AKE Protocol Combining PQC and QKD*

Loïc Ferreira

Orange Research, Applied Crypto Group, Caen, France
loic.ferreira@orange.com

Abstract. With the advent of quantum computing, which threatens the very foundations of classical cryptography, several authenticated key exchange (AKE) protocols have been proposed, combining classical and post-quantum cryptographic algorithms, and a quantum key distribution (QKD) sub-protocol. The goal being to associate the claimed information theoretic security of QKD, and the security based upon computational assumptions of classical and post-quantum cryptography. To our knowledge, in existing security proofs of such hybrid AKE protocols, the QKD phase is treated as a black box and the impact of establishing the output quantum key appears similar as setting a symmetric key beforehand at the two communicating parties. In this paper, we describe an hybrid AKE protocol and a security model that captures multiple security properties. Our security analysis integrates the security loss induced by the QKD phase as well as that of implied by the classical and post-quantum cryptographic algorithms involved in the protocol.

Keywords: Authenticated key exchange · Post-quantum cryptography · Quantum key distribution · Security model

1 Introduction

1.1 Context

Establishing a new secret key between two parties that communicate through a public channel is one of the most fundamental goal in cryptography. Numerous authenticated key exchange (AKE) protocols have been devised since the beginning of modern cryptography such as TLS 1.3 [45, 46, 99], IKEv2 [42, 75], QUIC [52, 71, 82], to mention only a few, widely used on internet. The security of these protocols (based in particular on asymmetric algorithms) rely upon mathematical assumptions. That is, the protocols ensure their claimed security properties as long as no algorithm efficient enough to solve the underlying mathematical problem, and no adapted computation power to execute the attack algorithm exist. This paradigm of computational hardness assumption and the "companion" framework of provable security enable rigorous logical reasoning to capture and characterise what makes a protocol secure. These hard

* This is the full and corrected version (see Section B) of the paper published by the IACR journal Communications in Cryptology (vol. 2, no. 4), licensed under a "CC BY 4.0" license.

mathematical problems are thoroughly analysed over long time periods. This enables uncovering weaknesses and discarding unsafe proposals, as illustrated again during the NIST PQC (post-quantum cryptography) process[1] with the cryptanalysis of the SIKE [35, 83, 101] and Rainbow [22] algorithms. However, no guarantee can be given regarding the (im)possibility to break, in the future, algorithms and protocols relying upon longstanding mathematical problems.

In 1984, Bennett and Brassard [19] take a radically different approach and describe a key exchange protocol (now known as BB84) based on quantum physics. This scheme as well as other proposals based on a similar approach (e.g. [18, 20, 50, 103]), coined as Quantum Key Distribution (QKD), aim at providing a security level beyond what (non-quantum) cryptography may offer, such as "unconditional security" (against a computationally unbounded adversary) or "everlasting security" (against an adversary computationally restricted during a preliminary epoch but not after some point in time [108]). Despite its great expectations, the QKD technology has not been as successful as expected in terms of impact and deployment [2, Chap. 5] (at least compared to public key cryptography). The promise of "absolute" security remains unmet so far, as illustrated by practical attacks (beyond denial of service) against commercially available QKD equipment [85], and this technology still suffers from practical limitations (e.g. cost, range, use-cases) [3, 4, 53, 91, 93].

A revised positioning tends to envisage the QKD technology as a complementary approach together with post-quantum cryptography.[2] Instead of considering security solely guaranteed by the "*laws of physics*" [34, 37, 106], this approach explores more realistic security models combined to additional assumptions with the goal to design protocols that can be analysed in the realm of provable security paradigm [2, Chap. 5]. In this paper, we aim at taking a similar path.

1.2 Related Work

Previous Protocols. Mosca, Stebila, Ustaoğlu [87, 88] describe an AKE protocol combining the QKD protocol BB84 [19] with a signature algorithm to authenticate the classical channel. The AKE protocol does not combine in its key exchange phase (classical or post-quantum) cryptographic algorithms with QKD. The security proof relies upon that of done by Christandl, Renner, Ekert [37] of the BB84 protocol. Alternative proposals of PQC-authenticated QKD exchanges exist (e.g. [113,116]) but they are not backed with a security analysis.

In a series of successive works, Dowling, Hansen, Paterson [48, 49], Bruckner, Ramacher, Striecks [31], Garms, Paraïso, Hanley, Khalid, Rafferty, Grant, Newman, Shields, Cid, O'Neill [57, 58], Battarbee, Striecks, Perret, Ramacher, Verhaeghe [9], and Buruaga, Bugler, Brito, Martin, Striecks [33] describe hybrid

[1] `https://csrc.nist.gov/projects/post-quantum-cryptography`

[2] Such a complementarity is however limited. Without public-key cryptography, our current digital world would not exist, and post-quantum cryptography aims at eventually replacing (wherever needed) classical cryptography which is ubiquitous in the digital services. QKD is restricted to niche use-cases.

AKE protocols combining QKD with classical and post-quantum cryptographic algorithms, titled Muckle, Muckle+, Muckle++, Muckle# and VMuckle respectively. Muckle relies upon pre-shared symmetric keys (PSK) and a MAC function to provide authentication. But, PSK-based AKE protocols do not scale well since $\mathcal{O}(n^2)$ keys must be distributed in a network involving n parties. Therefore, Muckle+ bears on a signature algorithm instead and Muckle++ can use either a MAC or a signature algorithm. Leveraging the currently greater efficiency on average of post-quantum key encapsulation mechanisms (KEM) compared to post-quantum signature schemes, in Muckle# authentication is ensured with static post-quantum KEM public keys (following previous works [78, 105]). VMuckle is an adaptation of Muckle+ to MACsec [70], supporting authentication with a preshared symmetric key or a signature.

The protocols described in [9, 31, 33, 48, 49, 57, 58] involve exchanges done solely on the classical channel, based on computations implying only classical and post-quantum algorithms. That is, the QKD phase is treated as a black box (as acknowledged in [9,48,49]) and appears merely in the form of its final output: a string of shared pseudo-random bits that is eventually used to e.g. key a PRF function. This is reflected in the different security proofs where the contribution of the QKD phase weighs the same, in terms of security loss, as the presence of a PSK at the two communicating parties. That is, if one neglects the mentioned origin of this secret parameter (output of a QKD run), then the reasoning and bounds in the security proofs given in [9, 31, 33, 48, 49, 57, 58] would be the same as a PSK-based protocol without QKD phase.

Hövelmanns, Planken, Schaffner, Verschoor [65] propose an hybrid AKE protocol combining QKD with classical and post-quantum cryptographic algorithms and its security model along with detailed security proofs. One difference with previous proposals (in particular Muckle and successive works) is that the adversary is allowed not only to learn (through reveal queries) but also to modify QKD keys. However, in the proposed AKE protocol and its analysis, the QKD phase is also treated as a symmetric key delivered to the communicating parties.

In addition, all these previous works assume that QKD keys are always available during a protocol run and do not take into account the opposite possibility which may be due to unintentional disturbances of the infrastructure, a too low key rate or a denial-of-service attack.

Security Models. The security model adopted by Mosca et al. [87, 88] focuses on the indistinguishability of the session key and does not incorporate other security properties (such as explicit party authentication, identity privacy), even though their "fresh session" definition precludes trivial misbinding identity attacks (i.e. replacement of the signatures computed by an honest party with signatures computed by the adversary with a revealed private key). Their main security property relates to an "*authenticated key exchange protocol*" but the corresponding definition [88, Def. 3] describes the ability for an adversary to distinguish between the "real" session key and a uniformly random key. Mosca et al. define also correctness but no proof of the latter is given. The security

proof relies upon that of done by Christandl, Renner, Ekert [37] of the BB84 protocol, and concludes that the overall security loss is negligible. The proof by Christandl et al. proves security asymptotically only. We observe that, in Mosca et al.'s AKE protocol, the uniqueness of the identifiers involved in each run conveniently avoids an additional security loss due to a possible collision of two identifiers if they were pseudo-random. Likely, this uniqueness implies that the protocol be stateful (e.g. keeping track of used identifers, using a monotonically counter as identifier) which incurs a specific burden compared to stateless key exchange protocols.

The security models used in [9, 31, 33, 48, 49, 57, 58] are similar (based on the HAKE framework presented in [48, 49]) and all focus on the indistinguishability of the session key. Soundness, explicit authentication and identity privacy are not formally defined. Intuitively, one might assume that authentication is related to the notion of "matching sessions" [48, Def. 1], and this notion is indeed leveraged in the proof (for key indistinguishability) of the described protocol. However, the existence of the notion of "matching sessions" in the paper does not necessarily imply that explicit authentication (if formally defined) would rely upon the latter (it could be based on a different notion such as the equality of the "original session key" as in [79]). Regarding the protocol, the intuition tells us that the security of the MAC function and the secrecy of the preshared key (and other parameters) forbid the adversary from impersonating an honest user. But, again, what authentication means in this context is not defined, and the paper does not explicit what are the key cryptographic components needed to ensure the authentication property.

The same holds for the subsequent works [9, 31, 33, 57, 58] where security is defined as key indistinguishability ([31, Def. 10], [58, Def. 13], [9, Def. 9], [33, Def. 10]). Again, soundness, explicit authentication and identity privacy are not defined, and the papers do not explain which cryptographic components and which underlying properties enable ensuring these aforementioned properties. In [9, 33] several protocol messages are encrypted (in particular the certificates and signatures), but no claim is made (nor proven) regarding identity privacy.

The security model described in [65] defines two main notions: correctness and key indistinguishability and detailed proofs are given. But no additional security properties are proposed.

Overall, with the previous works, the readers are left to their own reasoning regarding security properties that are not explicitly stated. They must dive into the security proofs to understand the extent of the additional security properties that could be ensured, besides key indistinguishability, and to assess the corresponding bounds.[3]

In this work, we adopt a strong security model that captures distinct security

[3] Note that key indistinguishability is trivially broken in [31], as explained in [65]. But this comes from the fact that the session key is supposed to be the parameter MS. The latter is involved during the protocol run to compute a parameter that can be revealed to the adversary (hence, distinguishability is straightforward). Defining $CATS$, $SATS$ (computed from MS and not used during the AKE run) as the session

properties (soundness, explicit authentication, key indistinguishability, identity privacy), and we provide the corresponding security proofs and bounds.

Other Protocols. Other proposals consist of hybrid AKE protocols combining classical, post-quantum algorithms and QKD exchanges [5, 59, 86, 96, 100, 120], some of them being a variant of TLS 1.2, TLS 1.3 or WireGuard integrating PQC and QKD [25, 32, 36, 56, 102]. These proposals are sometimes illustrated and evaluated with practical experiments. However, they lack a security analysis and the security properties guaranteed by the proposed protocol are somewhat unclear.

1.3 Contributions

Our contributions are the following ones:

- We devise a security model with specific security properties (as opposed to "all-in-one" models) to analyse an hybrid AKE protocol and make explicit the potential impact of the QKD phase (in addition to that of the (non-quantum) cryptographic schemes) regarding each of these properties. Existing analyses of hybrid AKE protocols focus mainly on the indistinguishability of the session key (see Section 1.2).
- We design an hybrid AKE protocol that combines different cryptographic techniques (in particular ephemeral and static asymmetric parameters, internal secret state) to enlighten the involvement of the QKD phase in the computation of the different (intermediary and final) protocol parameters during a run.
- We provide the full proofs for each security property captured by our security model and make clear the security loss implied by the QKD phase. The corresponding security bounds are qualitative rather than quantitative. Moreover, our work is opposite to previous ones where the quantum phase appears (in the proofs) merely as a symmetric key shared beforehand by the two communicating parties. In addition, our analysis is done in the standard model with (polynomial-time) classical or quantum (i.e. with access to a local source of quantum computing) adversaries.

Our analysis shows that, considering experimental deployments or commercially available equipment, the security loss induced by the QKD phase of the proposed hybrid AKE protocol is higher (although it can theoretically be negligible) than that of due to the (classical or post-quantum) cryptographic components of the protocol, and does not allow getting meaningful security bounds. Decreasing the QKD bound seems dependent on improvements of the (quantum) protocol implemented or the equipment deployed in practical QKD networks, or on a reassessment of QKD security analyses.

keys solves the issue. We observe also that, although MS is the key to be guessed by the adversary, the proofs aim at showing the indistinguishability of $CATS$, $SATS$.

When designing a two-party hybrid AKE protocol, the security loss induced by the QKD phase should be integrated in the analysis (as acknowledged in [9, 48, 49]) as well as that of implied by the classical and post-quantum cryptographic algorithms involved in the protocol. This work does not focus on nor investigate the security of QKD technologies but, based on existing security proofs of QKD protocols, aims at proposing a more comprehensive way to analyse hybrid AKE protocols.

2 Protocol Description

In this section, we describe the proposed AKE protocol (Figures 1 and 2). The function Run (Figure 3) corresponds to the specific AKE protocol presented in this section, and is used in some oracles of the generic security model presented in Section 3.

