5} \mathsf{Adv}^{\mathsf{F}_i^U}(\mathcal{A})\,.
\end{aligned}$$

Proof. The disjonction of these fives predicates can be shown by direct comparison. We show the other part of the proposition.

Let $\mathcal{A}$ be an adversary and let $X \in \{\mathsf{u\text{-}eCK^{w}}, \mathsf{u\text{-}eCK\text{-}PFS}\}$. Since the protocol is Init-unilateral, each use of corrupt by $\mathcal{A}$ on a responder does not give more information to $\mathcal{A}$. We construct an adversary $\widetilde{\mathcal{A}}$ that acts as $\mathcal{A}$ except for the following:

- each time $\mathcal{A}$ uses corrupt on a responder, $\widetilde{\mathcal{A}}$ does not use corrupt[31].
- $\widetilde{\mathcal{A}}$ memorizes all the necessary data to be able to tell at any time if the instance of the game is valid or not.
- when $\mathcal{A}$ outputs a bit, $\widetilde{\mathcal{A}}$ checks if the instance of the game is valid or invalid:
  - if valid, $\widetilde{\mathcal{A}}$ outputs the same bit that $\mathcal{A}$.
  - if invalid, $\widetilde{\mathcal{A}}$ samples and outputs a uniform bit.

We can see that $\widetilde{\mathcal{A}}$ never uses corrupt on a responder, uses the same number of oracle queries for any other cases and that $\mathsf{Adv}^X(\mathcal{A}) = \mathsf{Adv}^X(\widetilde{\mathcal{A}})$.

Second step: We now suppose that $\mathcal{A}$ never uses corrupt on a responder. We can see that for this adversary, an instance of the game X is valid if, and only if, it is $\widetilde{X}$-valid, where $\widetilde{X} = X \wedge \overline{\mathsf{CorR}}$.

Finally, a direct computation shows that $\mathsf{F}_1^U \vee \mathsf{F}_2^U \vee \mathsf{F}_3^U \vee \mathsf{F}_4^U = \widetilde{\mathsf{u\text{-}eCK^{w}}}$ and $\mathsf{F}_1^U \vee \mathsf{F}_2^U \vee \mathsf{F}_3^U \vee \mathsf{F}_4^U \vee \mathsf{F}_5^U = \widetilde{\mathsf{u\text{-}eCK\text{-}PFS}}$.

D Detailed Presentation of DAKE$^{\sigma}$ and DUAKE

In this section, we provide additional details on DAKE$^{\sigma}$ and DUAKE, including their definitions, the elements revealed by eph-key, and their security properties.

D.1 DAKE$^{\sigma}$: A Variant of DAKE Using a Signature

DAKE$^{\sigma}$ is a variant of DAKE that uses a signature, instead of a KEM and a MAC, to authenticate the responder. It is specified in Figure 16. It comes in two versions: a two-message variant, DAKE$_2^{\sigma}$, which provides weak Perfect Forward Secrecy (wPFS), and a three-message variant, DAKE$_3^{\sigma}$, which provides full PFS.

[31] Since the output of corrupt is ϵ, $\widetilde{\mathcal{A}}$ does not lose any information and can therefore continue to behave as $\mathcal{A}$.

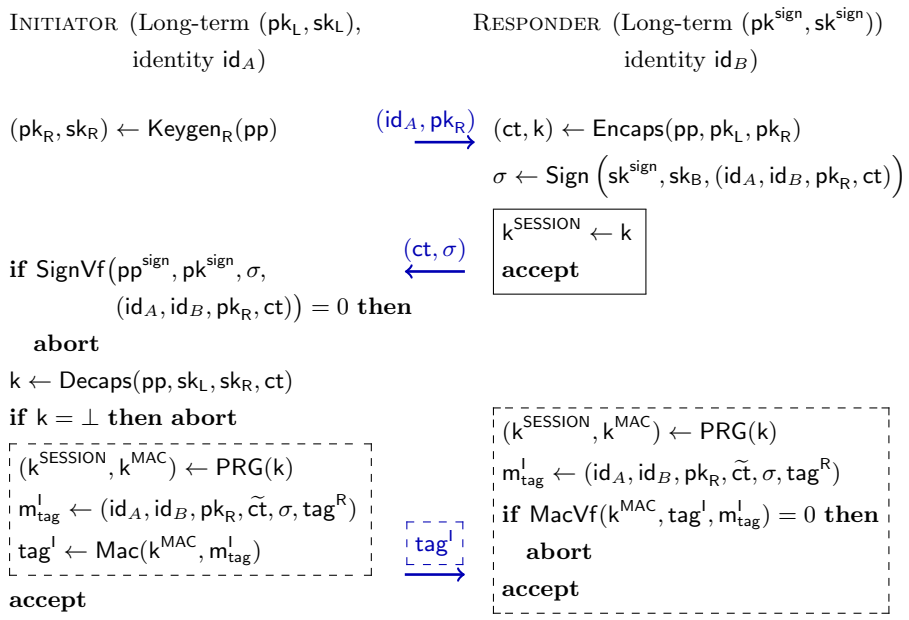

Fig. 16: The DAKE^{σ} protocol. Public parameters are (1) $(\mathsf{pp}, \mathsf{pp}^{\mathsf{sign}})$ with $\mathsf{pp} \leftarrow \mathsf{Setup}()$ and $\mathsf{pp}^{\mathsf{sign}} \leftarrow \mathsf{Setup}^{\mathsf{sign}}()$, (2) all party identities, and (3) a dictionary mapping identities to public keys. The initiator chooses its peer before sending the first message. DAKE^{σ}_2 (*resp.* DAKE^{σ}_3) is obtained by removing the dotted (*resp.* solid) boxes.

Leakage through eph-key. Concerning the elements leaked by eph-key:
- Applying eph-key to an initiator's session reveals the randomness used in $\mathsf{Keygen_R}$, which implies knowledge of $\mathsf{sk_R}$.
- For the responder, nothing is revealed: we assume that the randomness used in both the encapsulation and the signature is securely erased immediately after their computation.

Security properties. Key secrecy is given by the Theorem 7 of Appendix E. Match soundness and inverse-match soundness are studied in Section 3.3. Finally, explicit authentication is studied in Appendix F.

D.2 DUAKE: An Unilateral AKE Fitted for Maul

DUAKE is a variant of DAKE fitted for the unilateral setting, where only the initiator is authenticated. It is given in Figure 2 of Section 1. It has a two-message variant which provides weak Perfect Forward Secrecy (weak PFS) named DUAKE_2 and a three-message variant which provides PFS named DUAKE_3.

Leakage through eph-key. Concerning the elements leaked by eph-key:
- Applying eph-key to an initiator's session reveals the randomness used in $\mathsf{Keygen_R}$, which implies knowledge of $\mathsf{sk_R}$.
- For the responder, nothing is revealed: we assume that the randomness of the encapsulation is securely erased immediately after its computation.

Security properties. Key secrecy is given by the Theorem 8 in Appendix E. It is studied under the u-eCK^{w} model for the weak PFS and u-eCK-PFS for the PFS. These two models are respectively adaptations of the eCK^{w} and eCK-PFS models to the unilateral setting, their definitions being given and justified in Section C.4.

Match soundness and inverse-match soundness are studied in Section 3.3. Finally, explicit authentication is studied in Appendix F.

E Key Secrecy of DAKE, DAKE$^\sigma$, and DUAKE

We aim to prove the key secrecy of three protocols, each of which has both a two-message and a three-message variant:

- DAKE, with variants DAKE_2 and DAKE_3, presented in Figure 1 of Section 3. The key secrecy of DAKE_3 is given in Theorem 1, Section 3.3 and the key secrecy of DAKE_2 is given in Theorem 6 below.
- DAKE$^\sigma$, with variants DAKE_2^σ and DAKE_3^σ, presented in Figure 16 of Section D.1. Its key secrecy is given in Theorem 7 below.
- DUAKE, with variants DUAKE_2 and DUAKE_3, presented in Figure 2 of Section D.2. Its key secrecy is given in Theorem 8 below.

The remainder of this section is organized as follows: Appendices E.1 and E.2 present the proof blueprint and required notations, while Appendices E.3, E.4, and E.5 contain the full security proof.

Theorem 6 (Key Secrecy of DAKE_2 in the Standard Model). *Let $\mathcal{A}$ be a PPT adversary against key secrecy of DAKE_2. Suppose it activates at most N parties and, for each party P, at most l sessions with actor or peer P. Assume that the underlying 2K-KEM is γ-spread and δ-correct while the standard KEM is δ'-correct. Let $\mathsf{Adv}^{\mathsf{w}}$ denote the advantage of $\mathcal{A}$ against $\mathsf{eCK}^{\mathsf{w}}$-KeySec. Then, in the standard model, there exist PPT adversaries $\mathcal{B}_1, \cdots, \mathcal{B}_7$ such that:*

$$\mathsf{Adv}^{\mathsf{w}} \leq \alpha + 4\beta + 6Nl^2 \left(\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\cdot]}_{\mathsf{dKEM}}(\mathcal{B}_6) + \mathsf{Adv}^{[\cdot,\mathsf{IND\text{-}CCA}]}_{\mathsf{dKEM}}(\mathcal{B}_7) \right),$$

where $\alpha = \mathsf{Adv}^{\mathsf{MacColl}}_{\mathsf{MAC}}(\mathcal{B}_1) + N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\widetilde{\mathsf{KEM}}}(\mathcal{B}_2) + N^2 l\Big(2\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\widetilde{\mathsf{KEM}}}(\mathcal{B}_3) + l\delta' + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{B}_4)\Big)$ and $\beta = Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}_5) + 2Nl^2\,(\delta + 2^{-\gamma})$.

Moreover, (1) $\mathcal{B}_3$ queries $\mathcal{O}_{\mathsf{CCA}}$ at most l times, (2) $\mathcal{B}_4$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most l time. (3) $\mathcal{B}_6$ query $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$ at most l times, (4) $\mathcal{B}_7$ does not query $\mathcal{O}_{\mathsf{leak_L}}$ and queries $\mathcal{O}_{\mathsf{CCA_R}}$ at most one time.

Theorem 7 (Key Secrecy of DAKE^σ in the Standard Model). *Let $\mathcal{A}$ be a PPT adversary against key secrecy of DAKE_3^σ. Suppose it activates at most N parties and, for each party P, at most l sessions with actor or peer P. Assume that the underlying 2K-KEM is γ-spread and δ-correct. Let $\mathsf{Adv}^{\mathsf{w}}$ denote the advantage of $\mathcal{A}$ against $\mathsf{eCK}^{\mathsf{w}}$-KeySec and $\mathsf{Adv}^{\mathsf{PFS}}$ its advantage against eCK-PFS-KeySec. Then, in the standard model, there exist PPT adversaries $\mathcal{B}_1, \cdots, \mathcal{B}_{11}$ such that, denoting*

$$\alpha = \mathsf{Adv}^{\mathsf{MBS}}_{\mathsf{Sign}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{M\text{-}S\text{-}UEO}}_{\mathsf{Sign}}(\mathcal{B}_2) + N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\mathsf{Sign}}(\mathcal{B}_3) + N^2 l\mathsf{Adv}^{\mathsf{sUF\text{-}CMA}}_{\mathsf{Sign}}(\mathcal{B}_4)\,,$$

and $\beta = Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}_5) + 2Nl^2\,(\delta + 2^{-\gamma})$, we have:

$$\mathsf{Adv}^{\mathsf{w}} \leq \alpha + 4\beta + 6Nl^2 \Big(\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\cdot]}_{\mathsf{dKEM}}(\mathcal{B}_6) + \mathsf{Adv}^{[\cdot,\mathsf{IND\text{-}CCA}]}_{\mathsf{dKEM}}(\mathcal{B}_7) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{B}_8) \Big),$$

$$\mathsf{Adv}^{\mathsf{PFS}} \leq \mathsf{Adv}^{\mathsf{w}} + \beta + 2Nl^2 \Big(2\,\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\cdot]}_{\mathsf{dKEM}}(\mathcal{B}_9) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{B}_{10}) + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{B}_{11}) \Big)\,.$$

Moreover, (1) $\mathcal{B}_4$ queries $\mathcal{O}_{\mathsf{Sign}}$ at most l times, (2) $\mathcal{B}_6, \mathcal{B}_9$ query $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$ at most l times, (3) $\mathcal{B}_7$ does not query $\mathcal{O}_{\mathsf{leak_L}}$ and queries $\mathcal{O}_{\mathsf{CCA_R}}$ at most one time, and (4) $\mathcal{B}_{11}$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most one time.

We now consider a PPT adversary $\mathcal{A}$ against the key secrecy of DAKE_2^σ. Under the same assumptions as above, and letting $\mathsf{Adv}_2^{\mathsf{w}}$ denote its advantage against $\mathsf{eCK}^{\mathsf{w}}$-KeySec, $\mathsf{Adv}_2^{\mathsf{w}}$ satisfies the same upper bound as $\mathsf{Adv}^{\mathsf{w}}$, except that the term $2\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{B}_8)$ is omitted.

Theorem 8 (Key Secrecy of DUAKE in Standard Model). *Let $\mathcal{A}$ be a PPT adversary against key secrecy of DUAKE_3. Suppose it activates at most N parties and, for each party P, at most l sessions with actor or peer P. Assume that the underlying 2K-KEM is γ-spread and δ-correct. Let $\mathsf{Adv}^{\mathsf{w}}$ denote the advantage of $\mathcal{A}$ against $\mathsf{eCK}^{\mathsf{w}}$-KeySec and $\mathsf{Adv}^{\mathsf{PFS}}$ its advantage against eCK-PFS-KeySec. Then, in the standard model, there exist PPT adversaries $\mathcal{B}_1, \cdots, \mathcal{B}_8$ such that, denoting $\beta = Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}_1) + 2Nl^2\,(\delta + 2^{-\gamma})$, we have:*

$$\mathsf{Adv}^{\mathsf{w}} \leq \mathsf{Adv}^{\mathsf{HashColl}}_{\mathsf{H}}(\mathcal{B}_2) + 4\beta + 6Nl^2 \Big(\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\cdot]}_{\mathsf{dKEM}}(\mathcal{B}_3) + \mathsf{Adv}^{[\cdot,\mathsf{IND\text{-}CCA}]}_{\mathsf{dKEM}}(\mathcal{B}_4) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{B}_5) \Big),$$

$$\mathsf{Adv}^{\mathsf{PFS}} \leq \mathsf{Adv}^{\mathsf{w}} + \beta + 2Nl^2 \Big(2\,\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\cdot]}_{\mathsf{dKEM}}(\mathcal{B}_6) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{B}_7) + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{B}_8) \Big)\,.$$

Moreover, (2) $\mathcal{B}_3, \mathcal{B}_6$ query $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$ at most l times, (3) $\mathcal{B}_4$ does not query $\mathcal{O}_{\mathsf{leak_L}}$ and queries $\mathcal{O}_{\mathsf{CCA_R}}$ at most one time, and (4) $\mathcal{B}_8$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most one time.

We now consider a PPT adversary $\mathcal{A}$ against the key secrecy of DUAKE_2. Under the same assumptions as above, and letting $\mathsf{Adv}_2^{\mathsf{w}}$ denote its advantage against $\mathsf{eCK}^{\mathsf{w}}$-KeySec, $\mathsf{Adv}_2^{\mathsf{w}}$ satisfies the same upper bound as $\mathsf{Adv}^{\mathsf{w}}$, except that the term $2\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{B}_5)$ is omitted.

E.1 Context, Rationale of the Proof and Notations

In this section we outline the key secrecy proof for DAKE and DAKE$^{\sigma}$. The analysis of DUAKE, which reuses this approach, is given separately in Appendix E.5.

Note that DAKE and DAKE$^{\sigma}$ differ only in how the responder is authenticated:

- In DAKE, identification is achieved through a KEM and a MAC.
- In DAKE$^{\sigma}$, identification is achieved through a digital signature.

If we remove the responder's identification step altogether, we obtain the common underlying AKE depicted in Figure 17.

Our goal in this proof is to exploit the structural similarity between DAKE and DAKE$^{\sigma}$. To this end, we will not analyze them directly under the standard attack scenarios of eCK^{w} and eCK-PFS, expressed by the predicates $\mathsf{F}_1, \ldots, \mathsf{F}_5$ in Equation 14. Instead, we introduce a refined set of predicates, $\mathsf{U}, \mathsf{V}_1, \ldots, \mathsf{V}_5$[32] better suited to leverage the resemblance of the two protocols. Among these, only U requires us to analyze the responder's identification step, which differs between DAKE and DAKE$^{\sigma}$. For all other cases, the proof does not depend on the responder's identification and is therefore identical for both protocols. In fact, the proofs for $\mathsf{V}_1, \ldots, \mathsf{V}_5$ can be carried out using the simplified protocol of Figure 17.

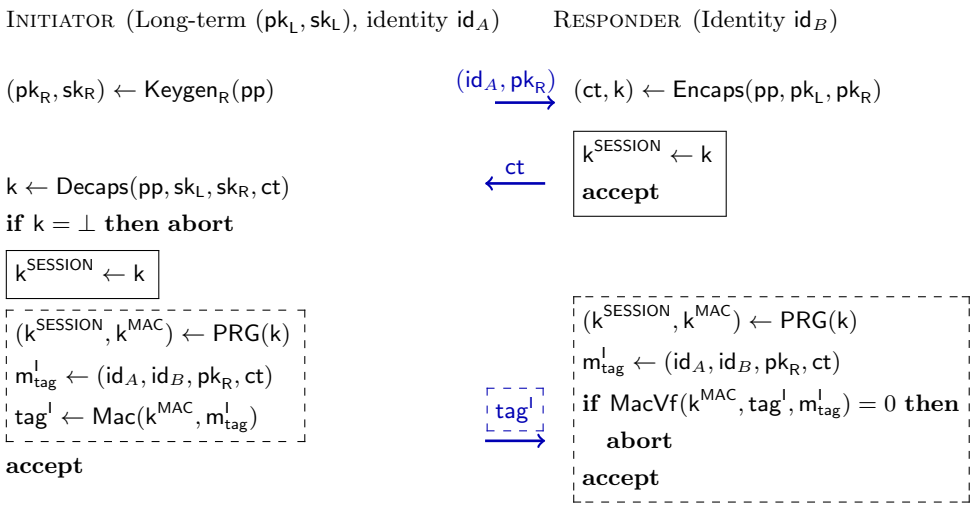

Fig. 17: A simplified AKE whose security can be proven under predicates $\mathsf{V}_1, \ldots, \mathsf{V}_5$. It is obtained from DAKE or DAKE$^{\sigma}$ by removing the components responsible for responder authentication. Instructions within dotted boxes must be removed to obtain the two-message version AKE_2.

In the remaining of this section, we formally define these new predicates. The detailed analysis is then organized as follows:

- Section E.2 define some necessary notations,
- Section E.3 analyzes the case U,
- Section E.4 covers the cases $\mathsf{V}_1, \ldots, \mathsf{V}_5$.

[32] For accuracy, we do not consider U alone, but rather $\mathsf{U} \wedge \mathsf{eCK}^{w}$ for eCK^{w} security, and $\mathsf{U} \wedge \mathsf{eCK\text{-}PFS}$ for eCK-PFS security.

Definition of the predicates.

$$\begin{aligned}
\mathsf{U} &= \mathsf{AisInit} \wedge \left((\overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{FirstMatch}}) \vee (\mathsf{Orig} \wedge \overline{\mathsf{FirstMatch}})\right) ,\\
\mathsf{V}_1 &= \mathsf{AisInit} \wedge \mathsf{FirstMatch} \wedge \overline{\mathsf{CorA}} \wedge \left(\overline{\mathsf{CorP}} \vee \overline{\mathsf{LeakRandO}}\right) \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} ,\\
\mathsf{V}_2 &= \mathsf{AisResp} \wedge \overline{\mathsf{CorP}} \wedge \left(\overline{\mathsf{CorA}} \vee \overline{\mathsf{LeakRandT}}\right) \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} ,\\
\mathsf{V}_3 &= \mathsf{AisInit} \wedge \mathsf{FirstMatch} \wedge \mathsf{CorA} \wedge \overline{\mathsf{LeakRandT}} \wedge \left(\overline{\mathsf{CorP}} \vee \overline{\mathsf{LeakRandO}}\right)\\
&\quad \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} ,\\
\mathsf{V}_4 &= \mathsf{AisResp} \wedge \mathsf{CorP} \wedge \mathsf{Orig} \wedge \overline{\mathsf{LeakRandO}} \wedge \left(\overline{\mathsf{CorA}} \vee \overline{\mathsf{LeakRandT}}\right)\\
&\quad \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} ,\\
\mathsf{V}_5 &= \mathsf{AisResp} \wedge \mathsf{corPafterT} \wedge \overline{\mathsf{Orig}} \wedge \left(\overline{\mathsf{CorA}} \vee \overline{\mathsf{LeakRandT}}\right) \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}} .
\end{aligned} \tag{16}$$

Here, FirstMatch holds if and only if the test session has a first-matching session s, which is defined by the following conditions: (1) the first message send and received by $\mathsf{s}^{\mathsf{Test}}$ are equal to the first messages received and sent by s, and (2) they have distinct roles and (3) $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}} = \mathsf{s}_{\mathsf{Peer}} \wedge \mathsf{s}_{\mathsf{Actor}} = \mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$. Note that in the two-message protocols, if s and $\mathsf{s}^{\mathsf{Test}}$ are completed and first-matching, they are matching. However, this equivalence no longer holds in the three-message variants.