2.1 Overview

Overall Description. A stage run between two instances is made of four messages sent through the classical channel and a set of quantum states sent through the quantum channel. The quantum states are sent by the initiator's instance to the responder's instance. Regarding the classical messages:

- The first message (m_1), sent by the initiator's instance, transports the ephemeral KEM public key. The KEM scheme provides post-quantum security for the key exchange and can be an hybrid scheme made of classical and post-quantum algorithms.
- The second message (m_2) transports the ephemeral KEM ciphertext, a MAC tag and data (related to the QKD phase) protected with an authenticated encryption (AE) function. The AE key is computed from the ephemeral KEM secret and an internal state pss that is output during the previous stage executed between the two same instances. The MAC tag protects the KEM ciphertext and is keyed with a key computed from pss. The parameter pss is reminiscent of Muckle [48,49] and subsequent variants, as well as older protocols like ZRTP [121] (where the shared secret internal state ensures a form of "key continuity" among the successive runs). Note that, thanks to the authenticity ensured by the AE scheme and the share secret state pss, the AE data authenticate the responder's instance (but during the first stage if pss is initialised with a public value).
- The third message (m_3) carries the initiator's signature and authenticates the initiator's instance to the responder's instance. The initiator's identity (as well as data related to the QKD phase) is made opaque with an AE scheme.
- The fourth message (m_4) carries the responder's signature and authenticates the responder's instance to the initiator's instance. The AE scheme protects the responder's identity as well as a flag $flag_{\mathsf{pe}}$ that indicates whether the

QKD phase has been successful (and whether the ephemeral quantum secret must be involved in the computation of the session keys and the updated pss). The responder's signature protects also the parameter $flag_{\mathsf{pe}}$.

The data transported in messages m_2, m_3 and m_4 are encrypted with an AE scheme. The successive symmetric keys depend on the ephemeral KEM secret and the internal state pss, as well as a (distinct) transcript of the messages exchanged so far. These transcripts are also integrity protected by the AE scheme. This allows the receiver's instance to check, at each step of the run, that its vision of the session is the same as that of the sender's instance.

Remark 1. The AE parameters (key, nonce) are computed with two distinct calls to function Expand (see Figure 1) instead of a single call to output both AE parameters. Let us consider for instance the parameters hk_R, n_R (used to output the AE ciphertext c_R). Computing hk_R with a distinct call to Expand enables to ensure that if one of the inputs to H, Extract or Expand and necessary to output hk_R are not shared by two (partnered) instances, then hk_R is not shared either by the two instances (assuming collision resistance for H, Extract and Expand). On the contrary, if hk_R and n_R are output by a single call to Expand, then, under the same assumptions (collision resistance), the fact that $hk_R \| n_R$ be not shared by the two instances does not imply that hk_R alone is not shared.

This allows relying upon the KEY-CMT property of the AE function (see Definition 20 in Section 4.3) instead of a stronger key committing property depending on both hk_R and n_R.

Moreover, the AE nonces are computed with Expand instead of being locally generated with a pseudo-random generator to dispense from sending these values to the peer instance.

QKD Phase. The messages exchanged during the QKD phase through the classical channel correspond to the description of the "prepare and measure" protocol devised by Bennett and Brassard [19] (BB84)[4] as described by Tomamichel and Leverrier [109] (denoted $\texttt{qkd_pm}_{pe,ec,pa}$ in their paper). In Figure 1, the steps involving qubit manipulations (e.g. encoding/decoding, transmission) are surrounded with dashed boxes while the whole QKD phase is highlighted in blue.

1. The initiator's instance chooses uniformly at random (i) a set of M sending bases Φ_I, each among two possibilities $\{+, \times\}$, and (ii) a set of M bits D. Each bit is encoded (function denoted by Γ) into a quantum state according to the selected basis, which yields a set of quantum states ρ_I that are sent through the quantum channel to the responder's instance.
2. The responder's instance chooses uniformly at random a set of M receiving bases Φ_R to measure (i.e. "decode", with a function denoted by Υ) the quantum states $\rho_R = \mathcal{Q}(\rho_I)$, where $\mathcal{Q}$ denotes the possible alteration and

[4] BB84 is the most well-known and widely used QKD protocol, due to its expected security guarantees, simplicity, practical implementation, and best overall performance [85, 117, 118].

loss due to the imperfection of the quantum channel. A valid measurement yields a bit (0, 1) and an invalid measurement an error symbol ($\perp$). The responder's instance sends (m_2) its receiving bases Φ_R and the indexes Ω corresponding to valid measurements.

3. The initiator's instance keeps the indexes (Σ) corresponding to valid measurements and such that its bases coincide with the responder's ones (*sifting* step). If the set Σ of indexes is large enough, then it is divided in two: a set X corresponding to the bits the ephemeral quantum key is eventually computed from, and a set V corresponding to the verification bits. The initiator's instance sends (m_3) Σ, V, as well as syndrome Z and digest T to help the responder's instance correct its own computation of the raw quantum key.
4. The responder's instance computes its raw quantum key the same way as the initiator's instance, computes the error rate ϵ (with V), then, if the latter is low enough (*parameter estimation* step), corrects the raw quantum key (with Z), and eventually checks (with T) that the value is the same as that of the initiator's instance (*error correction* step). The flag $flag_{\mathsf{pe}}$ sent by the responder's instance (m_4) informs the initiator's instance if the ephemeral quantum key is indeed shared and can actually be involved in the computation of the session keys and *pss*.

Finally, if the QKD phase is successful (i.e. $flag_{\mathsf{pe}} = \texttt{true}$), then the quantum key is computed by the initiator's and the responder's instances as $K_I = \mathsf{H}_{\mathsf{pa}}(X)$ and $K_R = \mathsf{H}_{\mathsf{pa}}(\hat{X})$ respectively (*privacy amplification* step), where $X = \hat{X}$ with overwhelming probability.

The QKD phase involves classical operations done with functions chosen at random. The parameter S (sent in m_3) indicates the corresponding choices made by the initiator's instance: S^{Ξ}, $S^{\mathsf{H}_{\mathsf{ec}}}$, $S^{\mathsf{H}_{\mathsf{pa}}}$ denote the randomness (seeds) used to select the set $\Xi \in \Xi_{m,k}$, and the 2-universal hash functions H_{ec} and H_{pa} respectively. $\Xi_{m,k} = \{\Xi \subset \{0, \ldots, m-1\} \mid |\Xi| = k\}$ denotes the set of subsets of $\{0, \ldots, m-1\}$ of size k, and Ξ is used to split the raw quantum key into two sets X and V of size $m-k$ and k respectively (with $m = |\Sigma|$).

Cryptographic Components. The (non-quantum) key exchange phase is based on a key encapsulation mechanism (KEM) KEM. The latter can either be a pure post-quantum algorithm (e.g. ML-KEM [92], HQC [1]) or a combination of the latter and a classical KEM (e.g. DHKEM [8]) as long as the required security properties (see Section 4.1) are ensured. Authentication is provided with a (post-quantum) signature scheme SIG. The corresponding keys are long-term. Confidentiality and authenticity of data exchanged during an AKE run are guaranteed with an authenticated encryption (AE) algorithm AE. Note that intermediary AE keys are computed from the internal parameter *pss* shared by two communicating parties. Therefore, the AE scheme contributes also to the parties authentication. The MAC function MAC protects the KEM ciphertext. Other symmetric-key functions (in particular the hash function H, and the pseudo-random functions $\mathsf{Extract}$, Expand) are involved in the derivation of intermediary AKE parameters and the session keys. During the QKD phase,

the function corr and the 2-universal hash function H_{ec} (along with the syndrome function synd) enable correcting and verifying the received bits (by the responder's instance), and eventually computing the final quantum key with the 2-universal hash function H_{pa}. The classical data exchanged during the QKD phase are integrity protected with the AE and signature schemes.

2.2 Technical Considerations

Signature in the Last Message. The last message sent by the (initiator's or responder's) instance includes a signature. Doing so, a transcript of all the plaintext data exchanged during a run is protected by all the (traditional) secrets owned by the instance (i.e. the KEM shared secret, the internal parameter *pss*, and its private signature key). Regarding the responder's instance, the signature could be placed in message m_2. First, the signature would then protect only a subset of the exchanged data. Secondly, even without signature in m_2, the responder's instance authenticates thanks to the *pss* parameter shared by the two communicating instances (i.e. the initiator's instance does not accept any message m_2 as it would with a message authenticated with a key derived only from the KEM shared secret). Hence, our choice of a signature in the last message.

Desynchronisation. In Muckle, the responder computes a new *SecState* value (secret internal state preshared between the communicating instances) after receiving the first message. The initiator does the same upon reception of the second message. If the latter message is not received, then the two communicating parties are desynchronised with respect to the *SecState* parameter (i.e. the two parties do not share *SecState*). What we describe here is a generic issue since, in a two-party protocol where an internal state is updated, one party has to make the first move without guarantee that its peer will follow. This issue cannot be managed without an explicit procedure that aims at resynchronising on the fly the two parties (e.g. [7, 29, 72, 121]).

The Muckle protocol does not incorporate such a mechanism. Nor do its variants as well as our proposal. However, if the two parties share the same internal state at the beginning of a run, the execution completes successfully and, in particular, they compute the same session keys. During the next protocol run, the discrepancy is detected before the session keys be computed (and application data be exchanged), since one of the protocol messages depends on the internal state (hence it will be invalid to the verifier). In case of a desynchronisation, the two parties can merely use new instances. That is, with our protocol, *pss* is reset to its initial value (be it a constant public value or a secret symmetric key), somewhat similar to the procedure adopted in ZRTP [121] (although the latter does it on the fly). As said above, there exist more convoluted solutions but expanding on such an issue is out of the scope of this paper.

Involvement of the Ephemeral Quantum Secret. As presented above, the ephemeral quantum secret is involved in the session keys derivation (when the

Initiator $(lpk_I, lsk_I, cert_I, pss)$	Responder $(lpk_R, lsk_R, cert_R, pss)$

$\underline{\mathsf{RunInitQ}(\pi)}$
$\Phi_I \xleftarrow{\$} \{+,\times\}^M$ // choice of bases
$D \xleftarrow{\$} \{0,1\}^M$ // choice of bits
$\Xi \xleftarrow{\$} \Xi_{m,k}$ with seed S^{Ξ}
$\mathsf{H}_{\mathsf{ec}} \xleftarrow{\$} \mathcal{H}_{ec}$ with seed $S^{\mathsf{H}_{\mathsf{ec}}}$
$\mathsf{H}_{\mathsf{pa}} \xleftarrow{\$} \mathcal{H}_{pa}$ with seed $S^{\mathsf{H}_{\mathsf{pa}}}$
$\rho_I \leftarrow \Gamma(D, \Phi_I)$

ρ_I ⟹

$\underline{\mathsf{RunRespQ}(\pi, \rho_I)}$
$\rho_R \leftarrow \mathcal{Q}(\rho_I)$
$\Phi_R \xleftarrow{\$} \{+,\times\}^M$ // choice of bases
$U \leftarrow \Upsilon(\rho_R, \Phi_R)$ // $U \in \{0,1,\perp\}^M$
$\Omega = \{i, i \in [M] \mid U[i] \neq \perp\}$

$\underline{\mathsf{RunInit0}(\pi)}$
$(epk, esk) \leftarrow \mathsf{KEM.Kgen}()$
$m_1 \leftarrow epk$

m_1 ⟶

$\underline{\mathsf{RunResp1}(\pi, m_1, pss)}$
$(ct, ss) \leftarrow \mathsf{KEM.Encaps}(epk)$
$tr_0 \leftarrow \mathsf{H}(epk \| ct)$
$k_0 \leftarrow \mathsf{Extract}(0, pss)$
$hk_R^\tau \leftarrow \mathsf{Expand}(k_0, label_{hk_R^\tau}, tr_0)$
$\tau_R \leftarrow \mathsf{MAC.Tag}(hk_R^\tau, ct)$
$dk_0 \leftarrow \mathsf{Expand}(k_0, label_{dk_0}, \text{“”})$
$k_1 \leftarrow \mathsf{Extract}(dk_0, ss)$
$hk_R \leftarrow \mathsf{Expand}(k_1, label_{hk_R}, tr_0)$
$n_R \leftarrow \mathsf{Expand}(k_1, label_{n_R}, tr_0)$
$\ell_{c_R} \xleftarrow{\$} \{\ell_{min}, \ldots, \ell_{max}\}$
$c_R \leftarrow \mathsf{AE.Enc}(hk_R, \ell_{c_R}, n_R, tr_0, \Phi_R \| \Omega)$ // $|c_R| = \ell_{c_R}$
$m_2 \leftarrow (ct, \tau_R, c_R)$

m_2 ⟵

$\underline{\mathsf{RunInit2}(\pi, m_2, list_{lpk}, lsk_u, pss)}$
// computation of tr_0, k_0, hk_R^τ
If $(\mathsf{MAC.Vrfy}(hk_R^\tau, ct, \tau_R) = \texttt{false}$ then abort
$ss \leftarrow \mathsf{KEM.Decaps}(esk, ct)$
// computation of dk_0, k_1, hk_R, n_R
If $(\mathsf{AE.Dec}(hk_R, n_R, tr_0, c_R) = \perp)$ then abort
// $\Phi_R \| \Omega = \mathsf{AE.Dec}(hk_R, n_R, tr_0, c_R)$