A direct computation shows that the predicates $\mathsf{U}, \mathsf{V}_1, \cdots, \mathsf{V}_5$ are all mutually disjoint and:

$$\mathsf{eCK}^{\mathsf{w}} = (\mathsf{U} \wedge \mathsf{eCK}^{\mathsf{w}}) \vee \left(\bigvee_{i=1}^{4} \mathsf{V}_i\right) \qquad \mathsf{eCK\text{-}PFS} = (\mathsf{U} \wedge \mathsf{eCK\text{-}PFS}) \vee \left(\bigvee_{i=1}^{5} \mathsf{V}_i\right) . \tag{17}$$

It is thus possible to study separately the security of the AKEs for each of these predicates. More precisely, for any of the AKEs AKE and any adversary $\mathcal{A}$, we have:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{eCK}^{\mathsf{w}}}_{\mathsf{AKE}}(\mathcal{A}) &\leq \mathsf{Adv}^{(\mathsf{U} \wedge \mathsf{eCK}^{\mathsf{w}})}_{\mathsf{AKE}}(\mathcal{A}) + \sum_{i=1}^{4} \mathsf{Adv}^{\mathsf{V}_i}_{\mathsf{AKE}}(\mathcal{A}) ,\\
\mathsf{Adv}^{\mathsf{eCK\text{-}PFS}}_{\mathsf{AKE}}(\mathcal{A}) &\leq \mathsf{Adv}^{(\mathsf{U} \wedge \mathsf{eCK\text{-}PFS})}_{\mathsf{AKE}}(\mathcal{A}) + \sum_{i=1}^{5} \mathsf{Adv}^{\mathsf{V}_i}_{\mathsf{AKE}}(\mathcal{A}) .
\end{aligned}$$

E.2 Notations Used for the Proofs

First-Origin Session, First-Partner Session and First-Match Session. A session s is said to be a *first-origin session* of $\mathsf{s}^{\mathsf{Test}}$ if the first message it created is identical to the first message received by $\mathsf{s}^{\mathsf{Test}}$. It is important to distinguish this from an *origin session* because, in the three-message protocol, a first-origin session might not necessarily be an origin session. Making this distinction will highlight how this proof can be readily adapted to the three-messages protocol.

Similarly, a session s is said to be a *first-partner* of $\mathsf{s}^{\mathsf{Test}}$ if $\mathsf{s}^{\mathsf{Test}}$ is a first-origin session of s and s is a first-origin session of $\mathsf{s}^{\mathsf{Test}}$.

Finally, a session s is said to be a *first-match* of $\mathsf{s}^{\mathsf{Test}}$ if (1) $\mathsf{s}^{\mathsf{Test}}$ is a first-partner of s, and (2) they have distinct roles and (3) $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}} = \mathsf{s}_{\mathsf{Peer}} \wedge \mathsf{s}_{\mathsf{Actor}} = \mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$.

Elements of a session. Note that for a session s, we will denote the elements computed or received using this convention: $\mathsf{s.k}^{\mathsf{SESSION}}$, $\mathsf{s.ct}$, $\mathsf{s.pk_R}$, $\mathsf{s.sk_R}$, *etc.* These values are set to $\perp$ if they have not yet been computed or received. We also recall that $\mathsf{s}_{\mathsf{ActorId}}$ (*resp.* $\mathsf{s}_{\mathsf{PeerId}}$) denotes the identity of the actor (*resp.* peer) of the session s.

E.3 Key Secrecy for Predicates U ∩ eCK^{w} and U ∩ eCK-PFS

Proposition 2 (($\mathsf{U} \wedge \mathsf{eCK}^{\mathsf{w}}$)-KeySec and ($\mathsf{U} \wedge \mathsf{eCK\text{-}PFS}$)-KeySec-Security of DAKE). *We consider an adversary $\mathcal{A}$ against the Key Secrecy of DAKE_2 or DAKE_3. We suppose that $\mathcal{A}$ activates less than N parties and that for each fixed party P, it activates less than l sessions that have either actor or peer P. We also consider $\delta' \geq 0$ such that $\widetilde{\mathsf{KEM}}$ is δ'-correct. Then, in the standard model, for $\mathsf{X} \in \{\mathsf{Adv}^{\mathsf{U} \wedge \mathsf{eCK}^{\mathsf{w}}\text{-}\mathsf{KeySec}}_{\mathsf{DAKE}_2}(\mathcal{A}), \mathsf{Adv}^{\mathsf{U} \wedge \mathsf{eCK}^{\mathsf{w}}\text{-}\mathsf{KeySec}}_{\mathsf{DAKE}_3}(\mathcal{A}), \mathsf{Adv}^{\mathsf{U} \wedge \mathsf{eCK\text{-}PFS}\text{-}\mathsf{KeySec}}_{\mathsf{DAKE}_3}(\mathcal{A})\}$ there exist adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}$ such that,*

$$\mathsf{X} \leq \mathsf{Adv}^{\mathsf{MacColl}}_{\mathsf{MAC}}(\mathcal{B}) + N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\widetilde{\mathsf{KEM}}}(\mathcal{C}) + N^2 l\left(2\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\widetilde{\mathsf{KEM}}}(\mathcal{D}) + l\delta' + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{E})\right) .$$

Moreover, $\mathcal{D}$ queries $\mathcal{O}_{\mathsf{CCA}}$ at most l times and $\mathcal{E}$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most l times.

Proof. We recall that, as defined in Equation (16) (page 48):

$$\mathsf{U} = \mathsf{AisInit} \wedge \left((\overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{FirstMatch}}) \vee (\mathsf{Orig} \wedge \overline{\mathsf{FirstMatch}})\right) .$$

We will study the $\mathsf{U} \wedge \mathsf{eCK^w}$-KeySec security of DAKE_2, the proof for the $\mathsf{U} \wedge \mathsf{eCK^w}$-KeySec and $\mathsf{U} \wedge \mathsf{eCK^w}$-KeySec security of DAKE_3 being similar.

Note that this demonstration can also be reused to establish Resp-explicit authentication for DAKE_2 and DAKE_3, as explained in the following sidebar.

> *Reusing this demonstration to establish* Resp*-explicit authentication.*
> First, we note that the advantage in Resp-explicit authentication for DAKE_2 and DAKE_3 is exactly the probability that an instance of the Key Secrecy game satisfies the predicate $\mathsf{AisInit} \wedge \overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{FirstMatch}}$ and that the test session completes[33]. With this formulation, the test session corresponds to the session that the adversary must select in the Resp-explicit authentication game.
>
> Using this formulation of the Resp-explicit authentication game, we observe that the proof of Proposition 2 establishes Resp-explicit authentication for DAKE_2. Indeed, this proof bounds the probability that an instance of the game satisfies the predicate $\mathsf{AisInit} \wedge \overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{FirstMatch}}$ and that the test session completes. Moreover, the proof for DAKE_3 uses exactly the same arguments.

Game G_1: Origin session implies first match session. We recall that G_0 is the original security game for the AKE and that certain instances of this game can be designated as invalid, which is indicated by the event $\mathsf{E}_{\mathsf{ValidInstance}}$ being false. More precisely, an instance is invalid either if $\mathsf{U} \wedge \mathsf{eCK^w}$ is false or if no session has been tested. For an invalid instance, a random bit is returned instead of the adversary's result.

In G_1, additional instances are considered invalid. Specifically, any instance is deemed invalid if $\mathsf{s}^{\mathsf{Test}}$ is an initiator session and if there exists an origin session $\mathsf{s}^{\mathsf{Origin}}$ of $\mathsf{s}^{\mathsf{Test}}$ that is not a first matching session of $\mathsf{s}^{\mathsf{Test}}$.

Let us suppose that it is true and that $\mathsf{s}^{\mathsf{Test}}$ completes. Then, this directly leads to an attack against the MacColl-security of MAC. Indeed, $\mathsf{s}^{\mathsf{Origin}}$ creates a MAC $\tau \leftarrow \mathsf{Mac}(\mathsf{s}^{\mathsf{Origin}}.\widetilde{\mathsf{k}}, \mathsf{m}_1)$ for $\mathsf{m}_1 = (\mathsf{s}^{\mathsf{Origin}}_{\mathsf{PeerId}}, \mathsf{s}^{\mathsf{Origin}}_{\mathsf{ActorId}}, \mathsf{s}^{\mathsf{Origin}}.\mathsf{pk}_{\mathsf{R}}, \mathsf{s}^{\mathsf{Origin}}.\widetilde{\mathsf{ct}}, \mathsf{s}^{\mathsf{Origin}}.\mathsf{ct})$. Then, $\mathsf{s}^{\mathsf{Test}}$ receives τ and verifies that $\mathsf{MacVf}(\mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{k}}, \tau, \mathsf{m}_2)$ for $\mathsf{m}_2 = (\mathsf{s}^{\mathsf{Test}}_{\mathsf{ActorId}}, \mathsf{s}^{\mathsf{Test}}_{\mathsf{PeerId}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{pk}_{\mathsf{R}}, \mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{ct}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{ct})$. Moreover, because $\mathsf{s}^{\mathsf{Origin}}$ is not a first match, we know that $\mathsf{m}_1 \neq \mathsf{m}_2$. The tuple $(\mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{k}}, \mathsf{s}^{\mathsf{Origin}}.\widetilde{\mathsf{k}}, \mathsf{m}_1, \mathsf{m}_2, \tau)$ is thus a solution of MacColl.[34]

We thus have an adversary $\mathcal{B}$ such that:

$$\mathsf{Adv}^{\mathsf{G}_0}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{MacColl}}_{\mathsf{MAC}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A}).$$

In the valid instances of the next games, any origin session of $\mathsf{s}^{\mathsf{Test}}$ will also be a first matching session of $\mathsf{s}^{\mathsf{Test}}$.

In particular, we know that for this game and the next ones, the valid instances verify the predicate $\mathsf{AisInit} \wedge \overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{Orig}}$.

Game G_2: Removal of collisions of $\widetilde{\mathsf{pk}}$. In G_2, additional instances are considered invalid. Specifically, any instance is deemed invalid if there exist two distinct parties with role Resp that have the same public static key.

This probability can be bounded by $N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\mathsf{KEM}}(\mathcal{B})$ for some adversary $\mathcal{C}$ against the key recovery game of $\widetilde{\mathsf{KEM}}$. Thus, we have:

$$\left|\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A})\right| \leq N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\widetilde{\mathsf{KEM}}}(\mathcal{C}) .$$

Game G_3: Guessing multiple elements.

We guess $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$,$\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$ $\mathsf{s}^{\mathsf{Test}}$, we thus have: $\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A}) \leq N^2 l \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})$.

Game G_4: New win condition.

We create G_4 as G_3, with a new condition for victory. An instance of G_4 is won by the adversary $\mathcal{A}$ if, and only if, $\mathsf{s}^{\mathsf{Test}}$ completes without aborting and that nothing makes the instance of the game invalid (as per the conditions of G_3) until $\mathsf{s}^{\mathsf{Test}}$ completes. In particular, this means that the first message received by $\mathsf{s}^{\mathsf{Test}}$ contains a valid tag.

We stop the game at the moment $\mathsf{s}^{\mathsf{Test}}$ completes as this is sufficient to check if it is won. Because we stop the game at the moment when $\mathsf{s}^{\mathsf{Test}}$ completes, this implies that test-session is not even requested for

[34] The complete definition of MacColl is given in Section A.

$\mathsf{s}^{\mathsf{Test}}$. In particular, we know that in a valid session of G_4 and the following games, corrupt is not asked to $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$. Thus, the valid instances verify the predicate $\mathsf{AisInit} \wedge \overline{\mathsf{CorP}} \wedge \overline{\mathsf{Orig}}$.

Note that any winning instance of G_3 needs to satisfy the win conditions of G_4. Thus, we have: $\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A})$.

Game G_5: Replacement of $\mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{k}}$ by a uniform key.

We create game G_5 where:

- $\mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{k}}$ is replaced by a uniformly sampled key $\widetilde{\mathsf{k}}^*$,
- any responder session s with actor $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$ that receives the ciphertext $\mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{ct}}$ uses $\widetilde{\mathsf{k}}^*$ as $\mathsf{s}.\widetilde{\mathsf{k}}$ instead of computing it with $\widetilde{\mathsf{Decaps}}$.

To measure the difference of advantage between G_4 and G_5, we construct an adversary $\mathcal{D}$ against the IND-CCA security of $\widetilde{\mathsf{KEM}}$ as follows:

- $\mathcal{D}$ receives $(\mathsf{pp}, \mathsf{pk}_\star, \mathsf{ct}_\star, \mathsf{k}_u)$ where:
  - if $u = 0$, $\mathsf{ct}_\star$ is an encapsulation of k_u.
  - if $u = 1$, k_u has been uniformly, independently sampled.
- $\mathcal{D}$ begins by simulating G_4 to the adversary $\mathcal{A}$, using $\widetilde{\mathsf{Keygen}}(\mathsf{pp})$ to create the long-term keys of the parties with role responder, except $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$, that will uses $\mathsf{pk}_\star$. Note that in a valid instance, $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$ is not corrupted, there is thus no need to know the secret key associated to $\mathsf{pk}_\star$.
- Instead of computing the cipher and key computed by the session $\mathsf{s}^{\mathsf{Test}}$ with $\widetilde{\mathsf{Encaps}}(\widetilde{\mathsf{pp}}, \mathsf{pk}_\star)$, $\mathcal{D}$ uses $\mathsf{ct}_\star$ and k_u.
- If, at any moment, the (simulated) instance of G_4 becomes invalid, $\mathcal{D}$ stops the simulation and returns a uniform bit. We will assume in next items that this does not occur.
- For each session s such that $\mathsf{s}_{\mathsf{Actor}} = \mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$:
  - If the first message received by s contains ciphertext $\mathsf{ct}_\star$, the decapsulation of $\mathsf{s}.\widetilde{\mathsf{ct}} = \mathsf{ct}_\star$ computed by s is set to k_u.
  - Else, the decapsulation is computed using $\mathcal{O}_{\mathsf{CCA}}$.

A direct analysis of $\mathcal{D}$ shows that:

- if $u = 0$, $\mathcal{D}$ almost perfectly simulates G_4 to $\mathcal{A}$. The only issue with the simulation is that the decapsulation of the ciphertext received by $\mathsf{s}^{\mathsf{Test}}$ always succeeds, while in the real game, its result could be different of the encapsulated key. This difference can be measured by the correctness of $\widetilde{\mathsf{KEM}}$, and we have: $|\Pr[\mathcal{D} \text{ outputs } 1 \mid u = 0] - \Pr[1 \leftarrow \mathsf{G}_4(\mathcal{A})]| \leq l\delta'$.
- if $u = 1$, $\mathcal{D}$ perfectly simulates G_5 to $\mathcal{A}$. Thus, $\Pr[\mathcal{D} \text{ outputs } 1 \mid u = 1] = \Pr[1 \leftarrow \mathsf{G}_5(\mathcal{A})]$.

Finally, we obtain:

$$\left|\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_5}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\widetilde{\mathsf{KEM}}} + l\delta' \,.$$

Moreover, $\mathcal{D}$ queries the oracle $\mathcal{O}_{\mathsf{CCA}}$ at most l times.

Analysis of G_5. We upper bound $\mathsf{Adv}^{\mathsf{G}_5}(\mathcal{A})$ by the advantage of an adversary $\mathcal{E}$ winning the game sUnforg of the MAC.

$\mathcal{E}$ has access to an oracle $\mathsf{Mac}(\mathsf{k}, \cdot)$ for a key it does not know. When queried on an entry m, this oracle provides a tag tag such that $\mathsf{MacVf}(\mathsf{k}, \mathsf{tag}, m) = 1$ and registers the couple (m, tag) in a database Q. In order to win, $\mathcal{E}$ needs to provide a couple of message and tag $(m, \mathsf{tag}) \notin Q$ such that $\mathsf{MacVf}(\mathsf{k}, \mathsf{tag}, m) = 1$.

To do so, $\mathcal{E}$ simulates G_5 to $\mathcal{A}$ in such a way that k corresponds to $\mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{k}}$ and to $\mathsf{s}.\widetilde{\mathsf{k}}$ for each session s with actor $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$ that received $\mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{ct}}$. More precisely, for each of these sessions s, $\mathcal{E}$ does not compute $\mathsf{s}.\widetilde{\mathsf{k}}$, and uses the oracle $\mathsf{Mac}(\mathsf{k}^{\mathsf{MAC}}, \cdot)$ to compute the tag. They are the only situations where the oracle is used during the simulation.

When the simulation of G_5 is over, $\mathcal{E}$ returns $(\mathsf{m}, \mathsf{tag})$ to the challenger of sUnforg, where tag is the tag received by $\mathsf{s}^{\mathsf{Test}}$ and $\mathsf{m} = (\mathsf{s}^{\mathsf{Test}}_{\mathsf{ActorId}}, \mathsf{s}^{\mathsf{Test}}_{\mathsf{PeerId}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{pk}_{\mathsf{R}}, \mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{ct}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{ct})$. If $\mathsf{s}^{\mathsf{Test}}$ did not receive a tag, the output of $\mathcal{E}$ does not matter and it returns any couple.

We note that $\mathcal{E}$ provides a perfect simulation of G_5 to $\mathcal{A}$. Moreover, when the simulated instance of G_5 is won, we claim that $\mathcal{E}$ wins the game sUnforg. To see this, we first recall that in order for the simulated instance of G_5 to be won, the tag tag received by $\mathsf{s}^{\mathsf{Test}}$ needs to satisfy $\mathsf{MacVf}(\mathsf{k}, \mathsf{tag}, M) = 1$ and $(M, \mathsf{tag}) \notin Q$. $\mathsf{MacVf}(\mathsf{k}, \mathsf{tag}, M) = 1$ is true, so we focus on showing $(M, \mathsf{tag}) \notin Q$.

By definition of $\mathcal{E}$, in a valid instance, the messages that Q contains are of the form $M^* = (\mathsf{s}_{\mathsf{PeerId}}, \mathsf{s}_{\mathsf{ActorId}}, \mathsf{s}.\mathsf{pk}_{\mathsf{R}}, \mathsf{s}.\widetilde{\mathsf{ct}}, \mathsf{s}.\mathsf{ct})$ for sessions s with actor $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$ and that received $\mathsf{s}^{\mathsf{Test}}.\widetilde{\mathsf{ct}}$.

Because these sessions s are not origin session of $\mathsf{s}^{\mathsf{Test}}$, we can see that $(M^*, \mathsf{s}.\mathsf{tag}^{\mathsf{I}}) \neq (M, \mathsf{tag})$. As a result, $(M, \mathsf{tag}) \notin Q$.

We thus have $\mathsf{Adv}^{\mathsf{G}_5}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{sUnforg}}(\mathcal{E})$.

Proposition 3 ($(\mathsf{U} \wedge \mathsf{eCK^w})$-KeySec and $(\mathsf{U} \wedge \mathsf{eCK\text{-}PFS})$-KeySec-Security of DAKE^σ). *We consider an adversary $\mathcal{A}$ against the Key Secrecy of DAKE_2^σ or DAKE_3^σ. We suppose that $\mathcal{A}$ activates less than N parties and that for each fixed party P, it activates less than l sessions that have either actor or peer P. Then, in the standard model, for $\mathsf{Advantage} \in \{\mathsf{Adv}^{\mathsf{U}\wedge\mathsf{eCK^w}\text{-}\mathsf{KeySec}}_{\mathsf{DAKE}_2^\sigma}(\mathcal{A}), \mathsf{Adv}^{\mathsf{U}\wedge\mathsf{eCK^w}\text{-}\mathsf{KeySec}}_{\mathsf{DAKE}_3^\sigma}(\mathcal{A}), \mathsf{Adv}^{\mathsf{U}\wedge\mathsf{eCK}\text{-}\mathsf{PFS}\text{-}\mathsf{KeySec}}_{\mathsf{DAKE}_3^\sigma}(\mathcal{A})\}$ there exist adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}$ such that,*

$$\mathsf{Advantage} \leq \mathsf{Adv}^{\mathsf{MBS}}_{\mathsf{Sign}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{M\text{-}S\text{-}UEO}}_{\mathsf{Sign}}(\mathcal{C}) + N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\mathsf{Sign}}(\mathcal{D}) + N^2 l \mathsf{Adv}^{\mathsf{sUF\text{-}CMA}}_{\mathsf{Sign}}(\mathcal{E}) .$$

Moreover, $\mathcal{E}$ queries $\mathcal{O}_{\mathsf{Sign}}$ at most l times.

Proof. We study the $\mathsf{U} \wedge \mathsf{eCK^w}$-KeySec security of DAKE_2^σ, the proof for the $\mathsf{U} \wedge \mathsf{eCK^w}$-KeySec and $\mathsf{U} \wedge \mathsf{eCK^w}$-KeySec security of DAKE_3^σ being similar.