$\tilde{\Sigma} \leftarrow \{i \in \Omega \mid \Phi_I[i] = \Phi_R[i]\}$
If $(|\tilde{\Sigma}| < m)$ then abort
$\Sigma \leftarrow \{\tilde{\Sigma}[i], 0 \le i < m\}$ // $|\Sigma| = m$
$D' \leftarrow \{D[i] \mid i \in \Sigma\}$ // raw quantum key
$X \leftarrow \{D'[i] \mid i \notin \Xi\}$ // $|X| = n = m - k$
$V \leftarrow \{D'[i] \mid i \in \Xi\}$ // $|V| = k$
$S \leftarrow (S^{\Xi}, S^{\mathsf{H}_{\mathsf{ec}}}, S^{\mathsf{H}_{\mathsf{pa}}})$ // choice of Ξ, H_{ec}, H_{pa}
$Z \leftarrow \mathsf{synd}(X)$ // $|Z| = r$
$T \leftarrow \mathsf{H}_{\mathsf{ec}}(X)$ // $|T| = \lambda$

$tr_1 \leftarrow \mathsf{H}(tr_0 \| \Phi_R \| \Omega \| |c_R|)$
$hk_I \leftarrow \mathsf{Expand}(k_1, label_{hk_I}, tr_1)$
$n_I \leftarrow \mathsf{Expand}(k_1, label_{n_I}, tr_1)$
$\sigma_I \leftarrow \mathsf{SIG.Sign}(lsk_I, tr_1 \| Z \| T \| \Sigma \| V \| S \| cert_I)$
$\ell_{c_I} \xleftarrow{\$} \{\ell_{min}, \ldots, \ell_{max}\}$
$c_I \leftarrow \mathsf{AE.Enc}(hk_I, \ell_{c_I}, n_I, tr_1, Z \| T \| \Sigma \| V \| S \| cert_I \| \sigma_I)$ // $|c_I| = \ell_{c_I}$
$m_3 \leftarrow c_I$

$\xrightarrow{\qquad m_3 \qquad}$

$\underline{\mathsf{RunResp3}(\pi, m_3, list_{lpk}, lsk_u)}$
// computation of tr_1, hk_I, n_I
If $(\mathsf{AE.Dec}(hk_I, n_I, tr_1, c_I) = \perp)$ then abort
// $Z \| T \| \Sigma \| V \| S \| cert_I \| \sigma_I = \mathsf{AE.Dec}(hk_I, n_I, tr_1, c_I)$
If $(\mathsf{SIG.Vrfy}(lpk_I, tr_1 \| Z \| T \| \Sigma \| V \| S \| cert_I, \sigma_I) = \perp)$ then abort

// Responder accepts

$U' \leftarrow \{U[i] \mid i \in \Sigma\}$ // raw quantum key
$Y \leftarrow \{U'[i] \mid i \notin \Xi\}$ // $|Y| = n = m - k$
$W \leftarrow \{U'[i] \mid i \in \Xi\}$ // $|W| = k$
$\epsilon \leftarrow |\{i, i \in [k] \mid V[i] \neq W[i]\}|/k$
$flag_{\mathsf{pe}} \leftarrow \mathtt{true}$
If $(\epsilon \ge \epsilon_{max})$ then $flag_{\mathsf{pe}} \leftarrow \mathtt{false}$

If $(flag_{\mathsf{pe}} = \mathtt{true})$ then {
  $\hat{X} \leftarrow \mathsf{corr}(Y, Z)$
  $T' \leftarrow \mathsf{H}_{\mathsf{ec}}(\hat{X})$
  If $(T' = T)$ then $K_R \leftarrow \mathsf{H}_{\mathsf{pa}}(\hat{X})$ // $|K_R| = \ell$
  else $flag_{\mathsf{pe}} \leftarrow \mathtt{false}$
}

$tr_2 \leftarrow \mathsf{H}(tr_1 \| Z \| T \| \Sigma \| V \| S \| cert_I \| \sigma_I \| |c_I|)$
$\sigma_R \leftarrow \mathsf{SIG.Sign}(lsk_R, tr_2 \| flag_{\mathsf{pe}} \| cert_R)$
$hk'_R \leftarrow \mathsf{Expand}(k_1, label_{hk'_R}, tr_2)$
$n'_R \leftarrow \mathsf{Expand}(k_1, label_{n'_R}, tr_2)$
$\ell_{c'_R} \xleftarrow{\$} \{\ell_{min}, \ldots, \ell_{max}\}$
$c'_R \leftarrow \mathsf{AE.Enc}(hk'_R, \ell_{c'_R}, n'_R, tr_2, flag_{\mathsf{pe}} \| cert_R \| \sigma_R)$ // $|c'_R| = \ell_{c'_R}$
$m_4 \leftarrow c'_R$

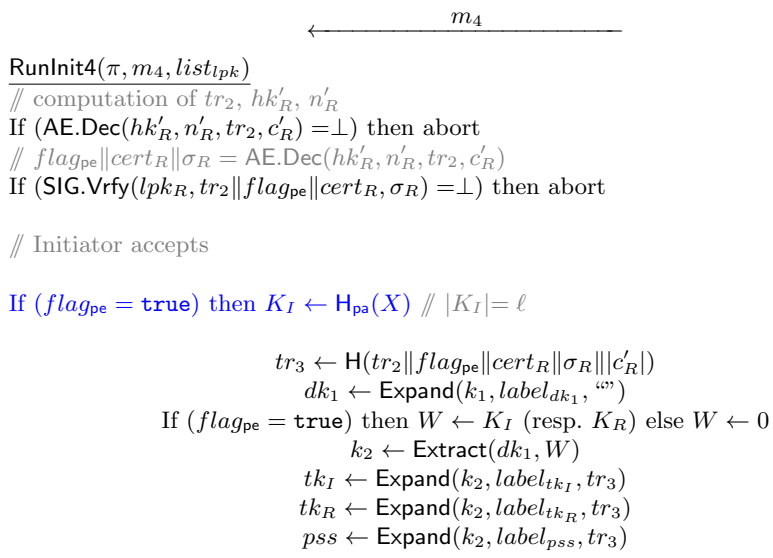


Fig. 1: Message flow corresponding to one stage of the hybrid AKE protocol. Boxed items denote qubit manipulations. The whole QKD phase is highlighted in blue. All labels ($label_*$) are distinct.

QKD phase is successful) but not in the computation of the intermediary symmetric keys (in opposite to the ephemeral KEM and pss parameter). Hence it does not contribute to protect the messages exchanged during an AKE run.

If a packet corresponding to the ephemeral KEM exchange is disrupted at the transport level, then the packet will be sent again and eventually delivered (the issue is managed at the transport level). On the contrary, regarding the quantum channel used during the QKD phase, any disruption may lead to a failure of the QKD run. This is why we decide not to incorporate the secret value output by the QKD in the computation of AKE messages. Moreover, in case of denial-of-service attacks (on the classical or the quantum channel), it is easier to redirect the packets through an alternate classical path than using an alternate quantum path.

Another option could be to design the AKE run such that a QKD run is performed first, followed by a classical key exchange (based on the ephemeral KEM and the secret internal state). This would allow (when the QKD run is successful) protecting the identity of the two communicating parties with an intermediary secret key that depends on all (quantum and traditional) ephemeral secret values. But this would add extra flows in each AKE run (since a QKD run implies that (authenticated) messages are exchanged on a classical channel). Consequently, we have decided to “compress” the run at the cost of loosing the security that the QKD phase may offer to protect the identities. In practice, given the maturity of the existing symmetric-key and public-key (both traditional and post-quantum) algorithms, this choice is reasonable and acceptable.

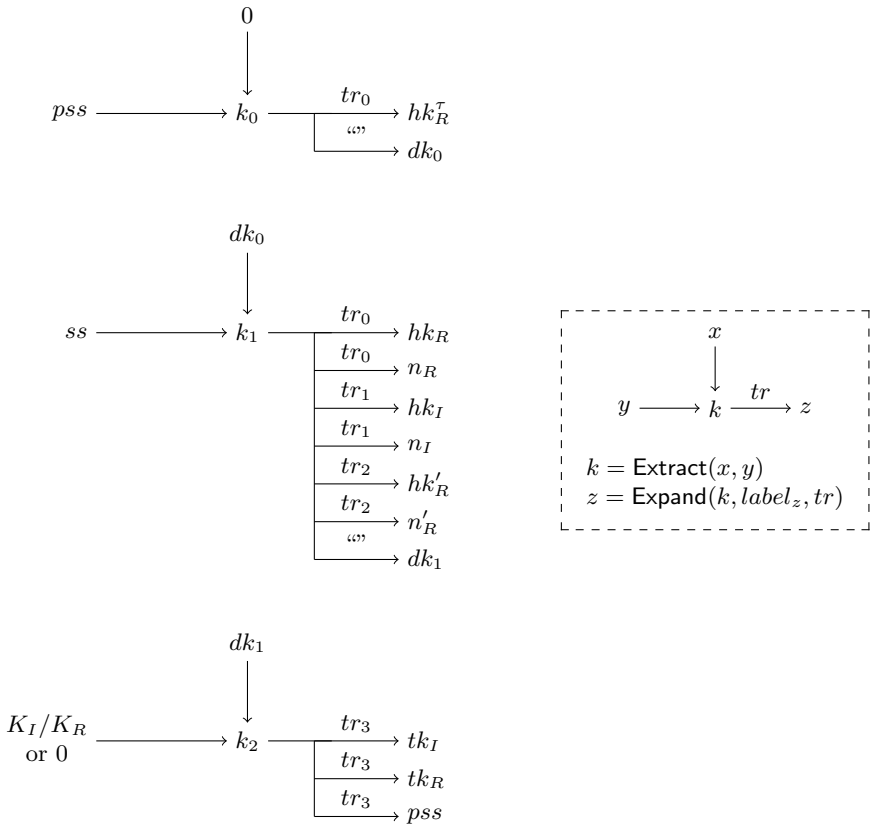


Fig. 2: Key schedule in the hybrid AKE protocol. “” denotes an empty string.

```
Run(π, m, ρ, list_lpk)
t ← π.stid
if (π.role = initiator) then                          // initiator
   if (π.status[t] =⊥) then
      π.status[t] ← running
      (π, ρ′) ← RunInitQ(π)                           // ρ = ∅
      (π, m′) ← RunInit0(π)                           // m = ∅
      return (π, m′, ρ′)                              // m′ = m1, ρ′ ≠ ∅
   else
      if (π.cid[t] =⊥) then
         u ← π.parent
         pss ← π.pss[t − 1]
         (π, m′) ← RunInit2(π, m, list_lpk, lsk_u, pss) // (m, ρ) = (m2, ∅)
         return (π, m′, ∅)                            // m′ = m3
      else
         (π, m′) ← RunInit4(π, m, list_lpk)           // (m, ρ) = (m4, ∅)
         return (π, ∅, ∅)
else                                                  // responder
   if (π.status[t] =⊥) then
      π.status[t] ← running
      π ← RunRespQ(π, ρ)                              // ρ ≠ ∅
      pss ← π.pss[t − 1]
      (π, m′) ← RunResp1(π, m, pss)                   // m = m1
      return (π, m′, ∅)                               // m′ = m2
   else
      u ← π.parent
      (π, m′) ← RunResp3(π, m, list_lpk, lsk_u)       // (m, ρ) = (m3, ∅)
      return (π, m′, ∅)                               // m′ = m4
```

Fig. 3: Definition of the Run function. Symbol ∅ denotes an absence of classical value or quantum state.

Finally, instead of considering one "self-contained" AKE run (as we do), we could also have designed an AKE protocol solely made of traditional and post-quantum algorithms, and a distinct QKD protocol continuously or regularly executed. During each AKE run, one symmetric key can then be drawn from a buffer fed with the multiple QKD runs. Such an implementation is described by some proposals (e.g. [84]). First, such a design does not decrease the amount of transmitted data nor the cryptographic computations but it increases the number of messages exchanged since the flows related to the AKE run and the QKD phase are then independent. Secondly, even if the buffer gathers a large amount of QKD keys, in order for one QKD key to be available for each key exchange performed on the classical channel, a single QKD exchange must be in average as fast as a KEM exchange.[5] Then, this implies that the protocol we describe (combining in a single run a QKD exchange and a KEM exchange) should also be efficient. Therefore, the kinematic we present should not incur more latency compared to the storage buffer scenario. The latter holds if involving a quantum key in each protocol run is required. However, if the delivery of quantum keys can be periodically suspended when the buffer is exhausted and while it is refilled, then the buffer setting enables executing the protocol at its own pace, solely based on the KEM exchange, with no impact on the latency and key rate.

As an additional note, such a storage buffer aims at compensating for the disruptions that may occur during a QKD run. We think it is not insurmountable for an adversary to perturb successive QKD runs so that the buffer is eventually exhausted. According to us, the approach based on a key storage buffer is valid if the (rare) disruptions on the quantum channel are only due to occasional, unintentional disturbances (e.g. temporary change in the environment).

Overall, with our proposal, in any case, the AKE run never aborts if an issue occurs at the transport level, and, when the QKD run completes successfully, the ephemeral quantum secret is also involved in the computation of the new session keys. In addition, recall that the session keys depend also on the *pss* parameter which is computed from the previous AKE runs (including the corresponding QKD phases).

End-to-end QKD. A QKD protocol can be established in several ways: end-to-end between two parties or point-to-point with several intermediary trusted nodes when the two parties are far away (to compensate for the signal loss induced by the distance) [3, 68]. Our AKE protocol requires that the QKD run be done between the two communicating parties without intermediate entity (trusted nodes). In practice, this limits the distance between two QKD endpoints to 100-200 Km (assuming a fiber-based quantum channel) where the key rate decreases with the distance [26, 27, 54]. Moreover, it is not realistic to consider that a dedicated fiber-based quantum channel be established between each pair of communicating parties. On the contrary, a single quantum channel is shared

[5] Unless the duration of a single QKD exchange cannot be lower than some threshold (corresponding to n KEM exchanges) but outputs enough quantum key material for at least n AKE exchanges.

by multiple parties, and the corresponding QKD endpoints accessed by these parties. The link between a party and a QKD endpoint (entry point to the quantum channel) must therefore be protected. We do not incorporate the latter into our security analysis and assume that such a link is trusted.

3 Security Model

We define our security model following the approach of Davis and Günther [45] and borrowing some components from the HAKE framework described in [48,49]. In our model, we incorporate explicit security properties in addition to key indistinguishability. The additional properties are: soundness, explicit authentication and identity privacy. Our security model captures also key-compromise impersonation attacks (KCI) [24], forward secrecy and post-compromise security (PCS) [38] (through the FreshAuth, FreshKey and FreshIdentity predicates defined in Section 3.1).

The adversary is active: it controls all (classical and quantum) communication and can transmit, relay, alter (up to some point regarding the quantum states), drop any message exchanged by honest parties, or insert new messages. The adversary is also able to corrupt long-term public keys, reveal session keys, and reveal intermediary secret parameters (internal state, ephemeral post-quantum or quantum keys). The goal of the adversary is to (i) distinguish the real session key from a value of same size uniformly drawn at random (key indistinguishability), (ii) make an instance accept without an honest partner (authentication), or (iii) distinguish which party out of two is active during a protocol run (identity privacy).

Quantum queries (in superposition) are not granted to the adversary (unless the latter can implement locally the targeted cryptographic function and use its quantum computing power).

3.1 Execution Environment

We describe the security model for AKE protocols. The corresponding security experiments are depicted by Figure 5.

Parties, Instances, Stages. A two-party AKE protocol can be implemented at different levels through parties, instances and stages.

Parties. A two-party AKE protocol is carried out by a set of parties, which maximum size is denoted by n_P. Each party u has an associated pair of long-term public and private keys (lpk, lsk). The algorithm LCKeyGen generates a new pair of long-term keys. The public keys (in the form of certificates *cert*) are stored in a list $list_{lpk}$. The parameter lck_u denotes the long-term private key of party u. When the private key is revealed (by querying the oracle RevealLCK), the parameter $\overline{\mathsf{lck}_u}$ is set to the time of reveal (we say that party u is *corrupted*).

Instances. Each party can take part in multiple sequential or parallel executions of the protocol. Each run of the protocol is called a session. To each session of a party u, an instance π is associated which embodies this session's execution (in stage t) of the protocol and maintains the following state specific to the session:

- $\mathsf{role} \in \{\texttt{initiator}, \texttt{responder}\}$: the role of the instance in the protocol execution, being either the initiator or the responder.
- parent: the party u which an instance π_u^i is associated to.
- pid: the identity of the communication partner party of the instance. It is learned and set during the protocol execution.
- stid: the stage of the instance that is currently ongoing or that has been completed the most recently.
- $\mathsf{sid}[t]$: the identifier of the stage t, initialised to $\perp$.
- $\mathsf{cid}[t]$: the contributive identifier of the stage t, initialised to $\perp$.
- $\mathsf{status}[t] \in \{\perp, \texttt{running}, \texttt{accepted}, \texttt{rejected}\}$: the status of the instance in stage t, initialised to $\perp$.
- $\mathsf{t}_{\mathsf{acc}}[t]$: the time when the instance accepts in stage t, initialised to 0.
- $\mathsf{sk}[t]$: the session key computed by the instance in stage t, initialised to $\perp$.
- $\mathsf{eqk}[t]$: the ephemeral quantum shared secret computed by the instance in stage t (i.e. the ephemeral shared secret output by the QKD phase), initialised to $\perp$.
- $\mathsf{eck}[t]$: the ephemeral asymmetric secret key computed by the instance in stage t, initialised to $\perp$.[6]
- $\mathsf{pss}[t]$: the internal secret state (initialised to $\perp$) computed by the instance during stage $t > 0$ and for use in stage $t + 1$. $\mathsf{pss}[0]$ denotes the value of the internal state at the beginning of the first stage.
- $\bar{\mathsf{x}}[t] \in \{\texttt{clean}, \texttt{revealed}\}$: for each parameter $\mathsf{x} \in \{\mathsf{sk}, \mathsf{eqk}, \mathsf{eck}, \mathsf{pss}\}$, it indicates whether $\mathsf{x}[t]$ is revealed; initialised to $\texttt{clean}$.
- $\mathsf{tested}[t] \in \{\texttt{false}, \texttt{true}\}$: indicates whether the instance is tested in stage t, initialised to $\texttt{false}$.
- $\mathsf{m}_\mathsf{s}[t]$: list of the timely ordered messages sent in stage t, updated by the instance during the stage, initialised to $\emptyset$.
- $\mathsf{m}_\mathsf{r}[t]$: list of the timely ordered messages received in stage t, updated by the instance during the stage, initialised to $\emptyset$.

The stages of a given instance are sequential. n_S and n_T denote the maximum number of instances per party, and the maximum number of stages per instance respectively.

We stress that the aforementioned parameters are managed by the challenger of the security experiments but not accessible to the adversary (unless a parameter stores a public value (e.g. $\mathsf{m}_\mathsf{s}[t]$, $\mathsf{m}_\mathsf{r}[t]$) or its content is available through an oracle provided to the adversary).

[6] In the AKE protocol that we describe next, the ephemeral private post-quantum key is computed by the initiator's instance only.

Partnering. We define the notion of partnering (see Definition 3) between two instances through that of *session identifier* (sid). In turn, in our context, we will define the sid through the notion of *matching instances* [16, 23, 47].

When two instances *match* in stage t, then all the messages sent by one instance are received (unmodified) by the other one, and conversely, during this stage. We also use the notion of *origin instance* [43]. An instance π is an origin instance of π' if, in particular, the messages received by π' originate (unmodified) from π (even though π' may have not received all the messages sent by π). The notion of origin instance allows considering attack scenarios when there exists no instance matching the targeted instance, without need to overly restrict the adversary's behaviour (the origin instance can be seen as a "potential" matching instance). These two notions are formally defined in Definitions 1 and 2 (see also Figure 4).

The origin and matching instances notions serve also to capture whether the adversary is active (it interacts with an honest instance) or passive (it merely eavesdrops on the messages exchanged between two honest instances).

Definition 1 (Matching instances). *An instance π matches an instance π' in stage t if $\pi.\mathsf{m_r}[t] = \pi'.\mathsf{m_s}[t]$ and $\pi.\mathsf{m_s}[t] = \pi'.\mathsf{m_r}[t]$.*

Definition 2 (Origin instance). *An instance π prefix-matches an instance π' in stage t if $\mathsf{trunc}(\pi.\mathsf{m_s}[t], |\pi'.\mathsf{m_r}[t]|) = \pi'.\mathsf{m_r}[t]$. An instance π is an origin instance of π' in stage t if π prefix-matches π' in stage t (and π has sent the last message) or $\pi.\mathsf{m_s}[t] = \pi'.\mathsf{m_r}[t]$ (and π' has sent the last message).*

Definition 3 (Partnering). *Two instances are partnered in stage t if they share the same session identifier sid in stage t.*

In our security proofs, we define the session identifier sid of an instance π in stage t as the concatenation of the timely ordered messages sent and received (resp. received and sent) if $\pi.\mathsf{role} = \texttt{initiator}$ (resp. $\pi.\mathsf{role} = \texttt{responder}$). Otherwise said, in our analysis, two instances are partnered in stage t if they are matching instances in stage t.

Moreover, we define the *contributive identifier* cid of an instance π in stage t from a subset of components that constitute $\pi.\mathsf{sid}[t]$. The cid is used to capture the minimal set of public values that two instances have to share in order to be "bound" together (this will be used in particular in the proof of explicit authentication). Two instances sharing the same cid can be seen as "partial" partners. In our proofs, we use $\mathsf{cid} = epk \| ct$ where epk and ct are the ephemeral KEM public key and ciphertext respectively.

Remark 2. With respect to the AKE protocol described in Section 2, we define sid in stage t as $m_1 \| \cdots \| m_4$ (when the stage completes successfully) and cid in stage t as $epk \| ct$. Note that, on the responder's side cid is set once the KEM encapsulation is done (cf. Figure 1), while, on the initiator's side, cid is set if m_2 is valid (i.e. the MAC tag τ_R and the AE decryption of c_R are correct). Then only, m_2 is included in the list $\mathsf{m_r}$ of the message received by the initiator's instance.

$\underline{\mathsf{match}(\pi, \pi', t)}$
if $(\pi.\mathsf{m_s}[t] = \pi'.\mathsf{m_r}[t] \wedge$
$\pi.\mathsf{m_r}[t] = \pi'.\mathsf{m_s}[t])$ then
return `true`
return `false`

$\underline{\mathsf{origin}(\pi, \pi', t)}$
if $(\pi.\mathsf{m_s}[t] = \pi'.\mathsf{m_r}[t] \vee$
$\mathsf{trunc}(\pi.\mathsf{m_s}[t], |\pi'.\mathsf{m_r}[t]|) = \pi'.\mathsf{m_r}[t])$ then
return `true`
return `false`

Fig. 4: Description of the matching instance and origin instance functions used in Figure 5

Adversarial Queries. Figure 5 and Figure 6 depict the AKE security experiments $G^{\mathsf{sec}}_{\Pi,\mathcal{A}}$.

The oracles INITIALISE, NEWPARTY, NEWSESSION$(w, role)$, NEWSTAGE(π) initialise the security experiments, create a new party, a new instance related to some party w and with role $role$, and a new stage for the instance π respectively. The oracle FINALISE terminates the security experiments and verifies if the adversary wins one of them.

The oracle SEND(π, m, ρ) allows the adversary to interact with an instance π which receives a message m sent through the classical channel, and a message ρ (quantum states) sent through the quantum channel, both chosen by the adversary. This oracle activates an honest execution of the protocol (reflected by the function Run defined according to the specific AKE protocol that is considered), updates the instance π and optionally returns (non-quantum and quantum) messages m' and ρ'.

As said above, the oracle SEND allows the adversary to transmit, relay, alter, drop any message exchanged by honest instances, or insert new messages. This is straightforward regarding classical (i.e. non-quantum) messages. Regarding the quantum communication, relaying a message means that the adversary chooses which instance will be the recipient of the quantum states output by some other instance and carried through a quantum channel, and remains passive apart from that. Moreover, altering a quantum message means that the adversary becomes the recipient of the message, possibly applies operations (such as measurements) to the quantum states, and then sends the modified quantum states through the quantum channel to a final recipient instance of its choice. Overall, we stress that such operations do not necessarily imply destructing the quantum states.

The adversary is allowed to reveal multiple types of parameters: session keys with REVEALSK, long-term keys with REVEALLCK, ephemeral KEM keys with REVEALECK, ephemeral quantum keys with REVEALEQK, and secret internal states with REVEALPSS. Not all oracles can be queried at any time. The different freshness predicates (defined in Section 3.2) exclude the combinations of queries that lead to trivial attacks.

The oracles DRAWVID and FREEVID enable the adversary to select and release the two identities involved in the identity privacy experiment, whereas the oracle TEST, used during the key indistinguishability experiment, outputs either

an "ideal" (uniformly random) key or the real session key. The oracle TEST can be requested once only per security experiment.

3.2 Security Properties

Main Properties. We present below the four main security properties captured by our security model: soundness, explicit authentication, key indistinguishability and identity privacy.

Soundness. The soundness predicate Sound, when true, guarantees that only two session identifiers can be equal in stage t, and that two instances ending in accepting state in stage t which share the same session identifier, mutually agree on their partner's identity, and have opposite roles, do compute the same session key.

Explicit Authentication. The explicit authentication predicate ExplicitAuth, when true, ensures that all instances accepting in stage t, such that FreshAuth is true (meaning essentially that the peer is uncorrupted and the internal state *pss* unrevealed), are partnered (i.e. share the same session identifier as another instance), and: the instance and its partner have opposite roles, share the same contributive identifier, share the same *pss* parameter before and after stage t's completion, and they mutually agree on their partner's identity (upon acceptance). Our model captures also key-compromise impersonation attacks [24] by allowing the parent party of the targeted instance to be corrupted at any point in time.