Note that this demonstration can also be reused to establish Resp-explicit authentication for DAKE_2^σ and DAKE_3^σ, as explained in the following sidebar:

> *Reusing this demonstration to establish* Resp*-explicit authentication.*
> First, we note that the advantage in Resp-explicit authentication for the protocols DAKE_2^σ and DAKE_3^σ is exactly the probability that an instance of the Key Secrecy game satisfies the predicate $\mathsf{AisInit} \wedge \overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{FirstMatch}}$ and that the test session completes[35]. With this formulation, the test session corresponds to the session that the adversary must select in the Resp-explicit authentication game.
>
> Using this formulation of the Resp-explicit authentication game, we observe that the proof of Proposition 3 also establishes Resp-explicit authentication for DAKE_2^σ. Indeed, this proof bounds the probability that an instance of the game satisfies the predicate $\mathsf{AisInit} \wedge \overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{FirstMatch}}$ and that the test session completes. Moreover, the proof for DAKE_3^σ uses exactly the same arguments.

Game G_1: (Origin session implies first Match session) $\wedge$ (no signature collision). We recall that G_0 is the original security game for the AKE and that certain instances of this game can be designated as invalid. We call G_0 is the original security game for the AKE. We recall that certain instances of this game can be designated as invalid, which is indicated by the event $\mathsf{E}_{\mathsf{ValidInstance}}$ being false. An instance are invalid either if $\mathsf{U} \wedge \mathsf{eCK^w}$-KeySec is false or if no session has been tested. For an invalid instance, a random bit is returned instead of the adversary's result.

In G_1, additional instances are considered invalid. Specifically, any instance is deemed invalid if one of these two conditions is true :

- There exist two distinct sessions $\bar{\mathsf{s}}, \tilde{\mathsf{s}}$ with same responder party computing the same signature but for distinct messages.
- There exists a session s which is an origin session of $\mathsf{s}^{\mathsf{Test}}$ but not a first match of $\mathsf{s}^{\mathsf{Test}}$.

The first item directly creates a solution to the MBS game[36] by using an adversary that simulates G_0. The second item gives a signature $\sigma = \mathsf{s}.\sigma$, two messages

$$\mathsf{m}_1 = (\mathsf{s}^{\mathsf{Test}}_{\mathsf{ActorId}}, \mathsf{s}^{\mathsf{Test}}_{\mathsf{PeerId}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{pk}_{\mathsf{R}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{ct}) ,$$
$$\mathsf{m}_2 = (\mathsf{s}_{\mathsf{PeerId}}, \mathsf{s}_{\mathsf{ActorId}}, \mathsf{s}.\mathsf{pk}_{\mathsf{R}}, \mathsf{s}.\mathsf{ct}) ,$$

and two public keys: pk_1 the public key of $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$ and pk_2 the public key of $\mathsf{s}_{\mathsf{Actor}}$, such that :

$$\mathsf{SignVf}(\mathsf{pp}^{\mathsf{sign}}, \mathsf{pk}_1, \sigma, \mathsf{m}_1) = 1 \wedge \mathsf{SignVf}(\mathsf{pp}^{\mathsf{sign}}, \mathsf{pk}_2, \sigma, \mathsf{m}_2) = 1 ,$$
$$(\mathsf{pk}_1 \neq \mathsf{pk}_2) \vee (\mathsf{m}_1 \neq \mathsf{m}_2) ,$$

The inequalities being a direct consequence of the fact s is not a first matching session of $\mathsf{s}^{\mathsf{Test}}$.

We see that if $\mathsf{pk}_1 = \mathsf{pk}_2$, this leads to a solution of MBS while, if $\mathsf{pk}_1 \leq \mathsf{pk}_2$, this leads to a solution of M-S-UEO.

Thus, we can create two adversaries $\mathcal{B}, \mathcal{C}$ such that:

$$\left|\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A})\right| \leq \mathsf{Adv}^{\mathsf{MBS}}_{\mathsf{Sign}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{M\text{-}S\text{-}UEO}}_{\mathsf{Sign}}(\mathcal{C}) .$$

Game G_2: Removal of collisions of $\mathsf{pk}^{\mathsf{sign}}$. In G_2, additional instances are considered invalid. Specifically, any instance is deemed invalid if there exist two distinct parties with role Resp that have the same public static key,

[36] The complete definition of MBS and M-S-UEO are given in Section A.

This probability can be bounded by $N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\mathsf{Sign}}(\mathcal{B})$ for some adversary $\mathcal{D}$ of the key recovery game of Sign. Thus, we have:

$$\left|\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A})\right| \leq N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\mathsf{Sign}}(\mathcal{D})\,.$$

Game G_3: Guessing multiple elements. Same game than the game 3 of the proof of Proposition 2. We guess $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$,$\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$ $\mathsf{s}^{\mathsf{Test}}$. We have $\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A}) \leq N^2 l\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})$.

Game G_4: New win condition.

Same game than the game 4 of the proof of Proposition 2.

We create G_4 as G_3, with a new condition for victory. An instance of G_4 is won by the adversary $\mathcal{A}$ if and only if $\mathsf{s}^{\mathsf{Test}}$ complete without aborting and that nothing makes the instance of the game invalid (as per the conditions of G_3) until $\mathsf{s}^{\mathsf{Test}}$ completes. In particular, this means that the first message received by $\mathsf{s}^{\mathsf{Test}}$ contains valid signature.

We stop the game at the moment $\mathsf{s}^{\mathsf{Test}}$ completes as this is sufficient to check if it is won. Because we stop the game at the moment when $\mathsf{s}^{\mathsf{Test}}$ completes, this implies that $\mathsf{test\text{-}session}$ is not even requested for $\mathsf{s}^{\mathsf{Test}}$. In particular, we know that in a valid session of G_4 and the following games, $\mathsf{corrupt}$ is not asked to $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$. Thus, the valid instances verify the predicate $\mathsf{AisInit} \wedge \overline{\mathsf{CorP}} \wedge \overline{\mathsf{Orig}}$.

For G_3 to be won, any instance of G_3 needs to satisfy the win conditions of G_4. Thus, we have:

$$\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A})\,.$$

Analysis of G_4. We upper bound $\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A})$ by the advantage of an adversary $\mathcal{D}$ winning the strong forgery game $\mathsf{sUF\text{-}CMA}$ of the signature and using at most l times the oracle $\mathcal{O}_{\mathsf{Sign}}$.

$\mathcal{E}$ receives public parameters and a challenge public signature key $(\mathsf{pp}^{\mathsf{sign}}, \mathsf{pk}_\star)$, moreover, it has access to an oracle $\mathcal{O}_{\mathsf{Sign}}$ that can give signatures for the secret key associated to $\mathsf{pk}_\star$. When $\mathcal{O}_{\mathsf{Sign}}$ construct a signature σ' for a message m', it registers the couple (m', σ') in a database Q.

$\mathcal{E}$ simulates the game G_4 using $\mathsf{pk}_\star$ as the public key of $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$—note that this party is not corrupted in valid instances. When $\mathcal{D}$ needs to simulate the construction of the signature of a responder session with actor $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$, it uses the oracle $\mathcal{O}_{\mathsf{Sign}}$.

We denote by σ the signature received by $\mathsf{s}^{\mathsf{Test}}$. This signature is verified with public key $\mathsf{s}^{\mathsf{Test}}.\mathsf{pk}^{\mathsf{sign}}$ for the message $\mathsf{m} = (\mathsf{s}^{\mathsf{Test}}_{\mathsf{ActorId}}, \mathsf{s}^{\mathsf{Test}}_{\mathsf{PeerId}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{pk}_{\mathsf{R}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{ct})$. $\mathcal{D}$ returns (m, σ).

When the simulated instance wins G_5, (m, σ) wins the game $\mathsf{sUF\text{-}CMA}$. Indeed, we already know that in this case, $\mathsf{SignVf}(\mathsf{pp}^{\mathsf{sign}}, \mathsf{pk}^{\mathsf{sign}}, \mathsf{m}, \sigma) = 1$. Moreover, because there is no origin session of $\mathsf{s}^{\mathsf{Test}}$, we can see that $(\mathsf{m}, \sigma) \notin Q$.

We thus have $\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{sUF\text{-}CMA}}(\mathcal{E})$.

E.4 Key Secrecy for Predicates $\mathsf{V}_1, \cdots, \mathsf{V}_5$

As explained in E.1, the proof we are presenting now is the same for DAKE, its variant DAKE^σ, and even the protocol described in Figure 17, which has no responder authentication.[37] This is because we never rely on the authentication of the responder.

We provide the proof only for the protocol of Figure 17—the less secure one—which we denote by AKE, and by AKE_2 or AKE_3 when it is necessary to distinguish the two- or three-message variants.

We consider $\mathcal{A}$ as a PPT adversary participating in the KeySec game for AKE. We assume that $\mathcal{A}$ activates fewer than N parties, and that for each fixed party P, it activates fewer than l sessions, with P acting either as the actor or as the peer. Moreover, we recall that $\mathsf{2K\text{-}KEM}$ is δ-correct and γ-spread for some $\delta \geq 0$ and $\gamma > 0$.

The security analysis made in this section proceeds as follows:

- For $\mathsf{V}_1, \ldots, \mathsf{V}_4$ we analyze AKE_2; the proof extends verbatim to AKE_3.
- For V_5 we analyze AKE_3, since in this case the security of AKE_2 does not hold.

V_1-KeySec Security in the Standard Model for the Two- and Three-Message Variants. We recall that, as defined in (16) (page 48):

$$\mathsf{V}_1 = \mathsf{AisInit} \wedge \mathsf{FirstMatch} \wedge \overline{\mathsf{CorA}} \wedge \left(\overline{\mathsf{CorP}} \vee \overline{\mathsf{LeakRandO}}\right) \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,.$$

We study the security of AKE_3, whose only difference from AKE_2 is that the fourth game (which relies on the security of the PRG) is required only for AKE_3.

[37] Note that this protocol is not DUAKE: DUAKE adds a hash value in the first message sent by the responder.

Assume that dKEM is δ-correct and γ-spread. We will show that, in the standard model, there exist adversaries $\mathcal{B}, \mathcal{C}$ and $\mathcal{D}$ such that:

$$\mathsf{Adv}^{\mathsf{V}_1\text{-}\mathsf{KeySec}}_{\mathsf{AKE}_3}(\mathcal{A}) \leq Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}) + Nl^2\left(\delta + 2^{-\gamma+1}\right) + 2Nl^2\left(\mathsf{Adv}^{[\mathsf{IND\text{-}CCA},\,\text{-}\,]}_{\mathsf{dKEM}}(\mathcal{C}) + \mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D})\right).$$

Moreover, $\mathcal{C}$ queries the oracles $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$ at most $(l-1)$ times.

Note that the desired bound for AKE_2 is obtained simply by removing the advantage of $\mathsf{IND\text{-}PRG}$.

Proof. **Game G_1: Removal of collisions with $\mathsf{pk_R}$ and ct.**

We recall that G_0 is the original security game for the AKE and that certain instances of this game can be designated as invalid, which is indicated by the event $\mathsf{E_{ValidInstance}}$ being false. For an invalid instance, a random bit is returned instead of the adversary's result.

In G_1, additional instances are considered invalid. Specifically, any instance is deemed invalid if the following conditions hold:

- For any party P with role Init, there is a collision in the right key $\mathsf{pk_R}$ computed by the sessions with P as the actor.
- For any party P with role Init, there is a collision in the ciphertexts computed by the sessions with P as the peer.
- For any party P with role Init, a ciphertext ct exists such that:
  - A session with P as the actor receives a first message with ciphertext ct.
  - Later, a session with peer P, sends its first message, this message having ciphertext ct.

The probability of the first item can be bounded by $Nl\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B})$ for some adversary $\mathcal{B}$. For the second and third items, we leverage the spreadness of dKEM to bound the probability of these events by $2Nl^2 2^{-\gamma}$.

Thus, we have:

$$\left|\mathsf{Adv}^{\mathsf{G}_0}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A})\right| \leq Nl\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}) + 2Nl^2 2^{-\gamma}.$$

Note that in a valid instance of G_1, there exists a first matching session $\mathsf{s}^{\mathsf{FirstM}}$ of $\mathsf{s}^{\mathsf{Test}}$. It is in particular an origin session of $\mathsf{s}^{\mathsf{Test}}$ because $\mathsf{s}^{\mathsf{Test}}$ only receives one message. Moreover, it is the only origin session of $\mathsf{s}^{\mathsf{Test}}$.

Game G_2: Guessing $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, $\mathsf{s}^{\mathsf{Test}}$ and $\mathsf{s}^{\mathsf{FirstM}}$.

Note that, for a fixed party P, in any instance of G_1, we can enumerate each session s where P is either the actor or the peer, based on its order of creation. We denote its order by $l_{\mathsf{ActorOrPeer}=P}(\mathsf{s})$. By definition, we have $l_{\mathsf{ActorOrPeer}=P}(\mathsf{s}) \in \mathsf{Range}(l)$. G_2 is similar to G_1 with two modifications:

- We begin by uniformly sampling three numbers from respectively $\mathsf{Range}(N)$, $\mathsf{Range}(l)$ and $\mathsf{Range}(l)$.
- If these numbers do not respectively match the numbers $\mathsf{s}^{\mathsf{Test}}_{\mathsf{ActorInternId}}$, $l_{\mathsf{ActorOrPeer}=\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}}(\mathsf{s}^{\mathsf{Test}})$ and $l_{\mathsf{ActorOrPeer}=\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}}(\mathsf{s}^{\mathsf{FirstM}})$, the session is invalid. In such cases, instead of producing $\mathcal{A}$'s result, we output a uniformly random bit.

  Thus, we have $\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A}) \leq Nl^2\mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A})$.

Note that, because they are guessed at the beginning of the game, the long-term key $\mathsf{pk_R}$ for $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, the ephemeral key $\mathsf{pk_R}$ for $\mathsf{s}^{\mathsf{Test}}$ and the key and encapsulation computed by $\mathsf{s}^{\mathsf{Test}}$ can be precomputed.

Game G_3: Replacement of $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}$ and $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$ by a uniform key.

We create game G_3 where the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$ and $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}$ are replaced by a uniformly sampled key $\widetilde{\mathsf{k}}$.

To measure the difference of advantage between G_2 and G_3, we construct an adversary $\mathcal{C}$ against the $[\mathsf{IND\text{-}CCA},\,\text{-}\,]$ security of dKEM as follows:

- $\mathcal{C}$ receives $(\mathsf{pp}, \mathsf{pk}_{\mathsf{R},\star}, \mathsf{sk}_{\mathsf{R},\star}, \mathsf{r}_{\mathsf{R},\star}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{ct}_\star, \mathsf{k}_u)$ where:
  - if $u = 0$, $\mathsf{ct}_\star$ is an encapsulation of k_u with public keys $\mathsf{pk}_{\mathsf{L},\star}$ and $\mathsf{pk}_{\mathsf{L},\star}$.
  - if $u = 1$, k_u has been uniformly, independently sampled.

  $\mathcal{C}$ starts by simulating G_2 to the adversary $\mathcal{A}$, using pp as public parameters of the protocol, $\mathsf{pk}_{\mathsf{L},\star}$ as the static public key of $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, $\mathsf{pk}_{\mathsf{R},\star}$ as $\mathsf{s}^{\mathsf{Test}}.\mathsf{pk_R}$, $\mathsf{sk}_{\mathsf{R},\star}$ as $\mathsf{s}^{\mathsf{Test}}.\mathsf{sk_R}$, $\mathsf{ct}_\star$ as $\mathsf{s}^{\mathsf{Test}}.\mathsf{ct}$, the ciphertext created by $\mathsf{s}^{\mathsf{FirstM}}$ and k_u as the key encapsulated by $\mathsf{s}^{\mathsf{FirstM}}$ under $\mathsf{ct}_\star$. $\mathcal{C}$ also uses k_u as the key decapsulated by $\mathsf{s}^{\mathsf{Test}}$, instead of computing its decapsulation.
- If, at any moment, the (simulated) instance of G_2 becomes invalid, $\mathcal{C}$ stops the simulation and returns a uniform bit. We will assume in next items that this does not occur.
- For each session $\mathsf{s} \neq \mathsf{s}^{\mathsf{Test}}$ such that $\mathsf{s}_{\mathsf{Actor}} = \mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, each couple of keys $(\mathsf{pk_R}, \mathsf{sk_R})$, normally created by $\mathsf{Keygen_R}(\mathsf{pp})$ in the first message of s, is instead computed by $\mathcal{C}$ with $\mathcal{O}_{\mathsf{leak_R}}$. This allows $\mathcal{C}$ to use $\mathcal{O}_{\mathsf{CCA_L}}$ on them.[38]

[38] Note that $\mathcal{O}_{\mathsf{leak_R}}$ also provides the randomness used to compute them, which can then be returned when $\mathsf{eph\text{-}key}$ is queried.

– For each session $\mathsf{s} \neq \mathsf{s}^{\mathsf{Test}}$ such that $\mathsf{s}_{\mathsf{Actor}} = \mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, $\mathcal{C}$ uses $\mathcal{O}_{\mathsf{CCA_L}}$ to compute the decapsulation.

A direct analysis of $\mathcal{C}$ shows that:

– if $u = 0$, $\mathcal{C}$ almost perfectly simulates G_2 to $\mathcal{A}$. The only issue with the simulation is that the decapsulation of the ciphertext received by $\mathsf{s}^{\mathsf{Test}}$ always succeeds, while in the real game, its result could be different of the encapsulated key. This difference can be measured by the correctness of dKEM, and we have: $|\Pr[\mathcal{C} \text{ outputs } 1 \mid u = 0] - \Pr[1 \leftarrow \mathsf{G}_2(\mathcal{A})]| \leq \delta$.

– if $u = 1$, $\mathcal{C}$ perfectly simulates G_3 to $\mathcal{A}$. Thus, $\Pr[\mathcal{C} \text{ outputs } 1 \mid u = 1] = \Pr[1 \leftarrow \mathsf{G}_3(\mathcal{A})]$.

Finally, we have[39]:

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{[\mathsf{IND\text{-}CCA},\,\text{-}\,]}_{\mathsf{dKEM}}(\mathcal{C}) + \delta\,.$$

We also note that, by construction, $\mathcal{C}$ uses at most $l - 1$ times the oracles $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$.

Game G_4: Replacement of $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{SESSION}}$ and $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}^{\mathsf{SESSION}}$ by a uniform key (together with their MAC keys).

We create game G_4 where:

– the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{SESSION}}$ and $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}^{\mathsf{SESSION}}$ are replaced by a uniformly sampled key.

– the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{MAC}}$ and $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}^{\mathsf{MAC}}$ are replaced by a uniformly sampled key.

To measure the difference of advantages between G_3 and G_4, we construct an adversary $\mathcal{D}$ against $\mathsf{IND\text{-}PRG}$:

– We denote by u the bit sampled by the challenger. The adversary is given out_u and tries to guess u where :
  - $\mathsf{out}_1 = \mathsf{PRG}(\mathsf{k})$ for a key k uniformly taken in the set of session keys.
  - out_0 is uniformly taken in the image of PRG.

– $\mathcal{D}$ simulates G_4, but uses out_u instead of $(\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{SESSION}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{MAC}})$ and of $(\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}^{\mathsf{SESSION}}, \mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}^{\mathsf{MAC}})$.

– If, at any moment, the—simulated—instance of G_4 becomes invalid, $\mathcal{D}$ returns a uniform bit.

– At the end of the simulation, $\mathcal{D}$ returns 1 if, and only if, $\mathcal{A}$ wins.

We can see that:

– When $u = 0$, $\mathcal{D}$ perfectly simulates G_3 to $\mathcal{A}$. Hence, $\Pr[\mathcal{D} \text{ outputs } 1 \mid u = 0] = \Pr[1 \leftarrow \mathsf{G}_3(\mathcal{A})]$.

– When $u = 1$, $\mathcal{D}$ perfectly simulates G_4 to $\mathcal{A}$. Hence, $\Pr[\mathcal{D} \text{ outputs } 1 \mid u = 1] = \Pr[1 \leftarrow \mathsf{G}_4(\mathcal{A})]$.

Finally, we have:

$$\left|\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D}).$$

Study of G_4 and conclusion. In G_4, whether b is equal to 0 or 1, the session key of the tested session is uniformly sampled. Consequently, $\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) = 0$, allowing us to conclude.

V_2-KeySec Security in the Standard Model for the Two- and Three-Message Variants. We recall that, as defined in (16) (page 48):

$$\mathsf{V}_2 = \mathsf{AisResp} \wedge \overline{\mathsf{CorP}} \wedge \left(\overline{\mathsf{CorA}} \vee \overline{\mathsf{LeakRandT}}\right) \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,.$$

We study the security of AKE_3, whose only difference from AKE_2 is that the fourth game (which relies on the security of the PRG) is required only for AKE_3.

Assume that dKEM is δ-correct and γ-spread. We will show that, in the standard model, there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ such that:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{V_2\text{-}KeySec}}_{\mathsf{AKE}_2}(\mathcal{A}) \leq & Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}) + 2Nl^2\left(\delta + 2^{-\gamma}\right) + \\ & + 4Nl^2\big(\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\,\text{-}\,]}_{\mathsf{dKEM}}(\mathcal{C}) + \mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D})\big).\end{aligned}$$

Moreover, $\mathcal{D}$ queries the oracles $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$ at most l times.