Key Indistinguishability. The key indistinguishability property, when verified, ensures that the session keys output in stage t are "ideal", in the sense that they are indistinguishable from values of same size uniformly chosen at random. Again, the FreshKey predicate defines (by excluding them) the trivial attacks. Likewise, we exclude the reveal of session keys but only for the tested instance and its partner instance. This captures key independence. That is, if the revealed keys are different from the tested instance's keys, then such keys must not enable computing the targeted session keys. The key indistinguishability property captures also forward secrecy by allowing the adversary to reveal the long-term asymmetric private key of the targeted instance's peer (upon acceptance), as well as post-compromise security [38]. The latter is achieved by allowing the adversary to reveal all the secret values of an instance as long as there has been at least one unrevealed secret value at some previous point in time such that the current session keys "inherit" from the secrecy of this unrevealed value.

Identity Privacy. The identity privacy property aims at ensuring the confidentiality of a party's identity. We define the corresponding security experiment with the notion of *virtual identity* proposed by Ouafi and Phan [94] and Hermans, Pashalidis, Vercauteren and Preneel [63].

A virtual identity $vid = (u, v)$ designates one out of two possible parties u

$G^{\mathsf{rsec}}_{\Pi,\mathcal{A}}$

INITIALISE
$\mathsf{time} \leftarrow 0$
$n \leftarrow 0$
$flag_{\mathsf{test}} \leftarrow \mathtt{false}$
$(\pi_{\mathsf{test}}, t_{\mathsf{test}}) \leftarrow (\bot, \bot)$
$\mathcal{V} \leftarrow \emptyset$
$\mathcal{P} \leftarrow \emptyset$
$\mathcal{I} \leftarrow \emptyset$
$b_{\mathsf{key}} \xleftarrow{\$} \{0,1\}$
$b_{\mathsf{id}} \xleftarrow{\$} \{0,1\}$

NEWPARTY
$n \leftarrow n + 1$
$(lpk_n, lsk_n) \leftarrow \Pi.\mathsf{LCKeyGen}()$
$\mathsf{lck}_n \leftarrow lsk_n$
$\overline{\mathsf{lck}_n} \leftarrow \infty$
return lpk_n

NEWSESSION$(w, role)$
$\pi.\mathsf{parent} \leftarrow \mathsf{realid}(w)$
if $(w \in \mathcal{V})$ then
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{\pi\}$
$\pi.\mathsf{role} \leftarrow role$
$\pi.\mathsf{stid} \leftarrow 0$
return π

NEWSTAGE(π)
$\pi.\mathsf{stid} \leftarrow \pi.\mathsf{stid} + 1$
$t \leftarrow \pi.\mathsf{stid}$
$\pi.\mathsf{sid}[t] \leftarrow \bot$
$\pi.\mathsf{cid}[t] \leftarrow \bot$
$\pi.\mathsf{sk}[t] \leftarrow \bot$
$\pi.\mathsf{eck}[t] \leftarrow \bot$
$\pi.\mathsf{eqk}[t] \leftarrow \bot$
$\pi.\mathsf{pss}[t] \leftarrow \bot$
$\pi.\overline{\mathsf{sk}}[t] \leftarrow \mathtt{clean}$
$\pi.\overline{\mathsf{eck}}[t] \leftarrow \mathtt{clean}$
$\pi.\overline{\mathsf{eqk}}[t] \leftarrow \mathtt{clean}$
$\pi.\overline{\mathsf{pss}}[t] \leftarrow \mathtt{clean}$
$\pi.\mathsf{tested}[t] \leftarrow \mathtt{false}$
$\pi.\mathsf{t}_{\mathsf{acc}}[t] \leftarrow 0$
$\pi.\mathsf{m}_{\mathsf{s}}[t] \leftarrow \emptyset$
$\pi.\mathsf{m}_{\mathsf{r}}[t] \leftarrow \emptyset$
$\pi.\mathsf{status}[t] \leftarrow \bot$
return π

SEND(π, m, ρ)
$t \leftarrow \pi.\mathsf{stid}$
$u \leftarrow \pi.\mathsf{parent}$
$(\pi, m', \rho') \leftarrow \mathsf{Run}(\pi, m, \rho, list_{lpk})$
if $(\pi.\mathsf{status}[t] = \mathtt{accepted})$ then
  $\mathsf{time} \leftarrow \mathsf{time} + 1$
  $\pi.\mathsf{t}_{\mathsf{acc}}[t] \leftarrow \mathsf{time}$
if $(\pi.\mathsf{status}[t] = \mathtt{rejected})$ then
  return $\bot$
$\pi.\mathsf{m}_{\mathsf{r}}[t] \leftarrow \pi.\mathsf{m}_{\mathsf{r}}[t] \| m$
$\pi.\mathsf{m}_{\mathsf{s}}[t] \leftarrow \pi.\mathsf{m}_{\mathsf{s}}[t] \| m'$
return (m', ρ')

REVEALSK(π, t)
if $(\pi.\mathsf{status}[t] \neq \mathtt{accepted})$ then
  return $\bot$
$\pi.\overline{\mathsf{sk}}[t] \leftarrow \mathtt{revealed}$
return $\pi.\mathsf{sk}[t]$

REVEALLCK(u)
$\mathsf{time} \leftarrow \mathsf{time} + 1$
$\overline{\mathsf{lck}_u} \leftarrow \mathsf{time}$
$\forall\ vid \in \mathcal{V} \mid (vid = (u, *) \vee vid = (*, u))$,
  $\overline{\mathsf{lck}_{vid}} \leftarrow \mathsf{time}$
return lck_u

REVEALECK(π, t)
$\pi.\overline{\mathsf{eck}}[t] \leftarrow \mathtt{revealed}$
return $\pi.\mathsf{eck}[t]$

REVEALEQK(π, t)
$\pi.\overline{\mathsf{eqk}}[t] \leftarrow \mathtt{revealed}$
return $\pi.\mathsf{eqk}[t]$

REVEALPSS(π, t)
$\pi.\overline{\mathsf{pss}}[t] \leftarrow \mathtt{revealed}$
return $\pi.\mathsf{pss}[t]$

DRAWVID(u, v)
if $(u \in \mathcal{P} \vee v \in \mathcal{P})$ then
  return $\bot$
$\mathcal{P} \leftarrow \mathcal{P} \cup \{u, v\}$
$vid \leftarrow (u, v)$
$\mathcal{V} \leftarrow \mathcal{V} \cup \{vid\}$
$\overline{\mathsf{lck}_{vid}} \leftarrow \min(\overline{\mathsf{lck}_u}, \overline{\mathsf{lck}_v})$
return vid

FreeVid(vid)
$\forall\ (\pi, t) \in \mathcal{I} \times \{1, \ldots, \pi.\mathsf{stid}\}$,
  if $(\pi.\mathsf{parent} = \mathsf{realid}(vid) \wedge$
    $\pi.\mathsf{status}[t] = \texttt{running})$ then
    $\pi.\mathsf{status}[t] \leftarrow \texttt{rejected}$
  $\forall\ \pi'$,
    if $(\pi'.\mathsf{sid}[t] = \pi.\mathsf{sid}[t] \wedge$
      $\pi'.\mathsf{status}[t] = \texttt{running})$ then
      $\pi'.\mathsf{status}[t] \leftarrow \texttt{rejected}$
if $(vid = (u, v))$ then
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \{u, v\}$
$\mathcal{V} \leftarrow \mathcal{V} \setminus \{vid\}$

Test(π, t)
if $(flag_{\mathsf{test}} = \texttt{true})$ then
  return $\perp$
if $(\pi.\mathsf{status}[t] \neq \texttt{accepted} \vee$
  $\pi.\mathsf{tested}[t])$ then
  return $\perp$
$\pi.\mathsf{tested}[t] \leftarrow \texttt{true}$
$(\pi_{\mathsf{test}}, t_{\mathsf{test}}) \leftarrow (\pi, t)$
$flag_{\mathsf{test}} \leftarrow \texttt{true}$
$sk^* \xleftarrow{\$} \{0,1\}^{|\pi.\mathsf{sk}[t]|}$
if $(b_{\mathsf{key}} = 1)$ then
  return sk^*
return $\pi.\mathsf{sk}[t]$

Finalise($[b^*]$)
return Finalise$^{\mathsf{sec}}([b^*])$

Sound
if $(\exists\ t, \text{distinct } \pi, \pi', \pi'' \mid$
  $\pi.\mathsf{sid}[t] = \pi'.\mathsf{sid}[t] = \pi''.\mathsf{sid}[t] \neq \perp)$ then
  return `false`
if $(\exists\ \pi, \pi', t \mid (\pi.\mathsf{status}[t] = \pi'.\mathsf{status}[t]$
    $= \texttt{accepted} \wedge$
    $\pi.\mathsf{sid}[t] = \pi'.\mathsf{sid}[t] \wedge$
    $\pi.\mathsf{pid} = \pi'.\mathsf{parent} \wedge$
    $\pi'.\mathsf{pid} = \pi.\mathsf{parent} \wedge$
    $\pi.\mathsf{role} \neq \pi'.\mathsf{role}))$ then
  if $(\pi.\mathsf{sk}[t] \neq \pi'.\mathsf{sk}[t])$ then
    return `false`
return `true`

ExplicitAuth
return $[[\forall\ \pi, t, \mathsf{FreshAuth}(\pi, t) = \texttt{true}$
  $\Rightarrow \exists\ \pi' \mid (\pi.\mathsf{pid} = \pi'.\mathsf{parent} \wedge$
    $\pi.\mathsf{cid}[t] = \pi'.\mathsf{cid}[t] \wedge$
    $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1] \wedge$
    $\pi.\mathsf{role} \neq \pi'.\mathsf{role} \wedge$
    $(\pi'.\mathsf{status}[t] = \texttt{accepted}$
     $\Rightarrow (\pi'.\mathsf{pid} = \pi.\mathsf{parent} \wedge$
       $\pi'.\mathsf{pss}[t] = \pi.\mathsf{pss}[t]))]]$

Fig. 5: Security experiments for hybrid AKE protocol ($\mathsf{sec} \in \{\mathsf{sound}, \mathsf{auth}, \mathsf{key\text{-}ind}, \mathsf{id\text{-}priv}\}$). In Finalise, $[b^*]$ denotes an optional input b^*. The Finalise$^{\mathsf{sec}}$ oracles are depicted by Figure 6.

Finalise$^{\mathsf{sound}}$
if $(\neg\mathsf{Sound})$ then
  return 1
return 0

Finalise$^{\mathsf{auth}}$
if $(\neg\mathsf{ExplicitAuth})$ then
  return 1
return 0

Finalise$^{\mathsf{key\text{-}ind}}(b^*)$
if $(\neg\mathsf{FreshKey}(\pi_{\mathsf{test}}, t_{\mathsf{test}}))$ then
  $b^* \leftarrow 0$
return $[[b_{\mathsf{key}} = b^*]]$

Finalise$^{\mathsf{id\text{-}priv}}(b^*)$
if $(\exists\ \pi \in \mathcal{I} \mid \neg\mathsf{FreshIdentity}(\pi))$ then
  $b^* \leftarrow 0$
return $[[b_{\mathsf{id}} = b^*]]$

Fig. 6: The Finalise$^{\mathsf{sec}}$ oracles for the security experiments depicted by Figure 5 ($\mathsf{sec} \in \{\mathsf{sound}, \mathsf{auth}, \mathsf{key\text{-}ind}, \mathsf{id\text{-}priv}\}$)

and v. The adversary wins the identity privacy experiment if it succeeds in guessing which party lies behind vid. $\mathsf{realid}(vid)$ denotes the real involved party that is hidden under $vid = (u, v)$, depending on the secret identity bit $b_{\mathsf{id}} \in \{0, 1\}$. This bit is sampled at the initialisation of the security experiment. If $b_{\mathsf{id}} = 0$ then $\mathsf{realid}(vid) = u$, otherwise $\mathsf{realid}(vid) = v$. Moreover, for any party u, we set $\mathsf{realid}(u) = u$.

The oracles DrawVid and FreeVid are used to create and release respectively a virtual identity. We say that a virtual identity $vid = (u, v)$ is corrupted if either party u or party v is corrupted. The parameter $\overline{\mathsf{lck}}_{vid}$ corresponds to the time when vid is corrupted (set to ∞ if vid is uncorrupted).

We follow also the paradigm proposed by Schäge, Schwenk and Lauer [104]. That is, the identity privacy property is incorporated together with the other security properties. This approach guarantees that the different security properties are independent of each other, and any security property holds in the presence of attacks targetting another security property. This leads to a stronger security model compared to models where security properties are considered separately and not all the adversarial queries are available in all the experiments.

Trivial Attacks. We use predicates to define (and exclude) trivial attacks. That is, a combination of oracle queries that enables the adversary to unavoidably win a security experiment. As such, each predicate lists the different combinations of secret values that the adversary is allowed to compromise without trivially breaking the considered security property. A predicate is associated to each of the three following security properties: explicit authentication (Definition 4 and Figure 7), key indistinguishability (Definition 5 and Figure 8), identity privacy (Definition 6 and Figure 9). Each predicate is specific to the protocol we analyse (cf. Figure 1) and to the security model used in the analysis.