Note that the desired bound for AKE_2 is obtained simply by removing the advantage of $\mathsf{IND\text{-}PRG}$.

Proof. **Game G_1: Removal of some collisions with $\mathsf{pk_R}$ and ct.**

It is the same game as G_1 for V_1 (page 53), the difference of advantage being the same.

Thanks to G_1, we know that if a first-partner session exists, it is unique. If so, we will denote it by $\mathsf{s}^{\mathsf{FirstP}}$.

[39] Note that in the theorems summarizing the security of the protocol, we use [IND-CK-CCA, -] security in order to simplify the formulae, this notion is stronger than [IND-CCA, -] security : the adversary simply computes $(\mathsf{pk}_{\mathsf{R},\star}, \mathsf{sk}_{\mathsf{R},\star}, \mathsf{r}_{\mathsf{R},\star})$ with $\mathcal{O}_{\mathsf{leak_R}}$ instead of receiving them from the challenger.

Game G_2: Guessing multiple elements.

Using the same method as in G_2 for V_1 (see page 53), we guess $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$, $\mathsf{s}^{\mathsf{Test}}$, whether $\mathsf{s}^{\mathsf{FirstP}}$ exists (so, we guess a if yes or not the $\mathsf{s}^{\mathsf{FirstP}}$ exists) and, if so, we also guess $\mathsf{s}^{\mathsf{FirstP}}$.

We have $\mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A}) \leq 2Nl^2\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})$.

Note that, because they are guessed at the beginning of the game:

- if $\mathsf{s}^{\mathsf{FirstP}}$ exists, we can precompute the long-term key pk_{L} for $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$, the ephemeral key pk_{R} of $\mathsf{s}^{\mathsf{FirstP}}$ as well as the cipher and encapsulated key computed by $\mathsf{s}^{\mathsf{Test}}$.
- if $\mathsf{s}^{\mathsf{FirstP}}$ does not exist, we can only precompute the long-term key pk_{L} for $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$.

Game G_3: Replacement of $\mathsf{s}.\mathsf{k}$ and $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}$ (if it exists) by a uniform key.

We create game G_3 where the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$ and $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}$ (if $\mathsf{s}^{\mathsf{FirstP}}$ exists) are replaced by a uniformly sampled key $\widetilde{\mathsf{k}}$.

Since we make an initial guess in the game about the existence of $\mathsf{s}^{\mathsf{FirstP}}$, we can separately prove the scenarios where it exists and where it does not.

If $\mathsf{s}^{\mathsf{FirstP}}$ exists:

We measure the difference of advantage between G_2 and G_3 by the advantage of an adversary $\mathcal{C}$ against the [IND-CCA, -] security of dKEM, as made in the game G_3 of V_1, and we obtain the bound

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{[\mathsf{IND\text{-}CCA},\,\text{-}\,]}_{\mathsf{dKEM}}(\mathcal{C}) + \delta\,. \tag{18}$$

Moreover, $\mathcal{C}$ queries the oracles $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$ and $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ at most $(l-1)$ times.

If $\mathsf{s}^{\mathsf{FirstP}}$ does not exist:

In this case, we create an adversary $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ against the [IND-CK-CCA, -] security of dKEM, as follows:

- $\mathcal{C}_1$ receives $(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star})$ and begins by simulating G_2 to the adversary $\mathcal{A}$, using pp as public parameters of the protocol and $\mathsf{pk}_{\mathsf{L},\star}$ as the static public key of $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$.
- If, at any moment, the (simulated) instance of G_2 becomes invalid, $\mathcal{C}$ stops the simulation and returns a uniform bit. We will assume in next items that this does not occur.
- For each session s such that $\mathsf{s}_{\mathsf{Actor}} = \mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$ (including $\mathsf{s}^{\mathsf{Test}}$), each couple of keys $(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$, normally created by $\mathsf{Keygen}_{\mathsf{R}}(\mathsf{pp})$ in the first message of s, is instead computed by $\mathcal{C}_1$ with $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$. This allows $\mathcal{C}$ to use $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ on them.
- When $\mathsf{s}^{\mathsf{Test}}$ receives its first message $(\mathsf{id}, \mathsf{pk}_{\mathsf{R},\star})$, $\mathcal{C}_1$ sends $\mathsf{pk}_{\mathsf{R},\star}$ to the challenger[40]. $\mathcal{C}_2$ then receives a ciphertext $\mathsf{ct}_\star$ and a key k_u. We recall that u is a bit uniformly sampled by the challenger and that:
  - if $u = 0$, $\mathsf{ct}_\star$ is an encapsulation of k_u with public keys $\mathsf{pk}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}$.
  - if $u = 1$, k_u has been uniformly and independently sampled.

  $\mathcal{C}_2$ uses $\mathsf{ct}_\star$ and k_u respectively as $\mathsf{s}^{\mathsf{Test}}.\mathsf{ct}$ and $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$.
- When an initiator session s with actor $\mathsf{s}_{\mathsf{Actor}} = \mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$ receives its first message, decapsulation is computed by $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$. Note that because no session is a first matching session of $\mathsf{s}^{\mathsf{Test}}$, $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ will always compute a decapsulation (*i.e.*, it will not directly return $\perp$).
- If $\mathcal{C}_2$ did not return a uniform bit before $\mathcal{A}$ ends, it returns 1 if $\mathcal{A}$ wins the simulated game and else 0.

A direct analysis of $\mathcal{C}$ shows that:

- if $u = 0$, $\mathcal{C}$ perfectly simulates G_2 to $\mathcal{A}$. Thus, $\Pr[\mathcal{C} \text{ outputs } 1 \mid u = 1] = \Pr[1 \leftarrow \mathsf{G}_2(\mathcal{A})]$.
- if $u = 1$, $\mathcal{C}$ perfectly simulates G_3 to $\mathcal{A}$. Thus, $\Pr[\mathcal{C} \text{ outputs } 1 \mid u = 1] = \Pr[1 \leftarrow \mathsf{G}_3(\mathcal{A})]$.

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\,\text{-}\,]}_{\mathsf{dKEM}}(\mathcal{C}). \tag{19}$$

Moreover, $\mathcal{C}$ queries the oracles $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$ and $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ at most l times.

Conclusion of the two scenarios:

Using (18) and (19), and using the fact that chosen-key is stronger than non-chosen-key security , we can create an adversary $\mathcal{C}$ such that:

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\,\text{-}\,]}_{\mathsf{dKEM}}(\mathcal{C}) + \delta. \tag{20}$$

Moreover, $\mathcal{C}$ queries the oracles $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$ and $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ at most l times.

Game G_4: Replacement of $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{SESSION}}$ and $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}^{\mathsf{SESSION}}$ (if it exists) by a uniform key (together with their MAC keys).

We create game G_4 where:

[40] For concision, we omit state.

– the key $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{SESSION}}$ is replaced by a uniformly sampled key. If $\mathsf{s}^{\mathsf{FirstP}}$ exists, $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}^{\mathsf{SESSION}}$ is also replaced by the same key.
– the key $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{MAC}}$ is replaced by a uniformly sampled key. If $\mathsf{s}^{\mathsf{FirstP}}$ exists, $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}^{\mathsf{MAC}}$ is also replaced by the same key.

To measure the difference of advantages between G_3 and G_4, we construct an adversary $\mathcal{D}$ against IND-PRG, as done in the G_4 for the study of V_1-KeySec (page 54).

This gives:

$$\left|\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D}).$$

Study of G_4 and conclusion. In G_3, whether b is equal to 0 or 1, the session key of the tested session is uniformly sampled. Consequently, $\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) = 0$, allowing us to conclude.

V_3-KeySec Security in Both the Standard Model and the ROM for the Two- and Three-Message Variants We recall that, as defined in (16) (page 48), the predicate V_3 is equal to

$$\mathsf{AisInit} \wedge \mathsf{FirstMatch} \wedge \mathsf{CorA} \wedge \overline{\mathsf{LeakRandT}} \wedge \left(\overline{\mathsf{CorP}} \vee \overline{\mathsf{LeakRandO}}\right) \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,.$$

We study the security of AKE_3, whose only difference from AKE_2 is that the fourth game (which relies on the security of the PRG) is required only for AKE_3.

Assume that dKEM is δ-correct and γ-spread. We will show that in the standard model, there exist adversaries $\mathcal{B}$, and $\mathcal{C}$ such that:

$$\mathsf{Adv}^{\mathsf{V_3\text{-}KeySec}}_{\mathsf{AKE}_2}(\mathcal{A}) \leq Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}) + Nl^2\left(\delta + 2^{-\gamma+1}\right) + 2Nl^2\left(\mathsf{Adv}^{[\text{-},\mathsf{IND\text{-}CPA}]}_{\mathsf{dKEM}}(\mathcal{C}) + \mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D})\right).$$

Note that the desired bound for AKE_2 is obtained simply by removing the advantage of IND-PRG.

Proof. **Game G_1: Removal of some collisions with pk_{R} and ct.**

It is the same game as G_1 for V_1 (page 53), the difference of advantage being the same. Consequently, in any valid instance of G_1, there exists a unique origin session of $\mathsf{s}^{\mathsf{Test}}$, which is also a first matching session of $\mathsf{s}^{\mathsf{Test}}$. We denote this session as $\mathsf{s}^{\mathsf{FirstM}}$.

Game G_2: Guessing $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, $\mathsf{s}^{\mathsf{Test}}$ and $\mathsf{s}^{\mathsf{FirstM}}$.

Using the same method as in G_2 for V_1 (see page 53), we guess $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, $\mathsf{s}^{\mathsf{Test}}$ and $\mathsf{s}^{\mathsf{FirstM}}$. We have: $\mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A}) \leq Nl^2\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})$. Guessing those sessions at the beginning of the game allows to precompute the long-term key pk_{L} for $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, the ephemeral key pk_{R} for $\mathsf{s}^{\mathsf{Test}}$, the session key as well as the encapsulation computed by $\mathsf{s}^{\mathsf{FirstM}}$.

Game G_3: Replacement of $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$ and $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}$ by a uniform key.

We create game G_3 where the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$ and $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}$ are replaced by a uniformly sampled key $\widetilde{\mathsf{k}}$. To measure the difference of advantage between G_2 and G_3, we construct an adversary $\mathcal{C}$ against the $[\text{-},\mathsf{IND\text{-}CPA}]$ security of dKEM as follows:

– $\mathcal{C}$ receives $(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star}, \mathsf{r}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, \mathsf{ct}_\star, \mathsf{k}_u)$, where u is a bit uniformly sampled by the challenger such that:
  • If $u = 0$, $\mathsf{ct}_\star$ is an encapsulation of k_u with public keys $\mathsf{pk}_{\mathsf{L},\star}$ and $\mathsf{pk}_{\mathsf{R},\star}$.
  • If $u = 1$, k_u is uniformly and independently sampled.

  $\mathcal{C}$ starts by simulating G_2 to the adversary $\mathcal{A}$, using pp as public parameters of the protocol, $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star})$ as the static public and private keys of $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, $\mathsf{pk}_{\mathsf{R},\star}$ as $\mathsf{s}^{\mathsf{Test}}.\mathsf{pk}_{\mathsf{R}}$, $\mathsf{ct}_\star$ as $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{ct}$, the ciphertext created by $\mathsf{s}^{\mathsf{FirstM}}$ and k_u as the key encapsulated by $\mathsf{s}^{\mathsf{FirstM}}$ under $\mathsf{ct}_\star$. $\mathcal{C}$ also uses k_u as the key decapsulated by $\mathsf{s}^{\mathsf{Test}}$, instead of computing its decapsulation—note that $\mathsf{s}^{\mathsf{Test}}$ receives the cipher $\mathsf{ct}_\star$ because $\mathsf{s}^{\mathsf{FirstM}}$ is its first match session. Note also that $\mathcal{C}$ can give sk_{L} to the adversary on a $\mathsf{corrupt}(\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}})$ query.
– If anything makes the simulated session of G_2 invalid, $\mathcal{C}$ aborts and returns a uniform bit.
– If $\mathcal{C}$ did not return a uniform bit before $\mathcal{A}$ ends, it returns 1 if $\mathcal{A}$ wins the simulated game and else 0.

A direct analysis of $\mathcal{C}$ shows that:

– If $u = 0$, $\mathcal{C}$ almost perfectly simulates G_2 to $\mathcal{A}$. The only difference being that the decapsulation of the ciphertext received by $\mathsf{s}^{\mathsf{Test}}$ always succeeds, while in the real game, its result might differ from the encapsulated key. The probability that this happens can be upper bounded by the correctness of dKEM, hence $|\Pr[\mathcal{C} \text{ outputs } 1 \mid u = 0] - \Pr[1 \leftarrow \mathsf{G}_2(\mathcal{A})]| \leq \delta$.
– If $u = 1$, $\mathcal{C}$ perfectly simulates G_3 to $\mathcal{A}$, hence $\Pr[\mathcal{C} \text{ outputs } 1 \mid u = 1] = \Pr[1 \leftarrow \mathsf{G}_3(\mathcal{A})](\mathcal{A})$.

Finally, we deduce that:

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{[\,\cdot\,,\mathsf{IND\text{-}CPA}]}_{\mathsf{dKEM}}(\mathcal{C}) + \delta.$$

Game G_4: Replacement of $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{SESSION}}$ and $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}^{\mathsf{SESSION}}$ by a uniform key (together with their MAC keys).

We create game G_4 where:

- the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{SESSION}}$ and $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}^{\mathsf{SESSION}}$ are replaced by a uniformly sampled key.
- the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{MAC}}$ and $\mathsf{s}^{\mathsf{FirstM}}.\mathsf{k}^{\mathsf{MAC}}$ are replaced by a uniformly sampled key.

To measure the difference of advantages between G_3 and G_4, we construct an adversary $\mathcal{D}$ against IND-PRG, as done in the G_4 for the study of V_1-KeySec (page 54).

This gives:

$$\left|\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D}).$$

Study of G_4 and conclusion. In G_3, whether b is equal to 0 or 1, the session key of the tested session is uniformly sampled. Consequently, $\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) = 0$, allowing us to conclude.

V_4-KeySec Security in Both the Standard Model and the ROM for the Two- and Three-Message Variants We recall that, as defined in (16) (page 48):

$$\mathsf{V}_4 = \mathsf{AisResp} \wedge \mathsf{CorP} \wedge \mathsf{Orig} \wedge \overline{\mathsf{LeakRandO}} \wedge \left(\overline{\mathsf{CorA}} \vee \overline{\mathsf{LeakRandT}}\right) \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,,$$

We study the security of AKE_3, whose only difference from AKE_2 is that the fourth game (which relies on the security of the PRG) is required only for AKE_3.

Assume that dKEM is δ-correct and γ-spread. We will show that, in the standard model, there exist adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ such that:

$$\mathsf{Adv}^{\mathsf{V}_4\text{-}\mathsf{KeySec}}_{\mathsf{AKE}_2}(\mathcal{A}) \leq Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}) + Nl^2\left(\delta + 2^{-\gamma+1}\right) + 2Nl^2\left(\mathsf{Adv}^{[\,\cdot\,,\mathsf{IND\text{-}CCA}]}_{\mathsf{dKEM}}(\mathcal{C}) + \mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D})\right).$$

Moreover, $\mathcal{C}$ does not query $\mathcal{O}_{\mathsf{leak}_\mathsf{L}}$ and queries $\mathcal{O}_{\mathsf{CCA}_\mathsf{R}}$ at most one time.

Note that the desired bound for AKE_2 is obtained simply by removing the advantage of IND-PRG.

Proof. **Game G_1: Removal of some collisions with pk_R and ct.**

It is the same game as G_1 for V_1 (page 53), the difference of advantage being the same. Consequently, in any valid instance of G_1, there exists a unique origin session for $\mathsf{s}^{\mathsf{Test}}$. We denote this session as $\mathsf{s}^{\mathsf{Origin}}$. Note that, however, it is possible that this session is not a first-partner session.

Game G_2: Guessing $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, $\mathsf{s}^{\mathsf{Test}}$ and $\mathsf{s}^{\mathsf{Origin}}$.

Using the same method as in G_2 for V_1 (see page 53), we guess $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, $\mathsf{s}^{\mathsf{Test}}$ and $\mathsf{s}^{\mathsf{Origin}}$. We have $\mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A}) \leq Nl^2\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})$.

Guessing those sessions at the beginning of the game allows to precompute the long-term key pk_L for $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Actor}}$, the ephemeral key pk_R for $\mathsf{s}^{\mathsf{Test}}$, the session key as well as the encapsulation computed by $\mathsf{s}^{\mathsf{Origin}}$.

Game G_3: Replacement of $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$ and $\mathsf{s}^{\mathsf{Origin}}.\mathsf{k}$ (if it is first-partner) by a uniform key.

We create game G_3 where:

- $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$ is replaced by a uniformly sampled key $\widetilde{\mathsf{k}}$.
- if the cipher $\mathsf{s}^{\mathsf{Origin}}.\mathsf{ct}$ received by $\mathsf{s}^{\mathsf{Origin}}$ is equal to the cipher $\mathsf{s}^{\mathsf{Test}}.\mathsf{ct}$ computed by $\mathsf{s}^{\mathsf{Test}}$, the key $\mathsf{s}^{\mathsf{Origin}}.\mathsf{k}$ is also replaced by $\widetilde{\mathsf{k}}$. Note that this condition is equivalent to $\mathsf{s}^{\mathsf{Origin}}$ being a first-partner session of $\mathsf{s}^{\mathsf{Test}}$.

To measure the difference of advantage between G_2 and G_3, we construct an adversary $\mathcal{C}$ against the $[\,\cdot\,,\mathsf{IND\text{-}CCA}]$ security of dKEM as follows:

- $\mathcal{C}$ receives $(\mathsf{pp}, \mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star}, \mathsf{r}_{\mathsf{L},\star}, \mathsf{pk}_{\mathsf{R},\star}, \mathsf{ct}_\star, \mathsf{k}_u)$ where:
  - if $u = 0$, $\mathsf{ct}_\star$ is an encapsulation of k_u with public keys $\mathsf{pk}_{\mathsf{L},\star}$ and $\mathsf{pk}_{\mathsf{L},\star}$.
  - if $u = 1$, k_u has been uniformly, independently sampled.

  $\mathcal{C}$ starts by simulating G_2 to the adversary $\mathcal{A}$, using pp as public parameters of the protocol, $\mathsf{pk}_{\mathsf{R},\star}$ as $\mathsf{s}^{\mathsf{Origin}}.\mathsf{pk}_\mathsf{R}$ and $(\mathsf{pk}_{\mathsf{L},\star}, \mathsf{sk}_{\mathsf{L},\star})$ as the long-term keys of $\mathsf{s}^{\mathsf{Origin}}_{\mathsf{Actor}}$—note in particular that, because $\mathcal{C}$ knows $\mathsf{sk}_{\mathsf{L},\star}$, it can gives it to $\mathcal{A}$ if $\mathcal{A}$ uses corrupt on it.
- If, at any moment, the (simulated) instance of G_2 becomes invalid, $\mathcal{C}$ stops the simulation and returns a uniform bit. We will assume in next items that this does not occur.

– When $\mathsf{s}^{\mathsf{Test}}$ computes its first message, $\mathcal{A}$ uses $\mathsf{ct}_\star$ as $\mathsf{s}^{\mathsf{Test}}.\mathsf{ct}$ and k_u as $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$.
– When $\mathsf{s}^{\mathsf{Origin}}$ receives its first message, if the cipher is equal to $\mathsf{ct}_\star$, $\mathcal{C}$ uses k_u as the key decapsulated by $\mathsf{s}^{\mathsf{Origin}}$, instead of computing its decapsulation. Else, $\mathcal{C}$ uses the oracle $\mathcal{O}_{\mathsf{CCA}_\mathsf{R}}$ to compute the decapsulation.

A direct analysis of $\mathcal{C}$ shows that:

– if $u = 0$, $\mathcal{C}$ almost perfectly simulates G_2 to $\mathcal{A}$. The only issue with the simulation is that the decapsulation of the ciphertext received by $\mathsf{s}^{\mathsf{Origin}}$ always succeeds, while in the real game, its result could be different of the encapsulated key. This difference can be measured by the correctness of dKEM, and we have: $|\Pr[\mathcal{C} \text{ outputs } 1 \mid u = 0] - \Pr[1 \leftarrow \mathsf{G}_2(\mathcal{A})]| \leq \delta$.
– if $u = 1$, $\mathcal{C}$ perfectly simulates G_3 to $\mathcal{A}$. Thus, $\Pr[\mathcal{C} \text{ outputs } 1 \mid u = 1] = \Pr[1 \leftarrow \mathsf{G}_3(\mathcal{A})]$.

Finally, we deduce that:

$$\left|\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{[\cdot,\mathsf{IND\text{-}CCA}]}_{\mathsf{dKEM}}(\mathcal{C}) + \delta.$$

Moreover, $\mathcal{C}$ does not query $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$ and queries $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ at most one time.

We create game G_4 where:

– the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{SESSION}}$ and $\mathsf{s}^{\mathsf{Origin}}.\mathsf{k}^{\mathsf{SESSION}}$ are replaced by a uniformly sampled key.
– the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{MAC}}$ and $\mathsf{s}^{\mathsf{Origin}}.\mathsf{k}^{\mathsf{MAC}}$ are replaced by a uniformly sampled key.

To measure the difference of advantages between G_3 and G_4, we construct an adversary $\mathcal{D}$ against $\mathsf{IND\text{-}PRG}$, as done in the G_4 for the study of $\mathsf{V}_1\text{-}\mathsf{KeySec}$ (page 54).

This gives:

$$\left|\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D}).$$

Study of G_4 and conclusion. In G_4, whether b is equal to 0 or 1, the session key of the tested session is uniformly sampled. Consequently, $\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) = 0$, allowing us to conclude.

V_5-KeySec Security in the Standard Model for the Three-Message Variant. We recall that, as defined in (16) (page 48):

$$\mathsf{V}_5 = \mathsf{AisResp} \wedge \mathsf{corPafterT} \wedge \overline{\mathsf{Orig}} \wedge \left(\overline{\mathsf{CorA}} \vee \overline{\mathsf{LeakRandT}}\right) \wedge \overline{\mathsf{SkT}} \wedge \overline{\mathsf{SkM}}\,.$$

We study the security of AKE_3, as AKE_2 is not secure against $\mathsf{V}_5\text{-}\mathsf{KeySec}$. Assume that dKEM is δ-correct and γ-spread. We show that, in the standard model, there exist adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}$ and $\mathcal{E}$ such that:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{V}_5\text{-}\mathsf{KeySec}}_{\mathsf{AKE}_3}(\mathcal{A}) \leq\ & Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}) + 2Nl^2\left(\delta + 2^{-\gamma}\right) \\ & + 2Nl^2\left(2\,\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\,\cdot]}_{\mathsf{dKEM}}(\mathcal{C}) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D}) + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{E})\right)\,.\end{aligned}$$

Moreover, $\mathcal{C}$ queries $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$ and $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ at most l times, and $\mathcal{E}$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most one time.

Proof. We observe that this proof can also be reused to establish Init-explicit authentication for DAKE_3, DAKE_3^σ, and DUAKE_3

> *Reusing this demonstration to establish* Init*-explicit authentication.*
> First, we note that the advantage in Init-explicit authentication for the protocols $\mathsf{DAKE}_3, \mathsf{DAKE}_3^\sigma$ and DUAKE_3 is exactly the probability that an instance of the Key Secrecy game satisfies the predicate $\mathsf{AisInit} \wedge \overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{Match}}$ and that the test session completes, since in these protocols the initiator sessions send the final message. With this formulation, the test session corresponds to the session that the adversary must select in the Resp-explicit authentication game.
>
> We now consider the case of DAKE_3 and DAKE_3^σ: the case of DUAKE_3 will be treated at the end of this sidebar.
>
> We assume here that the reader is already familiar with the proof of V_5, and we only highlight the modifications needed to obtain Resp-explicit authentication.
>
> Although V_5 imposes additional restrictions on adversarial queries, the only restriction actually used in the proof is that $\mathsf{corrupt}$ is not applied to the peer of $\mathsf{s}^{\mathsf{Test}}$ before $\mathsf{s}^{\mathsf{Test}}$ completes.
>
> To bound the probability of $\mathsf{AisInit} \wedge \overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{Match}}$, we can therefore reuse the games $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_4$ without any modification. Unlike in the proof of V_5, the use of G_3 is unnecessary here: we do not change

the winning condition, which remains $\mathsf{AisInit} \wedge \overline{\mathsf{corPbeforeT}} \wedge \overline{\mathsf{Match}}$ together with the completion of the test session.

The analysis of G_5 can also be reused, but here we crucially rely on the value of the message contained in the MAC constructed by the initiator, whereas in the proof of V_5 the exact value of this message is not relevant. Specifically:

- The proof of V_5 would work identically if the protocol were modified so that the message of the initiator's MAC were always 0.
- Here, however, the value of the message is essential: in G_5 it ensures that if an honest initiator generated the MAC received by $\mathsf{s}^{\mathsf{Test}}$, then this honest session is in fact a matching session with $\mathsf{s}^{\mathsf{Test}}$.

The case of DUAKE_3.
We observe that DUAKE is extremely close to the protocol AKE_3 (Figure 17), the only difference being that DUAKE includes an additional hash. This similarity between DUAKE_3 and AKE_3 implies that the argument for Resp-authentication explained above applies identically to DUAKE_3, since the additional hash is not relevant in this context.

Game G_1: Removal of some collisions with pk_R and ct.

It is the same game as G_1 for V_1 (page 53), the difference of advantage being the same.

Note that by the definition of V_5, there is no origin session for the test session in a valid instance of the game. However, it is still possible that a first-partner session exists.

Thanks to G_1, we know that if a first-partner session exists, it is unique. If such a session exists, we will denote it as $\mathsf{s}^{\mathsf{FirstP}}$.

Game G_2: Guessing multiple elements.

Using the same method as in G_2 for V_1 (see page 53), we guess $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$, $\mathsf{s}^{\mathsf{Test}}$, if $\mathsf{s}^{\mathsf{FirstP}}$ exists and, if so, $\mathsf{s}^{\mathsf{FirstP}}$.

We have $\mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A}) \leq 2Nl^2\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A})$.

Game G_3: New win condition.

We create G_3 as G_2, with a new condition for victory. An instance of G_3 is won by the adversary $\mathcal{A}$ if, and only if, $\mathsf{s}^{\mathsf{Test}}$ receives a valid tag (and thus completes) and nothing makes the instance of the game invalid (as per the conditions of G_2) until $\mathsf{s}^{\mathsf{Test}}$ receives the valid tag.

We stop the game at the moment when $\mathsf{s}^{\mathsf{Test}}$ completes—this implies that test-session is not even requested for $\mathsf{s}^{\mathsf{Test}}$.

For G_2 to be won, any instance of G_2 needs to satisfy the win conditions of G_3. Thus, we have $\mathsf{Adv}^{\mathsf{G}_2}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})$.

An important consequence of the definition of G_3 is that in all valid instances, $\mathsf{s}^{\mathsf{Test}}_{\mathsf{Peer}}$ remains uncorrupted.

Game G_4: Replacement of $\mathsf{s.k}$ and $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}$ (if it exists) by a uniform key.

We create game G_4 where the keys $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}$ and $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}$ (if $\mathsf{s}^{\mathsf{FirstP}}$ exists) are replaced by a uniformly sampled key $\widetilde{\mathsf{k}}$.

To study the difference in advantage between G_3 and G_4, we use the same argument as in the game G_3 for the study of V_2-KeySec (page 54). This argument shows that we can construct an adversary $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ against the [IND-CK-CCA, -] security of dKEM such that:

$$\left|\mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\,\text{-}\,]}_{\mathsf{dKEM}}(\mathcal{C}) + \delta.$$

Moreover, $\mathcal{C}$ queries the oracles $\mathcal{O}_{\mathsf{leak}_\mathsf{R}}$ and $\mathcal{O}_{\mathsf{CCA}_\mathsf{L}}$ at most l times.

Game G_5: Replacement of $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{MAC}}$ and $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}^{\mathsf{MAC}}$ (if it exists) by a uniform key (together with their session keys).

We create game G_5 where:

- the key $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{SESSION}}$ is replaced by a uniformly sampled key. If $\mathsf{s}^{\mathsf{FirstM}}$ exists, $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}^{\mathsf{SESSION}}$ is also replaced by the same key.
- the key $\mathsf{s}^{\mathsf{Test}}.\mathsf{k}^{\mathsf{MAC}}$ is replaced by a uniformly sampled key. If $\mathsf{s}^{\mathsf{FirstM}}$ exists, $\mathsf{s}^{\mathsf{FirstP}}.\mathsf{k}^{\mathsf{MAC}}$ is also replaced by the same key.

To measure the difference of advantages between G_4 and G_5, we construct an adversary $\mathcal{D}$ against IND-PRG, as done in the G_4 for the study of V_1-KeySec (page 54).

This gives:

$$\left|\mathsf{Adv}^{\mathsf{G}_5}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{G}_4}(\mathcal{A})\right| \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D}).$$

Analysis of G_5. We upper bound $\mathsf{Adv}^{\mathsf{G}_5}(\mathcal{A})$ by the advantage of an adversary $\mathcal{F}$ winning the game $\mathsf{sUnforg}$ of the MAC and that queries at most one times the oracle $\mathcal{O}_{\mathsf{Mac}}$.

$\mathcal{F}$ has access to an oracle $\mathcal{O}_{\mathsf{Mac}} = \mathsf{Mac}(\mathsf{k}, \cdot)$ for a key it does not know. When queried on an entry m, this oracle provides a tag tag such that $\mathsf{MacVf}(\mathsf{k}, \mathsf{tag}, m) = 1$ and registers the couple (m, tag) in a database Q. In order to win, $\mathcal{F}$ needs to provide a couple of message and tag $(m, \mathsf{tag}) \notin Q$ such that $\mathsf{MacVf}(\mathsf{k}, \mathsf{tag}, m) = 1$.

To do so, $\mathcal{F}$ simulates G_5 to $\mathcal{A}$ with one key difference: if $\mathsf{s}^{\mathsf{FirstP}}$ exists, $\mathcal{F}$ uses the oracle $\mathcal{O}_{\mathsf{Mac}}$ with message $\left(\mathsf{s}^{\mathsf{FirstP}}_{\mathsf{ActorId}}, \mathsf{s}^{\mathsf{FirstP}}_{\mathsf{PeerId}}, \mathsf{s}^{\mathsf{FirstP}}.\mathsf{pk}_{\mathsf{R}}, \mathsf{s}^{\mathsf{FirstP}}.\mathsf{ct}\right)$ in order to compute the tag computed by $\mathsf{s}^{\mathsf{FirstP}}$. It is the only situation where the oracle is used during the simulation.

Finally, when $\mathsf{s}^{\mathsf{Test}}$ receives a tag tag, $\mathcal{F}$ returns $(\mathsf{m}^{\mathsf{test}}, \mathsf{tag})$),to the challenger of $\mathsf{sUnforg}$, where [41]

$$\mathsf{m}^{\mathsf{test}} = \left(\mathsf{s}^{\mathsf{Test}}_{\mathsf{PeerId}}, \mathsf{s}^{\mathsf{Test}}_{\mathsf{ActorId}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{pk}_{\mathsf{R}}, \mathsf{s}^{\mathsf{Test}}.\mathsf{ct}\right) .$$

We note that $\mathcal{F}$ provides a perfect simulation of G_4 to $\mathcal{A}$ until it stops. Moreover, when the simulated instance of G_5 is won, we claim that $\mathcal{F}$ wins the game $\mathsf{sUnforg}$.

To see this, we first recall that in a valid instance of G_4, the tag tag received by $\mathsf{s}^{\mathsf{Test}}$ satisfies $\mathsf{MacVf}(\mathsf{k}, \mathsf{tag}, \mathsf{m}^{\mathsf{test}}) = 1$. We now need to verify that $(\mathsf{m}^{\mathsf{test}}, \mathsf{tag}) \notin Q$.

There are two possible cases:

- $\mathsf{s}^{\mathsf{FirstP}}$ exists. In this situation, we know that the tag sent to $\mathsf{s}^{\mathsf{Test}}$ is not equal to the tag created with the oracle—else, the session would be invalid as $\mathsf{s}^{\mathsf{FirstP}}$ would be an origin session. Consequently, $(\mathsf{m}^{\mathsf{test}}, \mathsf{tag}) \notin Q$.
- $\mathsf{s}^{\mathsf{FirstP}}$ does not exist. As a result, $Q = \emptyset$, and thus $(\mathsf{m}^{\mathsf{test}}, \mathsf{tag}) \notin Q$.

Hence, $\mathsf{Adv}^{\mathsf{G}_5}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{F})$ and this concludes the proof.

E.5 Key Secrecy of DUAKE

We can observe that DUAKE is extremely close of the protocol depicted in Figure 17 (Appendix E.1), the only difference being that DUAKE uses an additional hash.

Moreover, the predicates $\mathsf{F}^U_1, \cdots, \mathsf{F}^U_5$ (defined in Proposition 1 of Section C.4) are close to the predicates $\mathsf{V}_1, \cdots, \mathsf{V}_5$ under which the protocol of Figure 17 is studied (see Appendix E.1). More precisely, the only relevant difference is that F^U_1 and F^U_3 uses Orig while V_1 and V_3 use $\mathsf{FirstMatch}$.[42]

As a result, we can reuse the security proof of the protocol of Figure 17 if we can show that, in F^U_1 and F^U_3, the addition of the hash in DUAKE allows to say that:

$$(\mathsf{AisInit} \wedge \mathsf{Orig}) \Rightarrow (\mathsf{AisInit} \wedge \mathsf{FirstMatch}) .$$

We can directly see that this is true if there are no collisions between the hash computed by any of the responder sessions of the game. The security proof under $\mathsf{G}_1, \cdots, \mathsf{G}_5$ of the AKE of Figure 17 can thus be reused by just inserting an additional game to avoid the collisions of H.

As a result, we proved the Theorem 8.

F Explicit Authentication of DAKE and its Variants

In this section, we precisely study the explicit authentication of the protocols DAKE, DAKE^{σ}, or DUAKE. This analysis will establish that:

- Init-explicit authentication is achieved by DAKE_3, DAKE^{σ}_3 and DUAKE_3.
- Resp-explicit authentication is achieved by DAKE_3 and DAKE^{σ}_3.

[41] Note that the value of the message used to compute this tag differs across AKE_3, DAKE_3, DAKE^{σ}_3, and DUAKE_3. However, in each case it consists precisely of the transcript of the sent and received elements (except for the resulting tag).

[42] They are also some differences concerning requests that have absolutely no effect. More precisely, concerning using eph-key or corrupt on a responder session—which only return $\perp$ and thus gives no advantage to the adversary.

The complete bounds are given in Theorems 2 (stated in Section 3.3 and restated below for readability), 9 and 10.

In order to prove them, we will actually reuse the work made during the proof of key secrecy of these protocols. More precisely,

- The proof of Init-explicit authentication of $\mathsf{DAKE}_3, \mathsf{DAKE}_3^\sigma, \mathsf{DUAKE}_3$ can be derived from their key secrecy proof for the predicate V_5, as explained in the sidebar paragraph of Section E.4, page 58.
- The proof of Resp-explicit authentication of DAKE_2 and DAKE_3 can be derived from their key secrecy proof for the predicates $\mathsf{U} \wedge \mathsf{eCK}^{\mathsf{w}}$ and $\mathsf{U} \wedge \mathsf{eCK\text{-}PFS}$, as explained in the sidebar paragraph of Section E.3, page 49.
- The proof of Resp-explicit authentication of DAKE_2^σ and DAKE_3^σ can be derived from their key secrecy proof for the predicates $\mathsf{U} \wedge \mathsf{eCK}^{\mathsf{w}}$ and $\mathsf{U} \wedge \mathsf{eCK\text{-}PFS}$, as explained in the sidebar paragraph of Section E.3, page 51.

Theorem 2 (Explicit Authentication for DAKE). *Let $\mathcal{A}$ be a PPT adversary against* Init-ExplicitAuth *for* DAKE_3*. Suppose it activates at most N parties and, for each party* P*, at most l sessions with actor or peer* P*. Assume that the underlying* 2K-KEM *is γ-spread and δ-correct. Then, in the standard model, there exist PPT adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}$ and $\mathcal{E}$ such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Init\text{-}ExplicitAuth}}_{\mathsf{DAKE}_3}(\mathcal{A}) \leq\ & Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}) + 2Nl^2\left(\delta + 2^{-\gamma}\right) \\ & + 2Nl^2\left(2\,\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\,\cdot\,]}_{\mathsf{dKEM}}(\mathcal{C}) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D}) + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{E})\right).\end{aligned}$$

Moreover, $\mathcal{C}$ queries $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$ at most l times, and $\mathcal{E}$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most one time.

Now let $\Pi \in \{\mathsf{DAKE}_2, \mathsf{DAKE}_3\}$ and suppose that $\mathcal{A}$ is a PPT adversary against Resp-ExplicitAuth *for Π. Assume the same conditions on $\mathcal{A}$ as above, and that the standard* KEM *is δ'-correct. Then, in the standard model, there exist PPT adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}$ such that $\mathsf{Adv}^{\mathsf{Resp\text{-}ExplicitAuth}}_{\Pi}(\mathcal{A})$ is upper-bounded by:*

$$\mathsf{Adv}^{\mathsf{MacColl}}_{\mathsf{MAC}}(\mathcal{B}) + N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\widetilde{\mathsf{KEM}}}(\mathcal{C}) + N^2 l\left(2\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\widetilde{\mathsf{KEM}}}(\mathcal{D}) + l\delta' + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{E})\right).$$

Moreover, $\mathcal{D}$ queries $\mathcal{O}_{\mathsf{CCA}}$ at most l times and $\mathcal{E}$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most l times.

Theorem 9 (Resp-Explicit Authentication for DAKE_3^σ and DAKE_2^σ). *Let $\Pi \in \{\mathsf{DAKE}_3^\sigma, \mathsf{DAKE}_2^\sigma\}$ and let $\mathcal{A}$ be a PPT adversary against* Resp-ExplicitAuth *for Π. Suppose it activates at most N parties and, for each party* P*, at most l sessions with actor or peer* P*. Then, in the standard model, there exist PPT adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}$ and $\mathcal{E}$ such that:*

$$\mathsf{Adv}^{\mathsf{Resp\text{-}ExplicitAuth}}_{\Pi}(\mathcal{A}) \leq\ \mathsf{Adv}^{\mathsf{MBS}}_{\mathsf{Sign}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{M\text{-}S\text{-}UEO}}_{\mathsf{Sign}}(\mathcal{C}) + N\mathsf{Adv}^{\mathsf{Key\text{-}Recov}}_{\mathsf{Sign}}(\mathcal{D}) + N^2 l\mathsf{Adv}^{\mathsf{sUF\text{-}CMA}}_{\mathsf{Sign}}(\mathcal{E}).$$

Moreover, $\mathcal{E}$ queries $\mathcal{O}_{\mathsf{Sign}}$ at most l times.

Theorem 10 (Init-Explicit Authentication for DAKE_3^σ, and DUAKE_3). *Let $\Pi \in \{\mathsf{DAKE}_3^\sigma, \mathsf{DUAKE}_3\}$ and let $\mathcal{A}$ be a PPT adversary against* Init-ExplicitAuth *for Π. Suppose it activates at most N parties and, for each party* P*, at most l sessions with actor or peer* P*. Assume that the underlying* 2K-KEM *is γ-spread and δ-correct. Then, in the standard model, there exist PPT adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}$ and $\mathcal{E}$ such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Init\text{-}ExplicitAuth}}_{\Pi}(\mathcal{A}) \leq\ & Nl\,\mathsf{Adv}^{[\cdot,\mathsf{Key\text{-}Recov}]}_{\mathsf{dKEM}}(\mathcal{B}) + 2Nl^2\left(\delta + 2^{-\gamma}\right) \\ & + 2Nl^2\left(2\,\mathsf{Adv}^{[\mathsf{IND\text{-}CK\text{-}CCA},\,\cdot\,]}_{\mathsf{dKEM}}(\mathcal{C}) + 2\,\mathsf{Adv}^{\mathsf{IND\text{-}PRG}}_{\mathsf{PRG}}(\mathcal{D}) + \mathsf{Adv}^{\mathsf{sUnforg}}_{\mathsf{MAC}}(\mathcal{E})\right).\end{aligned}$$

Moreover, $\mathcal{C}$ queries $\mathcal{O}_{\mathsf{leak_R}}$ and $\mathcal{O}_{\mathsf{CCA_L}}$ at most l times, and $\mathcal{E}$ queries $\mathcal{O}_{\mathsf{Mac}}$ at most one time.

G Instantiating the Xue et al. AKE with One-Sided Chosen-Key Secure Double-KEMs

In [57], Xue *et al.* define an AKE that performs two encapsulations using a 2K-KEM, rather than one encapsulation with a 2K-KEM and another with a standard KEM, as in DAKE. In [57, Theorem 1], they

require the 2K-KEM to satisfy both [OW-CK-CCA, -] and [-, OW-CK-CPA] security. However, Maul does not meet [-, OW-CK-CPA], providing only [-, OW-CPA] security instead.[43,44]

However, we show in this section that a careful study of the demonstration reveals that [57, Theorem 1] still holds if we replace the [-, OW-CK-CPA] security requirement with [-, OW-CPA]. As a result, Maul can be used with their AKE. The rest of this section is devoted to explaining this.

The security of the AKE in [57] is proven in [57, Theorem 1]. We can directly see that the only place in the proof where the 2K-KEM's security is invoked is[57, Lemma 2], and[57, Table 4, p.12] shows that [-, OW-CK-CPA] is used only to bound the event $\mathsf{AskH} \wedge \mathsf{E}_5$. We therefore focus on this part of the proof, which appears in [57, Appendix 1, p.31].