Definition 4 (FreshAuth). *An instance π in stage t such that $\pi.\mathsf{status}[t] =$ `accepted` is fresh with respect to the explicit authentication (π is "auth-fresh") in the security experiment defined in Figure 5 if at least one of the following conditions hold:*

a) The two following conditions hold:
   1) RevealLCK$(\pi.\mathsf{pid})$ *has not been queried before π accepts in stage t.*
   2) RevealECK(π, t) *and* RevealECK(π', t) *have not been queried before π' accepts in stage t, for any instance π' that shares the same* cid *in stage t as π.*

b) If there exists an instance π' such that π and π' share the same value pss at the beginning of stage t (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$), then at least one of the following conditions holds:
   1) RevealECK(π, t') *and* RevealECK(π', t') *have not been queried where π' matches π in stages u such that $t' \leq u < t$, and* RevealPSS(π, u) *and* RevealPSS(π', u) *have not been queried, before the last accepting instance accepts in stage t.*

2) $\textsc{RevealEQK}(\pi, t')$ *and* $\textsc{RevealEQK}(\pi', t')$ *have not been queried where* $\pi.\mathsf{eqk}[t'] \neq \perp$ *and* $\pi'.\mathsf{eqk}[t'] \neq \perp$*, and* π' *matches* π *in stages* u *such that* $t' \leq u < t$*, and* $\textsc{RevealPSS}(\pi, u)$ *and* $\textsc{RevealPSS}(\pi', u)$ *have not been queried, before the last accepting instance accepts in stage* t*.*

3) pss is initialised with a secret value uniformly chosen at random, and $\textsc{RevealPSS}(\pi, 0)$ *and* $\textsc{RevealPSS}(\pi', 0)$ *have not been queried where* π' *matches* π *in stage* u *such that* $0 < u < t$*, and* $\textsc{RevealPSS}(\pi, u)$ *and* $\textsc{RevealPSS}(\pi', u)$ *have not been queried, before the last accepting instance accepts in stage* t*.*

In condition a) of $\mathsf{FreshAuth}$, the first subcase forbids the adversary from impersonating in stage t the peer parent of the targeted instance π (with the private signature key, the adversary can successfully complete a protocol run). The second subcase of condition a) aims at forbidding the adversary from computing the intermediary secret k_1 necessary to compute the symmetric key (hk_I if π is initiator, hk'_R if π is responder) that is used to send π's last message in stage t. π's putative partner instance accepts upon reception of this last message which includes a signature. But, the adversary can reveal the private signature key of another party $w \neq \pi.\mathsf{parent}$, compute a valid signature and eventually compute a valid last message (if it knows the corresponding AE key). In particular, if π is a responder's instance, then π accepts first and sends the last message of the protocol run, and the latter scenario is doable.[7] In such a case, the putative partner instance sets $\mathsf{pid} = w$ (instead of $\mathsf{pid} = \pi.\mathsf{parent}$) which implies $\mathsf{ExplicitAuth} = \texttt{false}$. To compute k_1, the two intermediary secret values eck and pss are necessary. Here, we consider the case when the KEM private key (eck) remains unknown to the adversary. Note that a query $\textsc{RevealECK}(\pi, t)$ returns $\perp$ if π is a responder's instance. For simplicity, we do not dissociate the two cases (initiator and responder).

Condition b) considers the case when *pss* (pss) remains unknown to the adversary (here again, we aim at forbidding the adversary from computing k_1). In order for *pss* to be "safe", it must be computed either from an uncompromised KEM exchange (subcase b)1)), or an uncompromised QKD phase (subcase b)2)), or an uncompromised initial *pss* value (uniformly drawn at random; subcase b)3)).

Note that the adversary is allowed to reveal at any point in time the long-term private key belonging to the parent party of the targeted instance. Consequently, the $\mathsf{FreshAuth}$ predicate captures also key-compromise impersonation attacks (KCI) [24].

Remark 3. In predicate $\mathsf{FreshAuth}$, the plausibility of cases b)1) and b)2) is due to case a) (and its security proof that we provide in Section A.2). That is, the

[7] We do not restrict condition a)2) to π being a responder's instance because we need this condition (π being an initiator's or a responder's instance) to rely upon the INT-CTXT security of the AE scheme (in the security proof, we will use the unforgeability of the first AE ciphertext received by π to show that π and its putative partner instance share the same *pss* at beginning of the stage).

existence of an instance π' that eventually matches π in cases b)1) and b)2) is (implicitly) justified by the fact that FreshAuth is proven to be true in case a). Likewise, while the two instances mutually authenticate with a private signature key in case a), they leverage an (evolving) pre-shared secret key (*pss*) in case b)3).

$\underline{\mathsf{FreshAuth}(\pi, t)}$
if $(\pi.\mathsf{status}[t] \neq \texttt{accepted})$ then
  return $\bot$
// *subcase a)*
if $(\pi.\mathsf{t_{acc}}[t] < \overline{\mathsf{lck}_{\pi.\mathsf{pid}}} \wedge$
  $\forall\ \pi' \mid \pi'.\mathsf{cid}[t] = \pi.\mathsf{cid}[t],$
    $(\pi'.\overline{\mathsf{eck}}[t] = \pi.\overline{\mathsf{eck}}[t] = \texttt{clean})$ until $\pi'.\mathsf{status}[t] = \texttt{accepted})$ then
  return `true`
// *subcase b)*
if $(\exists\ \pi' \neq \pi \mid \pi'.\mathsf{pss}[t-1] = \pi.\mathsf{pss}[t-1])$ then
  if ([[// *subcase b)1)*
    $\exists\ t', t' < t \mid (\pi.\overline{\mathsf{eck}}[t'] = \pi'.\overline{\mathsf{eck}}[t'] = \texttt{clean} \wedge$
      $\forall\ u, t' \leq u < t,$
        $(\mathsf{match}(\pi, \pi', u) = \texttt{true} \wedge$
          $\pi.\overline{\mathsf{pss}}[u] = \pi'.\overline{\mathsf{pss}}[u] = \texttt{clean}))$ until $\tilde{\pi}.\mathsf{status}[t] = \texttt{accepted}$ where
      $\tilde{\pi} \in \{\pi, \pi'\}$ and $\tilde{\pi}.\mathsf{t_{acc}}[t] = \max(\pi.\mathsf{t_{acc}}[t], \pi'.\mathsf{t_{acc}}[t])$
    $\vee$
    // *subcase b)2)*
    $\exists\ t', t' < t \mid (\pi.\mathsf{eqk}[t'] \neq \bot \wedge \pi'.\mathsf{eqk}[t'] \neq \bot \wedge$
      $((\pi.\overline{\mathsf{eqk}}[t'] = \pi'.\overline{\mathsf{eqk}}[t'] = \texttt{clean} \wedge$
        $\forall\ u, t' \leq u < t,$
          $(\mathsf{match}(\pi, \pi', u) = \texttt{true} \wedge$
            $\pi.\overline{\mathsf{pss}}[u] = \pi'.\overline{\mathsf{pss}}[u] = \texttt{clean}))$ until $\tilde{\pi}.\mathsf{status}[t] = \texttt{accepted}$ where
        $\tilde{\pi} \in \{\pi, \pi'\}$ and $\tilde{\pi}.\mathsf{t_{acc}}[t] = \max(\pi.\mathsf{t_{acc}}[t], \pi'.\mathsf{t_{acc}}[t]))))$
    $\vee$
    // *subcase b)3)*
    ($\pi.\mathsf{pss}[0]$ and $\pi'.\mathsf{pss}[0]$ are set to a uniformly random value $\wedge$
      $((\pi.\overline{\mathsf{pss}}[0] = \pi'.\overline{\mathsf{pss}}[0] = \texttt{clean} \wedge$
        $\forall\ u, 0 < u < t,$
          $(\mathsf{match}(\pi, \pi', u) = \texttt{true} \wedge$
            $\pi.\overline{\mathsf{pss}}[u] = \pi'.\overline{\mathsf{pss}}[u] = \texttt{clean}))$ until $\tilde{\pi}.\mathsf{status}[t] = \texttt{accepted}$ where
        $\tilde{\pi} \in \{\pi, \pi'\}$ and $\tilde{\pi}.\mathsf{t_{acc}}[t] = \max(\pi.\mathsf{t_{acc}}[t], \pi'.\mathsf{t_{acc}}[t])))$
    ]] = `true`) then
    return `true`
return `false`

Fig. 7: FreshAuth predicate

The predicate FreshKey is adapted from Garms et al. [57, 58], which in turn take inspiration from Dowling et al. [48, 49].

Definition 5 (FreshKey). *A (tested) instance π in stage t such that $\pi.\mathsf{status}[t] =$ `accepted` is fresh with respect to the session key (π is "key-fresh") in the security experiment defined in Figure 5 if the following conditions hold:*

a) $\textsc{RevealSK}(\pi, t)$ *has not been queried.*

b) For any instance π' that matches π in stage t, $\textsc{RevealSK}(\pi', t)$ *has not been queried.*

c) If, in stage t, there exists an instance π' such that
- *if π is an initiator's instance, then π and π' match;*
- *if π is a responder's instance, then π is an origin instance of π' and π' is an origin instance of π;*

then at least one of the following conditions holds:

1) $\textsc{RevealECK}(\pi, t)$ *and* $\textsc{RevealECK}(\pi', t)$ *have not been queried.*

2) $\textsc{RevealEQK}(\pi, t)$ *and* $\textsc{RevealEQK}(\pi', t)$ *have not been queried such that $\pi.\mathsf{eqk}[t] \neq \perp$ and $\pi'.\mathsf{eqk}[t] \neq \perp$.*

3) $\textsc{RevealECK}(\pi, t')$ *and* $\textsc{RevealECK}(\pi', t')$ *have not been queried where π' matches π in stages u such that $t' \leq u < t$, and* $\textsc{RevealPSS}(\pi, u)$ *and* $\textsc{RevealPSS}(\pi', u)$ *have not been queried.*

4) $\textsc{RevealEQK}(\pi, t')$ *and* $\textsc{RevealEQK}(\pi', t')$ *have not been queried where $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi'.\mathsf{eqk}[t'] \neq \perp$, and π' matches π in stages u such that $t' \leq u < t$, and* $\textsc{RevealPSS}(\pi, u)$ *and* $\textsc{RevealPSS}(\pi', u)$ *have not been queried.*

5) pss is initialised with a secret value uniformly chosen at random, and $\textsc{RevealPSS}(\pi, 0)$ *and* $\textsc{RevealPSS}(\pi', 0)$ *have not been queried where π' matches π in stage u such that $0 < u < t$, and* $\textsc{RevealPSS}(\pi, u)$ *and* $\textsc{RevealPSS}(\pi', u)$ *have not been queried.*

d) If there exists no instance π' that is an origin instance of π in stage t, then at least one of the following conditions holds:

1) $\textsc{RevealLCK}(\pi.\mathsf{pid})$ *has not been queried before π accepts in stage t, where there exists no instance that shares the same* cid *as π in stage t.*

2) $\textsc{RevealECK}(\pi, t)$ *and* $\textsc{RevealECK}(\pi'', t)$ *have not been queried before π accepts in stage t, where $\pi.\mathsf{cid}[t] = \pi''.\mathsf{cid}[t]$.*

3) $\textsc{RevealECK}(\pi, t')$ *and* $\textsc{RevealECK}(\pi'', t')$ *have not been queried where π'' matches π in stages u such that $t' \leq u < t$, and* $\textsc{RevealPSS}(\pi, u)$ *and* $\textsc{RevealPSS}(\pi'', u)$ *have not been queried before π accepts in stage t.*

4) $\textsc{RevealEQK}(\pi, t')$ *and* $\textsc{RevealEQK}(\pi'', t')$ *have not been queried where $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi''.\mathsf{eqk}[t'] \neq \perp$, and π'' matches π in stages u such that $t' \leq u < t$, and* $\textsc{RevealPSS}(\pi, u)$ *and* $\textsc{RevealPSS}(\pi'', u)$ *have not been queried before π accepts in stage t.*

5) pss is initialised with a secret value uniformly chosen at random, and $\textsc{RevealPSS}(\pi, 0)$ *and* $\textsc{RevealPSS}(\pi'', 0)$ *have not been queried where π'' matches π in stage u such that $0 < u < t$, and* $\textsc{RevealPSS}(\pi, u)$ *and* $\textsc{RevealPSS}(\pi'', u)$ *have not been queried before π accepts in stage t.*

e) If there exists an instance π' that is an origin instance of π in stage t (and π is not an origin instance of π'), then at least one of the following conditions holds:

1) π *is an initiator's instance and* RevealLCK(π.pid) *has not been queried before* π *accepts in stage* t*, or* π *is a responder's instance and* RevealLCK(π.pid) *has not been queried before* π *accepts in stage* t *and* π *is not an origin instance of* π' *when* π *computes the last message* (m_4) *of the run.*

2) RevealECK(π, t) *and* RevealECK(π', t) *have not been queried, where* π.cid$[t] = \pi'$.cid$[t]$.