To bound the probability of $\mathsf{AskH} \wedge \mathsf{E}_5$, the proof constructs an adversary $\mathcal{S}$ for the [-, OW-CK-CPA] game. $\mathcal{S}$ receives a right-hand key pair $(\mathsf{pk}_{A0}, \mathsf{sk}_{A0})$ from the [-, OW-CK-CPA] challenger, then uses the $\mathcal{O}_{\mathsf{leak_L}}$[45] oracle to obtain a left-hand key pair $(\mathsf{pk}_A, \mathsf{sk}_A)$, and finally provides pk_A to the challenger to obtain a challenge ciphertext and its corresponding (decapsulated) key. Since this is the [-, OW-CK-CPA] game, the challenger only generates $(\mathsf{pk}_{A0}, \mathsf{sk}_{A0})$; the left-hand key pk_A must be provided by $\mathcal{S}$. However, the key observation is that in this setting, $\mathcal{S}$ could equivalently generate $(\mathsf{pk}_A, \mathsf{sk}_A)$ using $\mathsf{Keygen_L}$[46] instead of $\mathcal{O}_{\mathsf{leak_L}}$.

We thus propose the following change to the proof: replace the [-, OW-CK-CPA] game with the [-, OW-CPA] game, where the [-, OW-CPA] challenger generates both key pairs $(\mathsf{pk}_{A0}, \mathsf{sk}_{A0})$ and $(\mathsf{pk}_A, \mathsf{sk}_A)$ via $\mathsf{Keygen_L}$, and sends $(\mathsf{pk}_A, \mathsf{sk}_A)$ to $\mathcal{S}$, along with pk_{A0}, the challenge ciphertext, and its corresponding key. To ensure that using [-, OW-CPA] leads to the same conclusion, we observes that in [-, OW-CK-CPA], creating $(\mathsf{pk}_A, \mathsf{sk}_A)$ via $\mathsf{Keygen_L}$ is equivalent to creating it via $\mathcal{O}_{\mathsf{leak1}}$, $\mathcal{O}_{\mathsf{leak1}}$ being impactful only in the CCA game [-, OW-CK-CCA][47]. Thus, in this case, using the [-, OW-CK-CPA] game with $\mathcal{S}$ calling $\mathcal{O}_{\mathsf{leak1}}$ is equivalent to using the [-, OW-CPA] game with the challenger providing $(\mathsf{pk}_A, \mathsf{sk}_A)$ by calling $\mathsf{Keygen_L}$. This change preserves the logic of the proof and therefore [-, OW-CPA] security suffices.

With this small modification to the proof, the 2K-KEM is only required to be [OW-CK-CCA, -] and [-, OW-CPA], making the AKE of [57] fully compatible with Maul.

H Additional Details Regarding Hint-MLWE and $\mathsf{Maul}_{\mathsf{PKE}}$

H.1 Hint-MLWE Hardness

Lattices and Gaussians A lattice $\Lambda \subset \mathbb{R}^n$ is defined as a discrete subgroup of $\mathbb{R}^n$. For a lattice $\Lambda \subset \mathbb{R}^n$, $\mathrm{span}(\Lambda)$ is the smallest linear subspace that contains Λ. The n-dimensional Gaussian function $\rho : \mathbb{R}^n \to (0, 1]$ is defined as $\rho(\mathbf{x}) := \exp(-\pi\|\mathbf{x}\|^2)$. Given a positive definite matrix $\mathbf{\Sigma} = \mathbf{B}\mathbf{B}^\intercal$, we can apply the invertible linear transformation $\mathbf{B}$ to define a Gaussian function $\rho_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x}) := \rho(\mathbf{B}^{-1}\mathbf{x}) = \exp(-\pi\mathbf{x}^t\mathbf{\Sigma}^{-1}\mathbf{x})$. For any $\mathbf{c} \in \mathrm{span}(\mathbf{\Sigma})$, the Gaussian function centered at $\mathbf{c}$ with parameter $\sqrt{\mathbf{\Sigma}}$ is $\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathbf{x}) := \rho_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x}-\mathbf{c})$. The discrete Gaussian distribution over a lattice Λ centered at $\mathbf{c}$ with parameter $\sqrt{\mathbf{\Sigma}}$ is defined by the probability mass function $D_{\Lambda,\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathbf{x}) := \frac{\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathbf{x})}{\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\Lambda)}$ for all $\mathbf{x} \in \Lambda$, where $\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\Lambda) := \sum_{\mathbf{x}\in\Lambda}\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}}(\mathbf{x})$. When $\mathbf{c} = \mathbf{0}$, we omit the center from the notation. When $\mathbf{\Sigma} = \sigma^2 \cdot \mathbf{I}_n$ for $\sigma > 0$ and $\mathbf{I}_n$ the $n \times n$ identity matrix, we write $D_{\Lambda,\sigma,\mathbf{c}}$ and refer to σ as the width parameter of $D_{\Lambda,\sigma,\mathbf{c}}$. We write $x \leftarrow\!\$\ D_{\mathbb{Z}^n,\sigma,\mathbf{c}}$ for $x \in \mathcal{R}$ to denote that the corresponding coefficient vector of x is sampled from $D_{\mathbb{Z}^n,\sigma,\mathbf{c}}$. For any positive $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest $s > 0$ such that $\rho(s \cdot \Lambda^*) \leq 1 + \varepsilon$, where $\Lambda^* := \{\mathbf{x} \in \mathrm{span}(\Lambda) \mid \langle \mathbf{x}, \Lambda\rangle \in \mathbb{Z}\}$ is the dual lattice of Λ.

Hint-MLWE Hardness In the following section, we work in the coefficient embedding, meaning that elements of $\mathcal{R}$ can be mapped to negacyclic matrices in $\mathbb{Z}^{n\times n}$ or to vectors in $\mathbb{Z}^n$.

Lemma 6 (Lemma 5 [37]). *For $0 < \varepsilon < 1/2$, a lattice Λ and a positive-definite matrix $\mathbf{\Sigma}$, we have*

$$s_1(\mathbf{\Sigma}^{-1}) \leq \eta_\varepsilon(\Lambda)^{-2} \quad \Rightarrow \quad \sqrt{\mathbf{\Sigma}} \geq \eta_\varepsilon(\Lambda).$$

[43] In [57], the notation [-, OW-CPA] actually refers to chosen-key security. Here, we use our notation to distinguish between chosen-key security ([-, OW-CK-CPA]) and standard one-way security ([-, OW-CPA]).
[44] Note that indistinguishability security directly implies one-way security.
[45] Called $\mathcal{O}_{\mathsf{leak1}}$ in [57].
[46] Note that in [57] they instead use the notation KeyGen1 for $\mathsf{Keygen_L}$, and KeyGen0 for $\mathsf{Keygen_R}$.
[47] The keys that can be used by the decapsulation oracle depends on the queries made to $\mathcal{O}_{\mathsf{leak1}}$.

Lemma 7 (Simplified Convolution Lemma [37]). *Let $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2$ be positive definite matrices such that*

$$\mathbf{\Sigma}_3^{-1} := \mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}$$

satisfies $\sqrt{\mathbf{\Sigma}_3} \geq \eta_\varepsilon(\mathbb{Z}^n)$ for $0 < \varepsilon < 1/2$. Then for an arbitrary $\mathbf{c} \in \mathbb{Z}^n$, the distribution

$$\left\{\mathbf{x}_1 + \mathbf{x}_2 \,\middle|\, \mathbf{x}_1 \leftarrow D_{\mathbb{Z}^n, \sqrt{\mathbf{\Sigma}_1}},\ \mathbf{x}_2 \leftarrow D_{\mathbb{Z}^n, \mathbf{c}, \sqrt{\mathbf{\Sigma}_2}}\right\}$$

is within statistical distance 2ε of $D_{\mathbb{Z}^n, \mathbf{c}, \sqrt{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}}$.

Lemma 8 (Derived from [37]). *Let ℓ, d be positive integers, $\mathbf{H}$ be a $d \times \ell$ matrix sampled from $\mathcal{F}$ in $\mathbb{Z}^{d \times \ell}$, $\sigma_1, \sigma_2 > 0$ be two real numbers and $\mathbf{\Sigma}_0 := (\frac{1}{\sigma_1^2} \cdot \mathbf{I}_\ell + \frac{1}{\sigma_2^2}\mathbf{H}^\top \cdot \mathbf{H})^{-1}$. If $\mathbf{\Sigma}_0$ is positive-definite, then the following two distributions over $\mathbb{Z}^\ell \times \mathbb{Z}^d$ are statistically identical:*

$$\left\{(\mathbf{r}, \mathbf{z}) \;\middle|\; \mathbf{r} \leftarrow D_{\mathbb{Z}^\ell, \sigma_1},\ \mathbf{y} \leftarrow D_{\mathbb{Z}^d, \sigma_2},\ \mathbf{z} = \mathbf{H} \cdot \mathbf{r} + \mathbf{y}\right\},$$

$$\left\{(\hat{\mathbf{r}}, \mathbf{z}) \;\middle|\; \mathbf{r} \leftarrow D_{\mathbb{Z}^\ell, \sigma_1},\ \mathbf{y} \leftarrow D_{\mathbb{Z}^d, \sigma_2},\ \mathbf{z} = \mathbf{H} \cdot \mathbf{r} + \mathbf{y},\ \mathbf{c} = \frac{1}{\sigma_2^2}\mathbf{\Sigma}_0 \cdot \mathbf{H}^\top \mathbf{z},\ \hat{\mathbf{r}} \leftarrow D_{\mathbb{Z}^\ell, \mathbf{c}, \mathbf{\Sigma}_0}\right\}.$$

Proof. The probability that the first random variable outputs $(\mathbf{v}, \mathbf{w}) \in \mathbb{Z}^\ell \times \mathbb{Z}^d$ can be written as follows:

$$\begin{aligned}
&\Pr\left[\mathbf{r} = \mathbf{v},\ \mathbf{H} \cdot \mathbf{r} + \mathbf{y} = \mathbf{w} \;\middle|\; \mathbf{r} \leftarrow D_{\mathbb{Z}^\ell, \sigma_1},\ \mathbf{y} \leftarrow D_{\mathbb{Z}^d, \sigma_2}\right] \\
&= D_{\mathbb{Z}^\ell, \sigma_1}(\mathbf{v}) \cdot D_{\mathbb{Z}^d, \sigma_2}(\mathbf{w} - \mathbf{H}\mathbf{v}) \\
&\propto \exp\left[-\pi\left(\frac{1}{\sigma_1^2} \cdot \mathbf{v}^\top \mathbf{v} + \frac{1}{\sigma_2^2} \cdot (\mathbf{w} - \mathbf{H}\mathbf{v})^\top (\mathbf{w} - \mathbf{H}\mathbf{v})\right)\right] \\
&= \exp\left[-\pi\left((\mathbf{v} - \mathbf{c})^\top \mathbf{\Sigma}_0^{-1}(\mathbf{v} - \mathbf{c}) - \mathbf{c}^\top \mathbf{\Sigma}_0^{-1}\mathbf{c} + \frac{1}{\sigma_2^2} \cdot \mathbf{w}^\top \mathbf{w}\right)\right]
\end{aligned}$$

where $\mathbf{c} = \frac{1}{\sigma_2^2}\mathbf{\Sigma}_0 \cdot \mathbf{H}^\top \mathbf{w}$. Hence, the conditional probability $\Pr[\mathbf{r} = \mathbf{v} \mid \mathbf{H} \cdot \mathbf{r} + \mathbf{y} = \mathbf{w}]$ is proportional to $\exp\left[-\pi(\mathbf{v} - \mathbf{c})^\top \mathbf{\Sigma}_0^{-1}(\mathbf{v} - \mathbf{c})\right]$, which implies

$$\Pr[\mathbf{r} = \mathbf{v} \mid \mathbf{H} \cdot \mathbf{r} + \mathbf{y} = \mathbf{w}] \equiv \rho_{\sqrt{\mathbf{\Sigma}_0}}(\mathbf{v} - \mathbf{c}) \equiv \Pr[\hat{\mathbf{r}} = \mathbf{v} \mid \mathbf{H} \cdot \mathbf{r} + \mathbf{y} = \mathbf{w}].$$

Therefore, the given two distributions are statistically identical. □

Theorem 4 (Hardness of Hint-MLWE, adapted from [37]).
Let k, m, ℓ, d, q, J be as in Definition 2, and let $\mathcal{F}$ be a distribution over

$$\mathcal{S}_B := \left\{\mathbf{H} \in \mathcal{R}_q^{\ell \times d} \;\middle|\; s_1(\mathbf{H})^2 \leq B\right\},$$

for $B > 0$. Let $\sigma_1, \sigma_2 > 0$, and define $\sigma > 0$ via $\frac{1}{\sigma^2} = 2\left(\frac{1}{\sigma_1^2} + \frac{B}{\sigma_2^2}\right)$. Let $\chi_h := D_{\mathbb{Z}^{\ell \cdot n}, \sigma_2}$, and let χ be a distribution over $\mathcal{R}^{k+m-d}$. Define χ_σ over $\mathcal{R}^{k+m}$ as the distribution where coordinates in J are drawn from $D_{\mathbb{Z}^{d \cdot n}, \sigma}$ and the rest from χ.

If $\sigma \geq \sqrt{2} \cdot \eta_\varepsilon(\mathbb{Z}^{d \cdot n})$ for some $0 < \varepsilon \leq 1/2$, then there exists an efficient reduction from $\mathsf{MLWE}^{\chi_\sigma}_{\mathcal{R},k,m,q}$ to $\mathsf{Hint\text{-}MLWE}^{\chi_{\sigma_1}, \chi_h, \mathcal{F}}_{\mathcal{R},k,m,q,\ell,d,J}$ that reduces the advantage by at most 2ε.

Proof. Let $\mathbf{H} \leftarrow \mathcal{F}$, $\mathbf{\Sigma}_0 = (\frac{1}{\sigma_1^2} \cdot \mathbf{I}_{nd} + \frac{1}{\sigma_2^2}\mathbf{H}^\top \cdot \mathbf{H})^{-1}$ and $(\mathbf{A}, \mathbf{b}) \in R_q^{m \times k} \times R_q^m$ be a given $\mathsf{MLWE}^{\chi_\sigma}_{R,k,m,q}$ instance. We suppose without loss of generality that the coordinates sample from χ are the last of $\binom{\mathbf{s}}{\mathbf{e}}$. To begin, we sample:

$$\mathbf{r}_1 \leftarrow D_{\mathbb{Z}^{dn}, \sigma_1}, \quad \mathbf{r}_2 \leftarrow \chi, \text{ and } \mathbf{y} \leftarrow D_{\mathbb{Z}^{\ell n}, \sigma_2},$$

$$\mathbf{t}_1 \leftarrow D_{\mathbb{Z}^{dn}, \mathbf{c}, \sqrt{\mathbf{\Sigma}_0 - \sigma^2 \cdot \mathbf{I}_{n\ell}}} \quad \text{for } \mathbf{c} = \frac{1}{\sigma_2^2}\mathbf{\Sigma}_0 \mathbf{H}^\top (\mathbf{H}\mathbf{r} + \mathbf{y}),$$

$$\mathbf{t}_2 \leftarrow \mathbf{0}^{n \cdot (k+m-d)}.$$

Define $(\mathbf{r}_1, \mathbf{r}_2)$ and $(\mathbf{t}_1, \mathbf{t}_2)$ as $\mathbf{r}$ and $\mathbf{t}$ respectively. Note that $\mathbf{\Sigma}_0 - \sigma^2 \cdot \mathbf{I}_{nd}$ is positive-definite, since the smallest singular value of $\mathbf{\Sigma}_0$ is higher than σ^2 (derived from $s_1(\mathbf{H})^2 \leq B$).

Then, we use the sampled polynomials to transform the given MLWE instance $(\mathbf{A}, \mathbf{b})$ into

$$(\mathbf{A},\ \mathbf{b} + [\mathbf{A} \mid \mathbf{I}_{nm}]\mathbf{t},\ \mathbf{H},\ \mathbf{z} = \mathbf{H} \cdot \mathbf{r}_1 + \mathbf{y})\ .$$

We distinguish two cases: the first where $\mathbf{b}$ is a MLWE sample, and the second where it is uniform. First, assume that $\mathbf{b} = [\mathbf{A} \mid \mathbf{I}_{nm}]\mathbf{r}'$ for $\mathbf{r}' \leftarrow \chi_\sigma$. Then, we have

$$\mathbf{b} + [\mathbf{A} \mid \mathbf{I}_{nm}]\mathbf{t} = [\mathbf{A} \mid \mathbf{I}_{nm}](\mathbf{r}' + \mathbf{t})\ ,$$

where $\mathbf{r}' + \mathbf{t}$ follows the distribution $\left(D_{\mathbb{Z}^{nd},\sigma} + D_{\mathbb{Z}^{nd},\mathbf{c},\sqrt{\mathbf{\Sigma}_0 - \sigma^2 \cdot \mathbf{I}_{nd}}}\right) \times \chi$.

To use the Convolution Lemma, we aim to show that $\sqrt{\mathbf{\Sigma}_3} \geq \eta_\varepsilon(\mathbb{Z}^{nd})$, where $\mathbf{\Sigma}_3^{-1} := \sigma^{-2} \cdot \mathbf{I}_{nd} + (\mathbf{\Sigma}_0 - \sigma^2 \cdot \mathbf{I}_{nd})^{-1}$. By Lemma 6, it suffices to show that $s_1(\mathbf{\Sigma}_3^{-1}) \leq \eta_\varepsilon(\mathbb{Z}^{nd})^{-2}$.

Recall that the smallest singular value of $\mathbf{\Sigma}_0 - \sigma^2 \cdot \mathbf{I}_{nd}$ is at least σ^2. Therefore, we obtain

$$s_1(\mathbf{\Sigma}_3^{-1}) \leq \sigma^{-2} + s_1((\mathbf{\Sigma}_0 - \sigma^2 \cdot \mathbf{I}_{nd})^{-1}) \leq \sigma^{-2} + \sigma^{-2} = 2\sigma^{-2} \leq \eta_\varepsilon(\mathbb{Z}^{nd})^{-2}.$$

So the distributions $D_{\mathbb{Z}^{nd},\sigma} + D_{\mathbb{Z}^{nd},\mathbf{c},\sqrt{\mathbf{\Sigma}_0 - \sigma^2 \mathbf{I}_{nd}}}$ are within the statistical distance 2ε of $D_{\mathbb{Z}^{nd},\mathbf{c},\sqrt{\mathbf{\Sigma}_0}}$. Therefore, the distribution of

$$(\mathbf{A},\ \mathbf{b} + [\mathbf{A} \mid \mathbf{I}_{nm}]\mathbf{t},\ \mathbf{H},\ \mathbf{z} = \mathbf{H} \cdot \mathbf{r}_1 + \mathbf{y})$$

is within statistical distance 2ε of

$$(\mathbf{A},\ [\mathbf{A} \mid \mathbf{I}_{nm}]\tilde{\mathbf{r}},\ \mathbf{H},\ \mathbf{z} = \mathbf{H} \cdot \mathbf{r}_1 + \mathbf{y}) \quad \text{for } \tilde{\mathbf{r}} \leftarrow D_{\mathbb{Z}^{nd},\mathbf{c},\sqrt{\mathbf{\Sigma}_0}} \times \chi.$$

As the last step, we apply Lemma 8 on $(\tilde{\mathbf{r}}, \mathbf{z})$, then by independence its distribution is identical to that of $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{z})$. As a result, the distribution of

$$(\mathbf{A},\ [\mathbf{A} \mid \mathbf{I}_{nm}]\tilde{\mathbf{r}},\ \mathbf{H},\ \mathbf{z} = \mathbf{H} \cdot \mathbf{r}_1 + \mathbf{y})\ ,$$

is identical to that of

$$(\mathbf{A},\ [\mathbf{A} \mid \mathbf{I}_{nm}]\mathbf{r},\ \mathbf{H},\ \mathbf{z} = \mathbf{H} \cdot \mathbf{r}_1 + \mathbf{y})\,,$$

which exactly follows the distribution of samples from $\mathsf{Hint\text{-}MLWE}^{\chi_{\sigma_1},\chi_h,\mathcal{F}}_{\mathcal{R},k,m,q,\ell,d,J}$ when $\mathbf{b}$ is not sample uniformly.

Secondly, if $\mathbf{b} \leftarrow \mathcal{U}(R_q^m)$, then $(\mathbf{A},\ \mathbf{b} + [\mathbf{A} \mid \mathbf{I}_{nm}]\mathbf{t},\ \mathbf{H},\ \mathbf{z} = \mathbf{H} \cdot \mathbf{r}_1 + \mathbf{y})$ follows the same distribution as $(\mathbf{A},\ \mathbf{u},\ \mathbf{H},\ \mathbf{z} = \mathbf{H} \cdot \mathbf{r}_1 + \mathbf{y})$ where $\mathbf{u} \leftarrow \mathcal{U}(R_q^m)$.

The reduction is then correct and reduces the advantage by at most 2ε. □

H.2 Complete Correctness and Security Proofs for $\mathsf{Maul}_{\mathsf{PKE}}$

In this section, we provide the complete proofs of Lemma 1 and Theorem 5.

Lemma 1 ($\mathsf{Maul}_{\mathsf{PKE}}$ correctness). *Using the notation from Figure 8, the scheme $\mathsf{Maul}_{\mathsf{PKE}}$ is δ-correct provided that*

$$\Pr\left[\left\|\begin{pmatrix} f_{\mathsf{L}} + e_{\mathsf{c_L}} + \mathbf{s}_{\mathsf{C}}^{\intercal}\mathbf{e}_{\mathsf{L}} - (\mathbf{e}_{\mathsf{C}} + \mathbf{e}_{\mathsf{c_C}})^{\intercal}\mathbf{s}_{\mathsf{L}} \\ f_{\mathsf{R}} + e_{\mathsf{c_R}} + \mathbf{s}_{\mathsf{C}}^{\intercal}\mathbf{e}_{\mathsf{R}} - (\mathbf{e}_{\mathsf{C}} + \mathbf{e}_{\mathsf{c_C}})^{\intercal}\mathbf{s}_{\mathsf{R}} \end{pmatrix}\right\|_\infty \leq \frac{q}{4}\right] > 1 - \delta\,,$$

where, $\mathbf{e}_{\mathsf{c_C}} = \mathsf{Decomp}_{(q,d_u)}(\mathsf{c_C}) - \mathbf{A}^{\intercal}\mathbf{s}_{\mathsf{C}} + \mathbf{e}_{\mathsf{C}}$, $e_{\mathsf{c_L}} = \mathsf{Decomp}_{(q,d_v)}(\mathsf{c_L}) - \mathbf{s}_{\mathsf{C}}^{\intercal}\mathbf{t}_{\mathsf{L}} + f_{\mathsf{L}} + m_{\mathsf{L}}$, *and* $e_{\mathsf{c_R}} = \mathsf{Decomp}_{(q,d_v)}(\mathsf{c_R}) - \mathbf{s}_{\mathsf{C}}^{\intercal}\mathbf{t}_{\mathsf{R}} + f_{\mathsf{R}} + m_{\mathsf{R}}$.

Proof. For a well-formed ciphertext $(\mathsf{c_C}, \mathsf{c_L}, \mathsf{c_R})$, define:

$$\begin{aligned}\mathbf{v}_{\mathsf{L}} &:= \mathsf{Decomp}_{(q,d_v)}(\mathsf{c_L}) - \mathsf{Decomp}_{(q,d_u)}(\mathsf{c_C})^{\intercal}\,\mathbf{s}_{\mathsf{L}} \\ &= \lceil q/2 \rceil \cdot m_{\mathsf{L}} + e_{\mathsf{c_L}} + \mathbf{s}_{\mathsf{C}}^{\intercal}\mathbf{t}_{\mathsf{L}} + f_{\mathsf{L}} - (\mathbf{e}_{\mathsf{c_C}} + \mathbf{A}^{\intercal}\mathbf{s}_{\mathsf{C}} + \mathbf{e}_{\mathsf{C}})^{\intercal}\mathbf{s}_{\mathsf{L}} \\ &= \lceil q/2 \rceil \cdot m_{\mathsf{L}} + e_{\mathsf{c_L}} + \mathbf{s}_{\mathsf{C}}^{\intercal}\mathbf{e}_{\mathsf{L}} + f_{\mathsf{L}} - (\mathbf{e}_{\mathsf{c_C}} + \mathbf{e}_{\mathsf{C}})^{\intercal}\mathbf{s}_{\mathsf{L}},\end{aligned}$$

and similarly $\mathbf{v}_{\mathsf{R}} := \lceil q/2 \rceil \cdot m_{\mathsf{R}} + e_{\mathsf{c_R}} + \mathbf{s}_{\mathsf{C}}^{\intercal}\mathbf{e}_{\mathsf{R}} + f_{\mathsf{R}} - (\mathbf{e}_{\mathsf{c_C}} + \mathbf{e}_{\mathsf{C}})^{\intercal}\mathbf{s}_{\mathsf{R}}$.

Thus, correct decryption—*i.e.* recovering $(m_{\mathsf{L}}, m_{\mathsf{R}}) = \lfloor (2/q) \cdot (\mathbf{v}_{\mathsf{L}}, \mathbf{v}_{\mathsf{R}}) \rceil$—is guaranteed if $\|\mathbf{v}_{\mathsf{L}} - \lceil q/2 \rceil \cdot m_{\mathsf{L}}\|_\infty \leq q/4$ and $\|\mathbf{v}_{\mathsf{R}} - \lceil q/2 \rceil \cdot m_{\mathsf{R}}\|_\infty \leq q/4$. □

Theorem 5 ([IND-CPA, IND-CPA] **Security of** $\mathsf{Maul}_{\mathsf{PKE}}$**).** *Let consider* $\mathsf{X} \in \{[\mathsf{IND\text{-}CPA}, \text{-}], [\text{-}, \mathsf{IND\text{-}CPA}]\}$ *and* $\mathcal{A}$ *a PPT adversary of the* X *security of* $\mathsf{Maul}_{\mathsf{PKE}}$*. There exist PPT adversaries* $\mathcal{B}, \mathcal{C}$ *with same time complexity as* $\mathcal{A}$ *such that:*

$$\mathsf{Adv}^{\mathsf{X}}_{\mathsf{Maul}_{\mathsf{PKE}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{MLWE}_{k,k,\mathcal{B}^k,\mathcal{B}^k}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}^{\chi,\chi_h,\mathcal{F}}_{\mathcal{R},k,k+1,q,1,2k,J}}(\mathcal{C}).$$

Where $\mathcal{F} = \mathcal{B}^k \times (-\mathcal{B}^k)$*,* $\chi_h = \mathcal{D}$*,* $\chi = \mathcal{B}^{2k} \times \mathcal{D}$ *and the subset* J *being the* $2k$ *first elements.*

Proof. By symmetry of the construction, it suffices to prove [IND-CPA, -] security. Furthermore, it is enough to consider the variant of the scheme in which the Compress and Decompress functions are replaced by the identity; we assume this is the case in the following.

We proceed with a game-based proof, with the full sequence of games provided in Figure 18.

$\mathsf{G}_0(\mathcal{A}) = [\mathsf{IND\text{-}CPA}, \text{-}](\mathcal{A})$

$\mathbf{A} \leftarrow\!\$\, \mathcal{U}\left(\mathcal{R}_q^{k\times k}\right)$
$\mathsf{sk}_\mathsf{L} = \mathbf{s}_\mathsf{L} \leftarrow\!\$\, \mathcal{B}^k$; $\mathsf{sk}_\mathsf{R} = \mathbf{s}_\mathsf{R} \leftarrow\!\$\, \mathcal{B}^k$
$\mathbf{e}_\mathsf{L} \leftarrow\!\$\, \mathcal{B}^k$; $\mathbf{e}_\mathsf{R} \leftarrow\!\$\, \mathcal{B}^k$
$\mathsf{pk}_\mathsf{L} = \mathbf{t}_\mathsf{L} \leftarrow \mathbf{A}\mathbf{s}_\mathsf{L} + \mathbf{e}_\mathsf{L}$
$\mathsf{pk}_\mathsf{R} = \mathbf{t}_\mathsf{R} \leftarrow \mathbf{A}\mathbf{s}_\mathsf{R} + \mathbf{e}_\mathsf{R}$
$(\mathsf{state}, m_{\mathsf{L},0}, m_{\mathsf{L},1}, m_\mathsf{R}) \leftarrow$
$\mathcal{A}_1(\mathbf{A}, \mathsf{pk}_\mathsf{R}, (\mathsf{sk}_\mathsf{R}, \mathbf{e}_\mathsf{R}), \mathsf{pk}_\mathsf{L})$
$(\mathbf{s}_\mathsf{C}, \mathbf{e}_\mathsf{C}) \leftarrow\!\$\, \mathcal{B}^k \times \mathcal{B}^k$
$(f_\mathsf{L}, f_\mathsf{R}) \leftarrow\!\$\, \mathcal{D} \times \mathcal{D}$
$\mathsf{c}_\mathsf{C} \leftarrow \mathbf{A}^\intercal \mathbf{s}_\mathsf{C} + \mathbf{e}_\mathsf{C}$
$c_{\mathsf{L},b} \leftarrow \mathbf{s}_\mathsf{C}^\intercal \mathbf{t}_\mathsf{L} + f_\mathsf{L} + \lceil q/2 \rfloor m_{\mathsf{L},b}$
$\mathsf{c}_\mathsf{R} \leftarrow \mathbf{s}_\mathsf{C}^\intercal \mathbf{t}_\mathsf{R} + f_\mathsf{R} + \lceil q/2 \rfloor m_\mathsf{R}$
$b \leftarrow\!\$\, \{0,1\}$
$b' \leftarrow \mathcal{A}_2\,(\mathsf{state}, (\mathsf{c}_\mathsf{C}, c_{\mathsf{L},b}, \mathsf{c}_\mathsf{R}))$
return $b = b'$

$\mathsf{G}_1(\mathcal{A})$

Two differences with G_0:
- $\mathsf{sk}_\mathsf{L}, \mathbf{e}_\mathsf{L}$ are no more defined
- $\mathsf{pk}_\mathsf{L} = \mathbf{t}_\mathsf{L} \leftarrow\!\$\, \mathcal{R}_q^k$

$\mathsf{G}_2(\mathcal{A})$

$\mathbf{B} \leftarrow\!\$\, \mathcal{U}\left(\mathcal{R}_q^{(k+1)\times k}\right)$
$\bar{\mathbf{s}} = \mathbf{s}_\mathsf{C} \leftarrow\!\$\, \chi = \mathcal{B}^k$
$\bar{\mathbf{e}} = (\mathbf{e}_\mathsf{C}^\intercal, f_\mathsf{L})^\intercal \leftarrow\!\$\, \chi = \mathcal{B}^k \times \mathcal{D}$
$(\mathsf{c}_\mathsf{C}^\intercal, \tilde{\mathsf{c}_\mathsf{L}})^\intercal \leftarrow \mathbf{B}\ \bar{\mathbf{s}} + \bar{\mathbf{e}}$
$\mathbf{H} = (\mathbf{e}_\mathsf{R}^\intercal, -\mathbf{s}_\mathsf{R}^\intercal) \leftarrow\!\$\, \mathcal{F} = \mathcal{B}^k \times (-\mathcal{B}^k)$
$f_\mathsf{R}^\intercal \leftarrow\!\$\, \chi_h = \mathcal{D}$
$\mathbf{r} = (\mathbf{s}_\mathsf{C}^\intercal, \mathbf{e}_\mathsf{C}^\intercal)^\intercal$
$z = \mathbf{H}\mathbf{r} + f_\mathsf{R}$
we define $\mathbf{A} \in \mathcal{R}_q^{k\times k}$ and
$\mathbf{t}_\mathsf{L} \in \mathcal{R}_q^k$ such that $\mathbf{B}^\intercal = (\mathbf{A} \mid \mathbf{t}_\mathsf{L})$
$\mathbf{t}_\mathsf{R} \leftarrow \mathbf{A}\mathbf{s}_\mathsf{R} + \mathbf{e}_\mathsf{R}$; $\mathsf{pk}_\mathsf{R} = \mathbf{t}_\mathsf{R}$, $\mathsf{sk}_\mathsf{R} = \mathbf{s}_\mathsf{R}$
$(\mathsf{state}, m_{\mathsf{L},0}, m_{\mathsf{L},1}, m_\mathsf{R}) \leftarrow$
$\mathcal{A}_1(\mathbf{A}, \mathsf{pk}_\mathsf{R}, (\mathsf{sk}_\mathsf{R}, \mathbf{e}_\mathsf{R}), \mathsf{pk}_\mathsf{L})$
$c_{\mathsf{L},b} \leftarrow \tilde{\mathsf{c}_\mathsf{L}} + \lceil q/2 \rfloor m_{\mathsf{L},b}$
$\mathsf{c}_\mathsf{R} \leftarrow \mathsf{c}_\mathsf{C}^\intercal \mathbf{s}_\mathsf{R} + z + \lceil q/2 \rfloor m_\mathsf{R}$
$b \leftarrow\!\$\, \{0,1\}$
$b' \leftarrow \mathcal{A}_2\,(\mathsf{state}, (\mathsf{c}_\mathsf{C}, c_{\mathsf{L},b}, \mathsf{c}_\mathsf{R}))$
return $b = b'$

$\mathsf{G}_3(\mathcal{A})$

One difference with G_2:
$(\mathsf{c}_\mathsf{C}^\intercal, \tilde{\mathsf{c}_\mathsf{L}})^\intercal \leftarrow\!\$\, \mathcal{R}_q^{(k+1)}$

Fig. 18: Complete games for the proof of Theorem 5.

Game G_0**.** The initial [IND-CPA, -] game.

Game G_1**.** In G_1, the vector $\mathbf{t}_\mathsf{L}$ is sampled uniformly at random, and the vectors $\mathbf{e}_\mathsf{L}$ and $\mathbf{s}_\mathsf{L}$ are no longer used. We can therefore construct an adversary $\mathcal{B}$ such that: $\left|\mathsf{Adv}^{\mathsf{G}_1}(\mathcal{A}) - \mathsf{Adv}^{[\mathsf{IND\text{-}CPA}, \text{-}]}(\mathcal{A})\right|$ is bounded by $\mathsf{Adv}^{\mathsf{MLWE}^{\mathcal{B}^{2k}}_{\mathcal{R},k,k,q}}(\mathcal{B})$.

Game G_2**.** In G_2, we equivalently recompute several elements from G_1 as follows. Let $\mathbf{A} \in \mathcal{R}^{k\times k}$ and $\mathbf{t}_\mathsf{L} \in \mathcal{R}^k$ be such that $\mathbf{B}^\intercal = (\mathbf{A} \mid \mathbf{t}_\mathsf{L})$ for a uniformly sampled matrix $\mathbf{B} \in \mathcal{R}^{(k+1)\times k}$.

This implies that c_C and $\tilde{\mathsf{c}_\mathsf{L}} = \mathbf{s}_\mathsf{C}^\intercal \mathbf{t}_\mathsf{L} + f_\mathsf{L}$ can be expressed as $(\mathsf{c}_\mathsf{C}^\intercal, \tilde{\mathsf{c}_\mathsf{L}})^\intercal = \mathbf{B}\mathbf{s}_\mathsf{C} + (\mathbf{e}_\mathsf{C}^\intercal, f_\mathsf{L})^\intercal$.

Moreover, the term $\mathbf{s}_\mathsf{C}^\intercal \mathbf{t}_\mathsf{R} + f_\mathsf{R}$ is replaced with the term $\mathsf{c}_\mathsf{C}^\intercal \mathbf{s}_\mathsf{R} + z$, where $z = (\mathbf{s}_\mathsf{C}^\intercal, \mathbf{e}_\mathsf{C}^\intercal)\begin{pmatrix}\mathbf{e}_\mathsf{R}\\ -\mathbf{s}_\mathsf{R}\end{pmatrix} + f_\mathsf{R}$. These two expressions are indeed equivalent, since:

$$\begin{aligned}\mathbf{s}_\mathsf{C}^\intercal \mathbf{t}_\mathsf{R} + f_\mathsf{R} = \mathbf{s}_\mathsf{C}^\intercal(\mathbf{A}\mathbf{s}_\mathsf{R} + \mathbf{e}_\mathsf{R}) + f_\mathsf{R} &= (\mathbf{s}_\mathsf{C}^\intercal \mathbf{A} + \mathbf{e}_\mathsf{C}^\intercal)\,\mathbf{s}_\mathsf{R} - \mathbf{e}_\mathsf{C}^\intercal \mathbf{s}_\mathsf{R} + \mathbf{s}_\mathsf{C}^\intercal \mathbf{e}_\mathsf{R} + f_\mathsf{R} \\ &= \mathsf{c}_\mathsf{C}^\intercal \mathbf{s}_\mathsf{R} + (\mathbf{s}_\mathsf{C}^\intercal, \mathbf{e}_\mathsf{C}^\intercal)\begin{pmatrix}\mathbf{e}_\mathsf{R}\\ -\mathbf{s}_\mathsf{R}\end{pmatrix} + f_\mathsf{R} = \mathsf{c}_\mathsf{C}^\intercal \mathbf{s}_\mathsf{R} + z.\end{aligned}$$

Thus, the advantage does not change.

Game G_3. This game is identical to G_2, except that $(\mathsf{c}_\mathsf{C}^\intercal, \tilde{\mathsf{c}_\mathsf{L}})^\intercal$ is now sampled uniformly at random. Consequently, $c_{\mathsf{L},b} = \tilde{\mathsf{c}_\mathsf{L}} + \mathsf{Encode}(m_{\mathsf{L},b})$ is indistinguishable from a uniformly random element for the adversary $\mathcal{A}$. Hence, $\mathsf{Adv}^{\mathsf{G}_3}(\mathcal{A}) = 0$.

The only difference between G_2 and G_3 lies in how $(\mathsf{c}_\mathsf{C}^\intercal, \tilde{\mathsf{c}_\mathsf{L}})^\intercal$ is generated: in G_2, it is computed as $\mathbf{Bs}_\mathsf{C} + (\mathbf{e}_\mathsf{C}^\intercal, f_\mathsf{L})^\intercal$, whereas in G_3 it is drawn uniformly from $\mathcal{R}_q^{(k+1)}$. This change directly corresponds to the distributional shift underlying the Hint-MLWE problem with the parameters specified in the theorem statement, and we may therefore conclude.

H.3 Further Parameter Selection and Heuristic Discussion for Maul

Various trade-offs arise when selecting parameters, primarily governed by one consideration: decryption failure rate. To ensure proper security, we emphasize a conservative parameter-selection approach. Specifically, parameters are chosen such that the failure rate remains strictly below the defined security threshold. Nevertheless, accurately quantifying the security degradation associated with these attacks remains a tough question. Recent studies, notably [19], have begun to address this challenge.

In this section, we present several parameter sets to accommodate different security requirements and performance trade-offs. Beyond simply tuning the decryption failure rate, accounting for or disregarding the previously outlined heuristics can yield substantially different parameter choices. Accordingly, we propose three parameter sets: one that reuses the ML-KEM parameters to preserve key and implementation compatibility—though with a higher decryption failure rate at NIST levels 1 and 5, while remaining sufficiently low at level 3—and two more conservative sets that do not rely on Heuristic 1.

Maul-Kyber. The main drawback of the conservative approach lies in its incompatibility with ML-KEM/Kyber implementations. To complement the core parameter sets, Table 7 also includes a set aligned with ML-KEM's parameters. However, this set exhibits a higher decryption failure rate for at NIST levels 1 and 5, potentially resulting in effective security below the target levels, while remaining sufficiently low for Maul768-K. The latter therefore represents a practical choice when implementation compatibility with ML-KEM-768 is desired. As for Maul512-K and Maul1024-K, following the analysis of [19], we emphasize that such parameter choices may be risky, particularly in our setting where one of the two keys is static, since even moderate failure probabilities could be exploitable.

Table 7: Parameter sets for Maul-K (Kyber-like), including sizes (in bytes).

	bikz	n	k	q	$\mathcal{B}$	$\mathcal{D}$	(d_u, d_v)	δ	ν	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{ct}\rvert$	$\lvert\mathsf{pk}\rvert + \lvert\mathsf{ct}\rvert$
Maul512-K	371	256	2	3329	$\mathcal{B}_3$	$\mathcal{D}_{48}$	$(-, 7)$	2^{-101}	384	749	1196	1945
Maul768-K	591	256	3	3329	$\mathcal{B}_2$	$\mathcal{D}_{32}$	$(-, 7)$	2^{-184}	512	1124	1572	2696
Maul1024-K	829	256	4	3329	$\mathcal{B}_2$	$\mathcal{D}_{32}$	$(-, 7)$	2^{-153}	640	1498	1946	3444

Heuristic Discussions on Discrete Gaussians Convolution. Continuous Gaussian distributions satisfy a crucial property: preservation under convolution. Namely, $\mathcal{D}_\sigma + \mathcal{D}_\sigma = \mathcal{D}_{\sigma\sqrt{2}}$ for a given standard deviation σ. However, this property does not hold for discrete Gaussians, although accurate approximations can be obtained, particularly when considering parameters above the smoothing parameter, as detailed in [44, Lemma 2.4]. In our case, the standard deviations reported in Tables 5 and 7 lie below the smoothing parameter.

To better quantify the deviation, we developed a script to compute the Rényi divergence between $\mathcal{D}_{\mathbb{Z},\sigma} + \mathcal{D}_{\mathbb{Z},\sigma}$ and $\mathcal{D}_{\mathbb{Z},\sigma\sqrt{2}}$, allowing us to assess the security loss and propose more conservative parameters accordingly.