3) RevealEQK(π, t) *and* RevealEQK(π', t) *have not been queried, where* π.eqk$[t] \neq \perp$ *and* π'.eqk$[t] \neq \perp$.

4) RevealECK(π, t') *and* RevealECK(π', t') *have not been queried where* π' *matches* π *in stages* u *such that* $t' \leq u < t$*, and* RevealPSS(π, u) *and* RevealPSS(π', u) *have not been queried.*

5) RevealEQK(π, t') *and* RevealEQK(π', t') *have not been queried where* π.eqk$[t'] \neq \perp$ *and* π'.eqk$[t'] \neq \perp$*, and* π' *matches* π *in stages* u *such that* $t' \leq u < t$*, and* RevealPSS(π, u) *and* RevealPSS(π', u) *have not been queried.*

6) pss is initialised with a secret value uniformly chosen at random, and RevealPSS($\pi, 0$) *and* RevealPSS($\pi', 0$) *have not been queried where* π' *matches* π *in stage* u *such that* $0 < u < t$*, and* RevealPSS(π, u) *and* RevealPSS(π', u) *have not been queried.*

Condition a) of FreshKey forbids the adversary from compromising the session key of the targeted instance π in stage t. Condition b) forbids the adversary from compromising the session key of any instance π' partnered with the targeted instance in stage t. Condition c) considers the context when π has (almost) a matching instance. But it may be possible that the last message of a run (m_4) be not received. In such a case, the two instances do not match. We capture this possibility in case c) so that the restrictions related to the oracles apply also in the latter scenario (missing m_4). Hence, case c) says that, as long as one out of the three main secret values (eck, eqk, pss) of the stage remains uncompromised, then so is the session key. This condition (second and fourth subcases) takes into account the fact that there may be no quantum shared secret computed during the stage. Condition d) considers the case when the adversary wants to impersonate an honest party. To be successful, it needs to correctly authenticate to the victim and to get the internal state pss stored by the victim, and get the ephemeral KEM shared secret ss, if the victim started a KEM exchange with an honest instance (since pss and ss are used to compute intermediary symmetric keys and also the session key). As long as the adversary does not compromise either a valid long-term asymmetric private key (case d)1)), the KEM shared secret (case d)2)) or the victim's internal state (cases d)3) to d)5)), then the targeted instance does not accept (and does not compute the stage's session key). Condition e) corresponds to the remaining case: π has an origin instance but no matching instance. We show that (in almost all subcases) the tested instance eventually never accepts because it does not share the same transcript of messages as its origin instance.

Case d)1) requires that π.cid$[t]$ is not shared with any other honest instance. This restriction is necessary as otherwise an attack scenario enables the adversary to successfully win the experiment. The scenario is the following. The

adversary executes a run between the tested instance π and another honest instance (let us call it $\tilde{\pi}$) such that the two instances do *not* share pss at beginning of stage t. The adversary can query any oracle but $\text{REVEALLCK}(\pi.\mathsf{pid}, t)$ (until π accepts). Therefore, with oracle queries, the adversary is able to get the secret values ss and pss necessary to compute the AE parameters computed on the one hand by π and, on the other hand, by $\tilde{\pi}$. The latter computes in particular a signature that is valid for π. The messages, decrypted and encrypted anew by the adversary, are valid for π which eventually accepts (since the signature computed by $\tilde{\pi}$ is indeed correct). In such a case, π has no origin instance. The adversary can also query $\text{REVEALEQK}(\pi, t)$ and finally compute the session keys. Consequently, the adversary trivially wins the security experiment. In this scenario, the two instances agree on their respective parent party's identity, do not share the same session keys, and the adversary knows their respective session keys (therefore it can access and relay the application data exchanged by the two instances). The other subcases in case d) forbid respectively the adversary from revealing either the KEM shared secret ss or the internal state pss.

The $\mathsf{FreshKey}$ predicate captures forward secrecy by allowing the adversary to compromise the long-term asymmetric private key of the targeted instance's peer (upon acceptance). It is captured by subcases d)1) and e)1). It is also captured by subcases c), d)2) to d)5) and e)2) to e)6) since, in the latter subcases, revealing the long-term asymmetric key of the peer is allowed at any point in time. This property guarantees that previous session keys remain uncompromised. The predicate captures also post-compromise security (PCS) [38] by allowing the adversary to compromise all the secret values of an instance as long as there has been at least one uncompromised secret value at some previous point in time such that the current session key "inherits" from the secrecy of this uncompromised value. This ensures that, upon compromising, security of the *attacked session* can still be recovered. We stress that, in our security model, PCS aims at protecting a given session established between two parties u and v but it does not necessarily preserves any other session established between the two same parties. Indeed, if the long-term private key of party u is disclosed, then the adversary can start and successfully complete a new (parallel) session with v (if pss is initialised with a public value). That is, our protocol does not guarantee *conversation PCS* (taking into account all sessions established between the two same parties u and v) but only *session PCS* [40, 41] (considering a given session, across its possibly multiple stages, established between u and v). That said, conversation PCS cannot be achieved in general if the long-term private key is disclosed [41].

$\mathsf{FreshKey}(\pi, t)$
if $((\pi, t) = (\perp, \perp))$ then
  return $\perp$
if $(\pi.\mathsf{status}[t] \neq \texttt{accepted})$ then
  return $\perp$

// *subcase a)*
if ($\pi.\overline{\mathsf{sk}}[t] = \texttt{revealed}$) then
  return `false`
// *subcase b)*
$\forall\ \pi' \neq \pi \mid \mathsf{match}(\pi, \pi', t) = \texttt{true}$, if $\left(\pi'.\overline{\mathsf{sk}}[t] = \texttt{revealed}\right)$ then
  return `false`
// *subcase c)*
if $(\exists\ \pi' \neq \pi \mid (\pi.\mathsf{role} = \texttt{initiator}\ \wedge\ \mathsf{match}(\pi, \pi', t) = \texttt{true}) \vee$
$(\pi.\mathsf{role} = \texttt{responder}\ \wedge\ \mathsf{origin}(\pi, \pi', t) = \texttt{true}\ \wedge\ \mathsf{origin}(\pi', \pi, t) = \texttt{true}))$ then
  if ([[// *subcase c)1)*
    $\pi'.\overline{\mathsf{eck}}[t] = \pi.\overline{\mathsf{eck}}[t] = \texttt{clean}$
    $\vee$
    // *subcase c)2)*
    $\pi.\mathsf{eqk}[t] \neq \perp \wedge\ \pi'.\mathsf{eqk}[t] \neq \perp\ \wedge\ \pi'.\overline{\mathsf{eqk}}[t] = \pi.\overline{\mathsf{eqk}}[t] = \texttt{clean}$
    $\vee$
    // *subcase c)3)*
    $\exists\ t', t' < t \mid (\pi.\overline{\mathsf{eck}}[t'] = \pi'.\overline{\mathsf{eck}}[t'] = \texttt{clean} \wedge$
    $\forall\ u, t' \leq u < t,$
    $(\mathsf{match}(\pi, \pi', u) = \texttt{true} \wedge$
    $\pi.\overline{\mathsf{pss}}[u] = \pi'.\overline{\mathsf{pss}}[u] = \texttt{clean}))$
    $\vee$
    // *subcase c)4)*
    $\exists\ t', t' < t \mid (\pi.\mathsf{eqk}[t'] \neq \perp \wedge\ \pi'.\mathsf{eqk}[t'] \neq \perp \wedge$
    $\pi.\overline{\mathsf{eqk}}[t'] = \pi'.\overline{\mathsf{eqk}}[t'] = \texttt{clean} \wedge$
    $\forall\ u, t' \leq u < t,$
    $(\mathsf{match}(\pi, \pi', u) = \texttt{true} \wedge$
    $\pi.\overline{\mathsf{pss}}[u] = \pi'.\overline{\mathsf{pss}}[u] = \texttt{clean}))$
    $\vee$
    // *subcase c)5)*
    ($\pi.\mathsf{pss}[0]$ and $\pi'.\mathsf{pss}[0]$ are set to a uniformly random value $\wedge$
    $\pi.\overline{\mathsf{pss}}[0] = \pi'.\overline{\mathsf{pss}}[0] = \texttt{clean} \wedge$
    $\forall\ u, 0 < u < t,$
    $(\mathsf{match}(\pi, \pi', u) = \texttt{true} \wedge$
    $\pi.\overline{\mathsf{pss}}[u] = \pi'.\overline{\mathsf{pss}}[u] = \texttt{clean}))$
  ]] $= \texttt{false}$) then
  return `false`
// *subcase d)*
if $(\forall\ \pi' \neq \pi, \mathsf{origin}(\pi', \pi, t) = \texttt{false})$ then
  if ([[// *subcase d)1)*
    $\pi.\mathsf{t}_{\mathsf{acc}}[t] < \overline{\mathsf{lck}_{\pi.\mathsf{pid}}}\ \wedge\ \nexists\ \pi'' \neq \pi \mid \pi''.\mathsf{cid}[t] = \pi.\mathsf{cid}[t]$
    $\vee$
    // *subcase d)2)*
    $\exists\ \pi'' \neq \pi \mid (\pi.\mathsf{cid}[t] = \pi''.\mathsf{cid}[t] \wedge$
    $\pi.\overline{\mathsf{eck}}[t'] = \pi''.\overline{\mathsf{eck}}[t'] = \texttt{clean}$ until $\pi.\mathsf{status}[t] = \texttt{accepted})$
    $\vee$
    // *subcase d)3)*
    $\exists\ t', t' < t, \pi'' \mid (\pi.\overline{\mathsf{eck}}[t'] = \pi''.\overline{\mathsf{eck}}[t'] = \texttt{clean} \wedge$
    $\forall\ u, t' \leq u < t,$
    $(\mathsf{match}(\pi'', \pi, u) = \texttt{true} \wedge$
    $\pi.\overline{\mathsf{pss}}[u] = \pi''.\overline{\mathsf{pss}}[u] = \texttt{clean}))$ until $\pi.\mathsf{status}[t] = \texttt{accepted}$
    $\vee$

```
        // subcase d)4)
        ∃ t′, t′ < t, π″ | (π.eqk[t′] ≠⊥ ∧ π″.eqk[t′] ≠⊥ ∧
                            ((π.ēqk[t′] = π″.ēqk[t′] = clean ∧
                              ∀ u, t′ ≤ u < t,
                                 (match(π″, π, u) = true ∧
                                  π.p̄ss[u] = π″.p̄ss[u] = clean)) until π.status[t] = accepted))
        ∨
        // subcase d)5)
        ∃ π″ | (π.pss[0] and π″.pss[0] are set to a uniformly random value ∧
                ((π.p̄ss[0] = π″.p̄ss[0] = clean ∧
                  ∀ u, 0 < u < t,
                     (match(π, π″, u) = true ∧
                      π.p̄ss[u] = π″.p̄ss[u] = clean)) until π.status[t] = accepted))
      ]] = false) then
      return false
// subcase e)
if (∃ π′ | origin(π′, π, t) = true ∧ origin(π, π′, t) = false) then
   if ([[// subcase e)1)
        (π.t_acc[t] < l̄ck_{π.pid} ∧
         (π.role = initiator
          ∨
          (π.role = responder ∧
           origin(π, π′, t) = false when π computes the last message (m4))))
        ∨
        // subcase e)2)
        π.cid[t] = π′.cid[t]) ∧ π.ēck[t] = π′.ēck[t] = clean
        ∨
        // subcase e)3)
        π.eqk[t] ≠⊥ ∧ π′.eqk[t] ≠⊥ ∧ π′.ēqk[t] = π.ēqk[t] = clean
        ∨
        // subcase e)4)
        ∃ t′, t′ < t | (π.ēck[t′] = π′.ēck[t′] = clean ∧
                        ∀ u, t′ ≤ u < t,
                           (match(π, π′, u) = true ∧ π.p̄ss[u] = π′.p̄ss[u] = clean))
        ∨
        // subcase e)5)
        ∃ t′, t′ < t | (π.eqk[t′] ≠⊥ ∧ π′.eqk[t′] ≠⊥ ∧
                        π.ēqk[t′] = π′.ēqk[t′] = clean ∧
                        ∀ u, t′ ≤ u < t,
                           (match(π, π′, u) = true ∧ π.p̄ss[u] = π′.p̄ss[u] = clean))
        ∨
        // subcase e)6)
        (π.pss[0] and π′.pss[0] are set to a uniformly random value ∧
         π.p̄ss[0] = π′.p̄ss[0] = clean ∧
         ∀ u, 0 < u < t,
            (match(π, π′, u) = true ∧ π.p̄ss[u] = π′.p̄ss[u] = clean))
      ]] = false) then
      return false
return true
```

Fig. 8: FreshKey predicate

In contrast with the FreshKey predicate, the FreshIdentity predicate is related to a given instance π (associated to a virtual identity) and all its stages since the same (hidden) identity is involved amongst all stages.