Definition 4 (Rényi divergence [45]). *Let $\mathcal{P}$ and $\mathcal{Q}$ be two distributions such that $\mathsf{Supp}(\mathcal{P}) \subset \mathsf{Supp}(\mathcal{Q})$. For $a \in (1, \infty)$, we define the Rényi divergence of order a by:*

$$R_a(\mathcal{P}||\mathcal{Q}) = \left(\sum_{x \in \mathsf{Supp}(\mathcal{P})} \frac{\mathcal{P}(x)^a}{\mathcal{Q}(x)^{a-1}} \right)^{\frac{1}{a-1}}.$$

In addition, we define the Rényi divergence of order ∞ by :

$$R_\infty(\mathcal{P}||\mathcal{Q}) = \max_{x \in \mathsf{Supp}(\mathcal{P})} \frac{\mathcal{P}(x)}{\mathcal{Q}(x)}.$$

Using the Rényi divergence, we quantify the security loss introduced by the use of Hint-MLWE with the distribution $D_{\mathbb{Z},\sigma\sqrt{2}}$. Specifically, given oracle access to the distributions $D_{\mathbb{Z},\sigma} + D_{\mathbb{Z},\sigma}$ and $D_{\mathbb{Z},\sigma\sqrt{2}}$, an adversary $\mathcal{A}$ making q_s queries can distinguish these distributions with an advantage at most $2^{-\lambda}$, provided the condition $R_a\left(D_{\mathbb{Z},\sigma} + D_{\mathbb{Z},\sigma}||D_{\mathbb{Z},\sigma\sqrt{2}}\right) \leq 1 + \frac{1}{4q_s}$ holds true for $a = 2(\lambda+1)$ [45, Equation 4].

In our scheme, querying these distributions equates to executing Maul; hence, we define $q_s = 2^{64}$ as the maximum number of queries to Maul. Additionally, we replace centered binomial distributions with discrete Gaussian distributions according to Heuristic 1. To establish an accurate equivalence, we set $D_{\mathbb{Z},\sigma} \sim \mathcal{B}_\eta$ with variance $\sigma^2 = \frac{\eta}{2}$.

Table 8: Parameter sets for Maul-C (conservative standard deviation), including sizes (in bytes).

	bikz	n	k	q	$\mathcal{B}$	$\mathcal{D}$	(d_u, d_v)	δ	ν	|pk|	|ct|	|pk| + |ct|
Maul512-C	347	256	2	7681	$\mathcal{B}_6$	$\mathcal{D}_{96}$	$(12,7)$	2^{-129}	384	826	1216	2042
Maul768-C	548	256	3	10753	$\mathcal{B}_6$	$\mathcal{D}_{96}$	$(12,6)$	2^{-193}	512	1286	1536	2822
Maul1024-C	755	256	4	13313	$\mathcal{B}_6$	$\mathcal{D}_{96}$	$(13,7)$	2^{-258}	640	1754	2112	3866

The resulting Rényi divergence profile, based on these distributions, is illustrated in Figure 19. This profile allows to determine an optimal σ for targeted security levels (128, 192, or 256 bits). Given that queries directly invoke Maul instead of a generic oracle, we maintain $q_s = 2^{64}$ across all security targets. Utilizing Figure 19, the required Rényi divergence bound of $\frac{1}{4q_s} = 2^{-66}$ is satisfied for $\sigma = 1.7$, resulting in the corresponding binomial parameter $\eta = 6$.

Table 9: Parameter sets for Maul-R (Rényi-based), including sizes (in bytes).

	bikz	n	k	q	$\mathcal{B}$	$\mathcal{D}$	(d_u, d_v)	δ	ν	|pk|	|ct|	|pk| + |ct|
Maul512-R	330	256	2	15361	$\mathcal{B}_{12}$	$\mathcal{D}_{192}$	$(12,6)$	2^{-128}	384	890	1280	2170
Maul768-R	523	256	3	22273	$\mathcal{B}_{12}$	$\mathcal{D}_{192}$	$(12,6)$	2^{-194}	512	1387	1536	2923
Maul1024-R	728	256	4	26113	$\mathcal{B}_{12}$	$\mathcal{D}_{192}$	$(14,8)$	2^{-256}	640	1879	2304	4183

In Table 8, we highlight a set of parameters following previous Rényi analysis ensuring low decryption failure rate. Meanwhile in Table 9, we present a fully conservative set of parameters taking into account an additional factor of $\sqrt{2}$ from Hint-MLWE reduction to MLWE leading to $\eta = 12$. Note that these different conservative choices degrade the estimated security measured through bikz size.

I Additional Uses Cases for DUAKE and Maul

As previously noted, the DUAKE_2 protocol offers unilateral implicit authentication, which can be straightforwardly upgraded to explicit authentication by appending a key confirmation round, as in DUAKE_3. This type of UAKE protocol is particularly well-suited to two prominent use cases: KEM-based TLS handshakes (see Figure 21) and authenticated continuous key agreement.

Authenticated Continuous Key Agreement. A Continuous Key Agreement (CKA) is a secure messaging subprotocol, initially introduced in [2], that enables secure messaging protocols to achieve

[47] https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/call-for-proposals

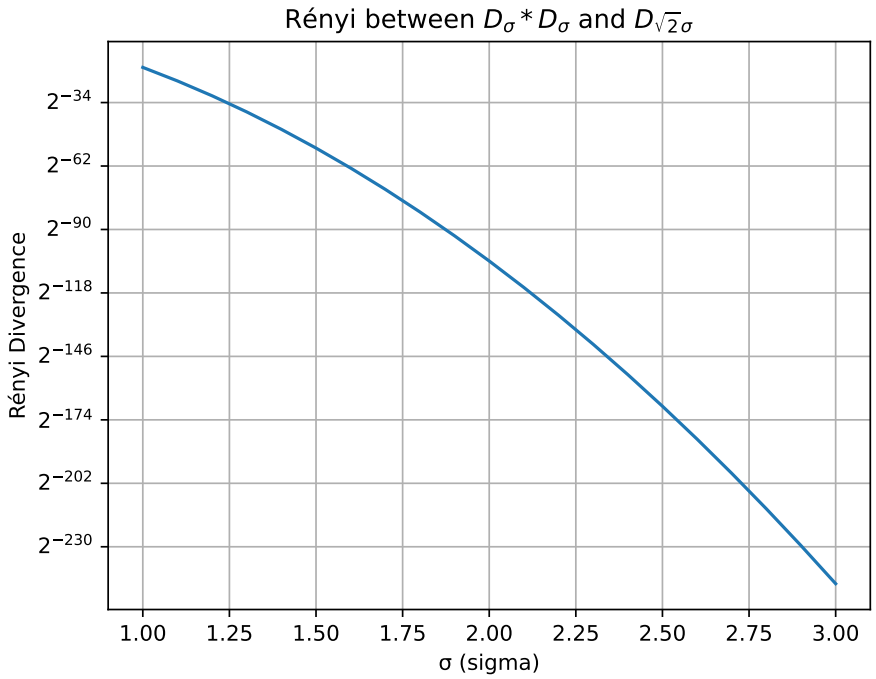

Fig. 19: Rényi divergence between $D_{\mathbb{Z},\sigma} + D_{\mathbb{Z},\sigma}$ and $D_{\mathbb{Z},\sigma\sqrt{2}}$ at increasing standard deviation σ (substracted by 1).

both forward secrecy and post-compromise security.[48] Standard secure messaging protocols typically begin with a (non-interactive) authenticated key exchange (AKE) to establish an initial shared secret known as the root key. This root key is then regularly updated using the CKA subroutine to provide continued security.

However, as emphasized by [22], an adversary who compromises one party may not remain passive but can actively launch a man-in-the-middle (MitM) attack by conducting its own key exchanges with both parties. Given that secure messaging sessions may last for years, such attacks can have enduring consequences. A natural countermeasure is to periodically execute an *authenticated* key exchange instead of relying solely on unauthenticated updates, leading to what [22] defines as an *Authenticated Continuous Key Agreement (ACKA)*.[49]

In parallel, post-quantum (PQ) hybridization has gained traction in secure messaging protocols, with several efforts incorporating PQ ratchets alongside classical Diffie-Hellman key exchanges. For instance, Apple's iMessage implements the PQ3 protocol [4], which hybridizes classical and PQ cryptography not only during the initial AKE but also within the CKA phase, introducing one PQ ratchet approximately every 50 classical ratchets [34]. More recently, [21] introduced the Triple-Ratchet protocol, an efficient hybrid CKA based on a novel *ratcheted-KEM*[50] called Katana, designed for integration into future versions of the Signal protocol.

A post-quantum ACKA requires periodic KEM-based implicit authentications[51] alongside ephemeral PQ ratchets. Each such authentication would traditionally incur the transmission of an additional ciphertext. However, our UAKE construction DUAKE_2, instantiated with the 2K-KEM scheme Maul, is particularly well-suited in this setting, as it reduces the communication cost of unilateral authentication from two ciphertexts to one, significantly improving communication efficiency, as shown in Table 3.

[48] Forward secrecy ensures that past communications remain secure even if long-term secrets are compromised. Post-compromise security refers to the ability of the protocol to recover and regain security after such a compromise.

[49] Note that the ACKA approach in [22] differs from ours, as it provides continuous authentication using a distinct mechanism rather than periodic AKEs.

[50] A ratcheted-KEM is a KEM introduced in [21] for PQ CKAs, where the ciphertext of one encapsulation can serve as a public key in subsequent encapsulations.

[51] We exclude PQ signature-based authentication due to its high communication overhead and computational cost compared to KEM-based methods.

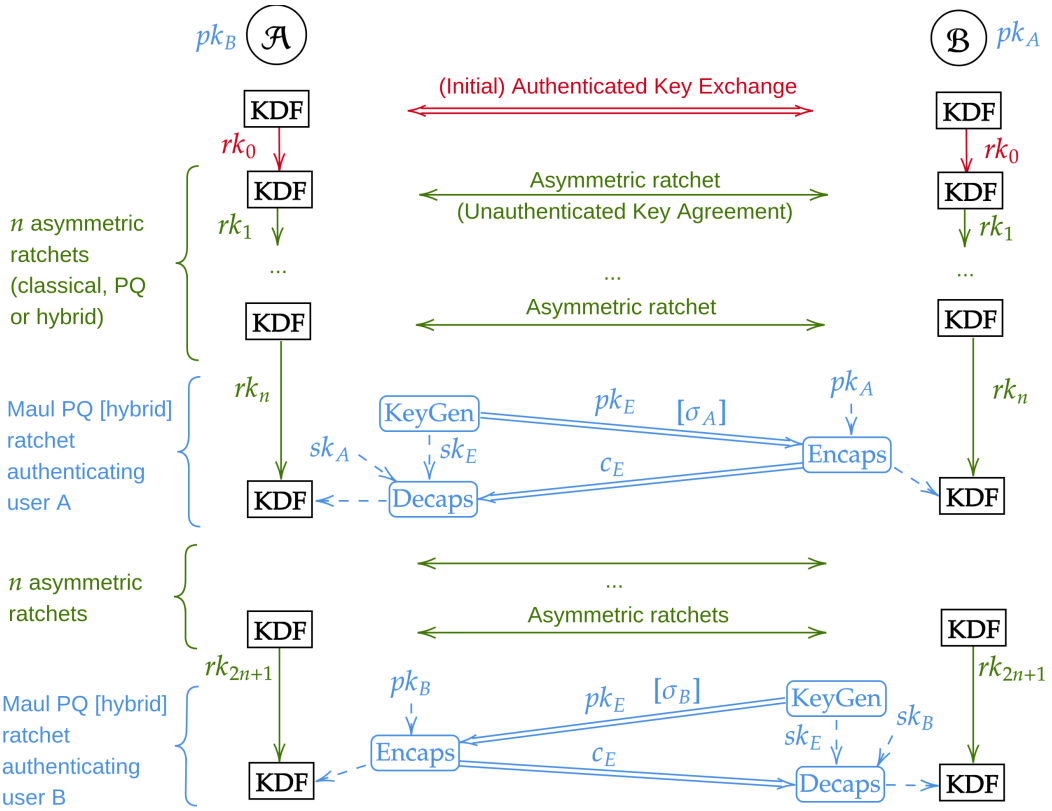

Fig. 20: Authenticated Continuous Key Agreement (ACKA) using our DUAKE (in blue) for periodic unilateral implicit authentication. In green are unauthenticated ratchets performed between authenticated ones, whose number is protocol-dependent. The variable rk_i denotes the i-th generation of the shared secret.

Moreover, full mutual authentication need not occur simultaneously. Instead, it can be distributed over successive ratchets, further reducing communication overhead. Specifically, the encapsulating party uses Maul with the recipient's ephemeral public key (previously received) and static authentication public key to produce a single ciphertext. The recipient decrypts this ciphertext, gaining a shared key and implicitly confirming the sender's identity. This authentication is made explicit upon the sender's first encrypted message using the derived key. Later, the recipient can reciprocate using the same mechanism, completing the mutual authentication in a staggered fashion.

Figure 20 illustrates this interplay between authenticated and unauthenticated PQ (or hybrid) ratchets within an ACKA protocol. We underline that the PQ authentication ratchet implemented with Maul car easily be hybridized, if needed, with a classical authentication mechanism such as a signature scheme.

Maul-Based TLS Handshake. Unilaterally authenticated key exchanges are widely employed in TLS handshakes, where typically only the server is authenticated. In light of the well-known drawbacks of post-quantum signature schemes, several works [49,50] have proposed a KEM-based variant of TLS, known as KEMTLS, which is currently under study by the IETF [56]. This mechanism combines two KEMs in parallel and would thus benefit significantly from the use of our Maul double-key KEM, which achieves the same functionality with reduced communication overhead.

However, adapting KEMTLS to benefit from Maul would require protocol changes: specifically, enabling the client to perform the double encapsulation, in contrast to the current version where the server performs one encapsulation followed by a client encapsulation.

Figure 21, shows the message flow of a TLS handshake using our Maul 2K-KEM.

The reversed order of exchanges introduced by our scheme, compared with KEMTLS, has notable implications. Indeed, in the regular TLS 1.3 handshake [46], as well as in KEMTLS, the `ServerHello` message is immediately followed by `EncryptedExtensions` – which is encrypted – and by the server's certificate. This early encryption—though only authenticated later—enhances the privacy guarantees of the handshake.

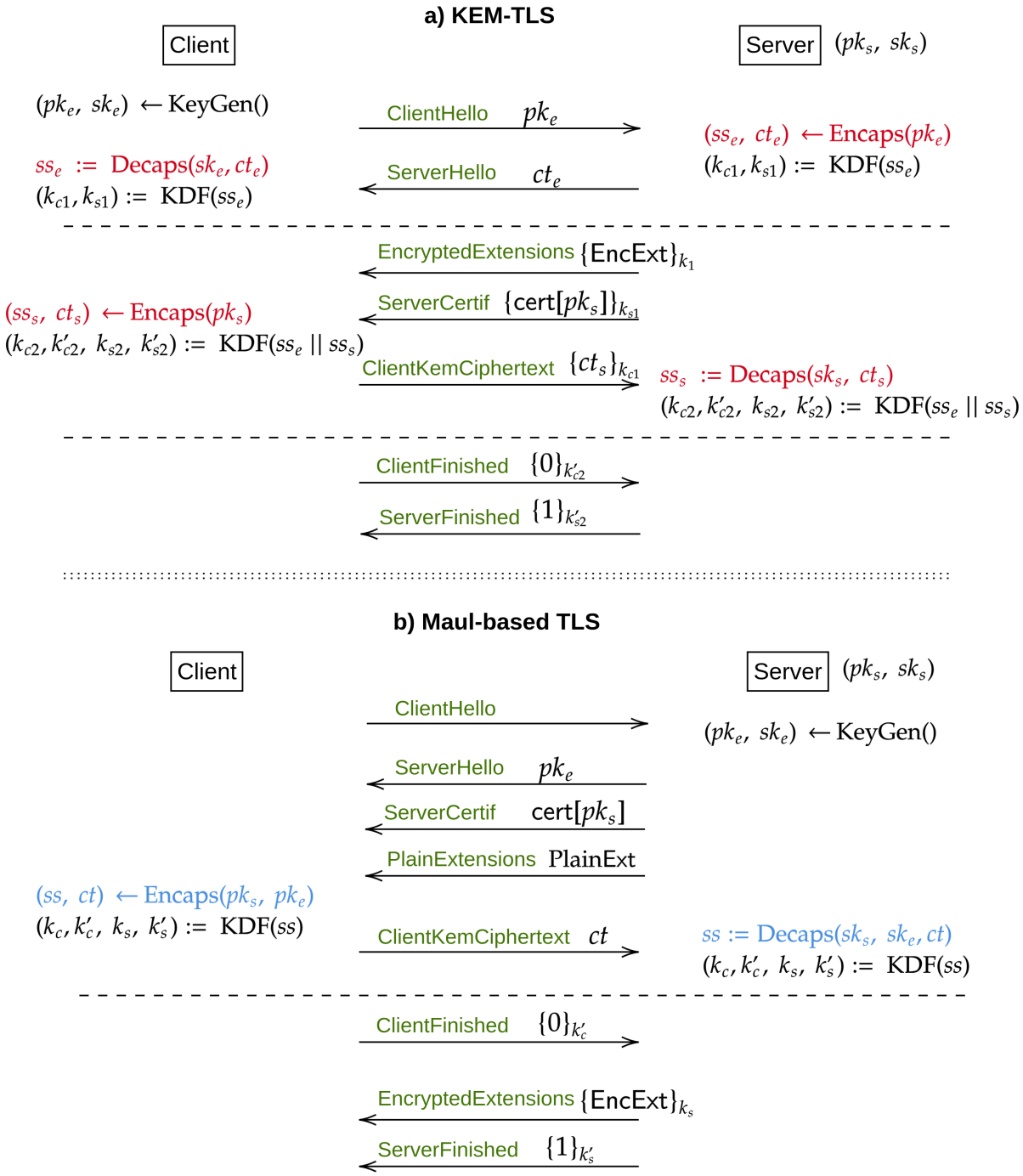

Fig. 21: Compared TLS handshakes (server unilateral-authentication) with KEMTLS (top) and with our Maul double-key KEM (bottom).

In contrast, with Maul's double encapsulation, the key exchange and server authentication occur simultaneously, which precludes this interim unauthenticated encryption. As a consequence:

- The server's certificate must be transmitted in the clear;
- The `EncryptedExtensions` message must either be sent in the clear as well, or deferred until the subsequent `ServerFinished` message, once the server has decrypted the client's ciphertext.

Table of Contents

DAKE: Bandwidth-Efficient (U)AKE from Double-KEM 1
Hugo Beguinet, Céline Chevalier, Guirec Lebrun, Thomas Legavre, Thomas Ricosset, Maxime Roméas, and Éric Sageloli

1 Introduction 1
1.1 Our Contributions 2
1.2 Technical Overview 3
1.3 Related Works 7
2 Preliminaries 8
3 Authenticated Key Exchanges from Double-KEMs 10
3.1 The eCK Security Model and Its Variants 10
3.2 Our AKE Protocols 12
3.3 Security of Our AKE Protocols 12
4 Chosen-key Fujisaki–Okamoto Transform 15
4.1 Combining Two PKE to Obtain a 2K2M-PKE 15
4.2 The CK-FO Transform and Its Security 16
5 Compact Lattice-based Double-KEM 17
5.1 Preliminaries on Lattices, MLWE, and Hint-MLWE Problems 17
5.2 $\mathsf{Maul}_{\mathsf{PKE}}$: A [IND-CPA, IND-CPA]-Secure 2K2M-PKE 18
5.3 Parameter Selection 19
5.4 Security Estimation 20
5.5 Concrete Security Loss 20
A Additional Preliminaries 25
B Proof of the CK-FO Transform 28
B.1 Proof in the Random Oracle Model 29
B.2 Proof in the Quantum-Accessible Random Oracle Model 33
C The eCK Model and its Variants 38
C.1 Modelization of Protocol, Participants and Adversarial Queries 39
C.2 Correctness, Key Secrecy and Freshness predicates 41
C.3 Definition of $\mathsf{eCK}^{\mathsf{w}}$ and eCK-PFS Models 42
C.4 Unilateral Versions of $\mathsf{eCK}^{\mathsf{w}}$ ($\mathsf{u\text{-}eCK}^{\mathsf{w}}$) and eCK-PFS (u-eCK-PFS) 43
D Detailed Presentation of DAKE^{σ} and DUAKE 44
D.1 DAKE^{σ}: A Variant of DAKE Using a Signature 44
D.2 DUAKE: An Unilateral AKE Fitted for Maul 45
E Key Secrecy of DAKE, DAKE^{σ}, and DUAKE 46
E.1 Context, Rationale of the Proof and Notations 47
E.2 Notations Used for the Proofs 48
E.3 Key Secrecy for Predicates $\mathsf{U} \cap \mathsf{eCK}^{\mathsf{w}}$ and $\mathsf{U} \cap$ eCK-PFS 48
E.4 Key Secrecy for Predicates $\mathsf{V}_1, \cdots, \mathsf{V}_5$ 52
E.5 Key Secrecy of DUAKE 60
F Explicit Authentication of DAKE and its Variants 60
G Instantiating the Xue et al. AKE with One-Sided Chosen-Key Secure Double-KEMs 61
H Additional Details Regarding Hint-MLWE and $\mathsf{Maul}_{\mathsf{PKE}}$ 62
H.1 Hint-MLWE Hardness 62
H.2 Complete Correctness and Security Proofs for $\mathsf{Maul}_{\mathsf{PKE}}$ 64
H.3 Further Parameter Selection and Heuristic Discussion for Maul 66
I Additional Uses Cases for DUAKE and Maul 67