Definition 6 (FreshIdentity)**.** *An instance π is fresh with respect to the identity (π is "id-fresh") in the security experiment defined in Figure 5 if the following conditions hold for any stage t:*

a) If there exists an instance π' that shares the same cid *as π in stage t, then at least one of the following conditions holds:*
  1) $\text{RevealECK}(\pi, t)$ *and* $\text{RevealECK}(\pi', t)$ *have not been queried.*
  2) $\text{RevealECK}(\pi, t')$ *and* $\text{RevealECK}(\pi', t')$ *have not been queried where π' matches π in stages u such that $t' \leq u < t$, and* $\text{RevealPSS}(\pi, u)$ *and* $\text{RevealPSS}(\pi', u)$ *have not been queried.*
  3) $\text{RevealEQK}(\pi, t')$ *and* $\text{RevealEQK}(\pi', t')$ *have not been queried where $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi'.\mathsf{eqk}[t'] \neq \perp$, and π' matches π in stages u such that $t' \leq u < t$, and* $\text{RevealPSS}(\pi, u)$ *and* $\text{RevealPSS}(\pi', u)$ *have not been queried.*
  4) pss is initialised with a secret value uniformly chosen at random, and $\text{RevealPSS}(\pi, 0)$ *and* $\text{RevealPSS}(\pi', 0)$ *have not been queried where π' matches π in stage u such that $0 < u < t$, and* $\text{RevealPSS}(\pi, u)$ *and* $\text{RevealPSS}(\pi', u)$ *have not been queried.*

b) If there exists no instance π' that shares the same cid *as π in stage t, then at least one of the following conditions holds:*
  1) $\text{RevealLCK}(\pi.\mathsf{pid})$ *has not been queried before π accepts in stage t, where π is responder.*
  2) $\text{RevealECK}(\pi, t')$ *and* $\text{RevealECK}(\pi'', t')$ *have not been queried where π'' matches π in stages u such that $t' \leq u < t$, and* $\text{RevealPSS}(\pi, u)$ *and* $\text{RevealPSS}(\pi'', u)$ *have not been queried.*
  3) $\text{RevealEQK}(\pi, t')$ *and* $\text{RevealEQK}(\pi'', t')$ *have not been queried where $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi''.\mathsf{eqk}[t'] \neq \perp$, and π'' matches π in stages u such that $t' \leq u < t$, and* $\text{RevealPSS}(\pi, u)$ *and* $\text{RevealPSS}(\pi'', u)$ *have not been queried.*
  4) pss is initialised with a secret value uniformly chosen at random, and $\text{RevealPSS}(\pi, 0)$ *and* $\text{RevealPSS}(\pi'', 0)$ *have not been queried where π'' matches π in stage u such that $0 < u < t$, and* $\text{RevealPSS}(\pi, u)$ *and* $\text{RevealPSS}(\pi'', u)$ *have not been queried.*

Condition a) in the FreshIdentity predicate is similar to condition c) in the FreshKey predicate. But, here, we do not require that the two instances π and π' match in stage t. It is enough if they are involved in the ephemeral KEM exchange. We translate the latter through their $\mathsf{cid} = epk \| ct$ being equal.[8] More

[8] In fact, this condition is slightly stronger than necessary. More precisely, let π be the targeted instance. If $\pi.\mathsf{role} = \texttt{initiator}$, then π and π' must indeed share $\mathsf{cid} = epk \| ct$. This ensures that ct is computed by π' *with* epk. Note that, if we restrict the condition to the equality of epk only, then we would have to involve

precisely, condition a) states that, as long as one out of the two secret values eck and pss used during the stage remains uncompromised, then the intermediary symmetric keys used to encrypt the certificates cannot be computed (hence the certificates cannot be decrypted). Here, we do not consider the quantum shared secret eqk computed during the current stage as the intermediary symmetric keys do not depend on the latter. This condition takes into account the fact that there may be no quantum shared secret computed during previous stages (subcase a)3)). Condition b) considers the scenario where the adversary interacts during the ephemeral KEM exchange with π. Therefore, the adversary knows the ephemeral KEM shared secret ss. However, as long as the internal secret state remains uncompromised the adversary cannot decrypt the certificate sent by π. This condition takes also into account that eqk may not be used during previous stages (subcase b)3)). Subcase b)1) says that the adversary, behaving as initiator, cannot decrypt π's certificate (even if it compromises all the secret values of the stage) as long as the adversary does not compromise the long-term asymmetric private key of π's putative partner, because the adversary must authenticate first.

The PCS property is captured in the $\mathsf{FreshIdentity}$ predicate, in the same way as in the $\mathsf{FreshKey}$ predicate.

$\underline{\mathsf{FreshIdentity}(\pi)}$
$\forall\ t \in \{1, \ldots, \pi.\mathsf{stid}\},$
  // subcase a)
  if $(\exists\ \pi' \neq \pi \mid \pi.\mathsf{cid}[t] = \pi'.\mathsf{cid}[t])$ then
    if ([[*// subcase a)1)*
      $\pi'.\overline{\mathsf{eck}}[t] = \pi.\overline{\mathsf{eck}}[t] = \texttt{clean}$
      $\vee$
      // subcase a)2)
      $\exists\ t', t' < t \mid (\pi.\overline{\mathsf{eck}}[t'] = \pi'.\overline{\mathsf{eck}}[t'] = \texttt{clean} \wedge$
        $\forall\ u, t' \leq u < t,$
          $(\mathsf{match}(\pi', \pi, u) = \texttt{true}\ \wedge\ \pi.\overline{\mathsf{pss}}[u] = \pi'.\overline{\mathsf{pss}}[u] = \texttt{clean}))$
      $\vee$
      // subcase a)3)
      $\exists\ t', t' < t \mid (\pi.\mathsf{eqk}[t'] \neq \perp \wedge \pi'.\mathsf{eqk}[t'] \neq \perp \wedge$
        $\pi.\overline{\mathsf{eqk}}[t'] = \pi'.\overline{\mathsf{eqk}}[t'] = \texttt{clean} \wedge$
        $\forall\ u, t' \leq u < t,$
          $(\mathsf{match}(\pi', \pi, u) = \texttt{true}\ \wedge\ \pi.\overline{\mathsf{pss}}[u] = \pi'.\overline{\mathsf{pss}}[u] = \texttt{clean}))$
      $\vee$

an additional property related to the KEM (such as $MAL\text{-}BIND\text{-}CT\text{-}PK$ [39]) in order to ensure that the KEM ciphertext ct received by π is related to the shared public key epk. On the other hand, if $\pi.\mathsf{role} = \texttt{responder}$, then π and π' must only share epk. In that case, we know that ct (sent by π) is computed by π. For simplicity, however, we define this condition upon the equality of the whole cid.

// *subcase a)4)*
($\pi.\mathsf{pss}[0]$ and $\pi'.\mathsf{pss}[0]$ are set to a uniformly random value $\wedge$
$\pi.\overline{\mathsf{pss}}[0] = \pi'.\overline{\mathsf{pss}}[0] = \texttt{clean} \wedge$
$\forall\, u, 0 < u < t,$
$(\mathsf{match}(\pi, \pi', u) = \texttt{true} \ \wedge \ \pi.\overline{\mathsf{pss}}[u] = \pi'.\overline{\mathsf{pss}}[u] = \texttt{clean}))$
$]] = \texttt{false}$) then
return `false`
// *subcase b)*
if ($\forall\, \pi' \neq \pi, \pi.\mathsf{cid}[t] \neq \pi'.\mathsf{cid}[t]$) then
if ([[// *subcase b)1)*
$\pi.\mathsf{t}_{\mathsf{acc}}[t] < \overline{\mathsf{lck}_{\pi.\mathsf{pid}}} \ \wedge \ \pi.\mathsf{role} = \texttt{responder}$
$\vee$
// *subcase b)2)*
$\exists\, t', t' < t, \pi'' \mid (\pi.\overline{\mathsf{eck}}[t'] = \pi''.\overline{\mathsf{eck}}[t'] = \texttt{clean} \wedge$
$\forall\, u, t' \leq u < t,$
$(\mathsf{match}(\pi, \pi'', u) = \texttt{true} \ \wedge \ \pi.\overline{\mathsf{pss}}[u] = \pi''.\overline{\mathsf{pss}}[u] = \texttt{clean}))$
$\vee$
// *subcase b)3)*
$\exists\, t', t' < t, \pi'' \mid (\pi.\mathsf{eqk}[t'] \neq \bot \wedge \pi''.\mathsf{eqk}[t'] \neq \bot \ \wedge$
$\pi.\overline{\mathsf{eqk}}[t'] = \pi''.\overline{\mathsf{eqk}}[t'] = \texttt{clean} \wedge$
$\forall\, u, t' \leq u < t,$
$(\mathsf{match}(\pi, \pi'', u) = \texttt{true} \ \wedge \ \pi.\overline{\mathsf{pss}}[u] = \pi''.\overline{\mathsf{pss}}[u] = \texttt{clean}))$
$\vee$
// *subcase b)4)*
$\exists\, \pi'' \mid (\pi.\mathsf{pss}[0]$ and $\pi''.\mathsf{pss}[0]$ are set to a uniformly random value $\wedge$
$\pi.\overline{\mathsf{pss}}[0] = \pi''.\overline{\mathsf{pss}}[0] = \texttt{clean} \wedge$
$\forall\, u, 0 < u < t,$
$(\mathsf{match}(\pi, \pi'', u) = \texttt{true} \ \wedge \ \pi.\overline{\mathsf{pss}}[u] = \pi''.\overline{\mathsf{pss}}[u] = \texttt{clean}))$
$]] = \texttt{false}$) then
return `false`
return `true`

Fig. 9: FreshIdentity predicate

Formal Definition of the Security Properties. We define below the four main security properties captured by our security model.

Definition 7 (Soundness). *An adversary $\mathcal{A}$ against a protocol Π breaks the soundness of Π if the predicate* Sound *is false. The adversary's advantage is defined as its winning probability:*

$$\mathsf{adv}^{\mathsf{sound}}_{\Pi}(\mathcal{A}) = \Pr\left[\mathsf{Sound} = \texttt{false}\right].$$

Definition 8 (Explicit Authentication). *An instance π of a protocol Π is said to have* accepted maliciously *in stage t during the explicit authentication security experiment if this instance in stage t renders the predicate* ExplicitAuth

false. That is,

$$\exists\ \pi, t \in \{1, \ldots, \pi.\mathsf{stid}\} \mid \begin{cases} \mathsf{FreshAuth}(\pi, t) = \mathtt{true} \\ \wedge \\ \forall\ \pi',\ (\pi.\mathsf{pid} \neq \pi'.\mathsf{parent} \vee \\ \qquad \pi.\mathsf{cid}[t] \neq \pi'.\mathsf{cid}[t] \vee \\ \qquad \pi.\mathsf{pss}[t-1] \neq \pi'.\mathsf{pss}[t-1] \vee \\ \qquad \pi.\mathsf{role} = \pi'.\mathsf{role} \vee \\ \qquad (\pi'.\mathsf{status}[t] = \mathtt{accepted} \wedge \\ \qquad\ (\pi'.\mathsf{pid} \neq \pi.\mathsf{parent} \ \vee \ \pi'.\mathsf{pss}[t] \neq \pi.\mathsf{pss}[t]))) \end{cases}$$

The adversary's advantage is defined as its winning probability:

$$\mathsf{adv}_{\Pi}^{\mathsf{auth}}(\mathcal{A}) = \Pr\left[\mathsf{ExplicitAuth} = \mathtt{false}\right].$$

Definition 9 (Key Indistinguishability). *An adversary $\mathcal{A}$ against a protocol Π, that issues its* Test*-query to instance $\pi_{\mathsf{test}} = \pi^*$ in stage $t_{\mathsf{test}} = t^*$ during the key indistinguishability security experiment,* answers the Test-challenge correctly *if it terminates with output b^*, such that*

(a) the predicate $\mathsf{FreshKey}(\pi^*, t^*)$ is true, and
(b) $b_{\mathsf{key}} = b^*$.

The adversary's advantage is defined as

$$\mathsf{adv}_{\Pi}^{\mathsf{key\text{-}ind}}(\mathcal{A}) = \left|\Pr\left[b_{\mathsf{key}} = b^* \mid \mathsf{FreshKey}(\pi_{\mathsf{test}}, t_{\mathsf{test}}) = \mathtt{true}\right] - \frac{1}{2}\right|.$$

Definition 10 (Identity Privacy). *An adversary $\mathcal{A}$ against a protocol Π breaks the identity privacy of Π if it terminates with output b^*, such that*

(a) for each $\pi \in \mathcal{I}$, the predicate $\mathsf{FreshIdentity}(\pi)$ is true, and
(b) $b_{\mathsf{id}} = b^*$.

The adversary's advantage is defined as

$$\mathsf{adv}_{\Pi}^{\mathsf{id\text{-}priv}}(\mathcal{A}) = \left|\Pr\left[b_{\mathsf{id}} = b^* \mid \forall\ \pi \in \mathcal{I}, \mathsf{FreshIdentity}(\pi) = \mathtt{true}\right] - \frac{1}{2}\right|.$$

Definition 11 (AKE Security). *Let Π be a two-party AKE protocol and $G^{\mathsf{sec}}_{\Pi,\mathcal{A}}$ the security experiments defined in Figure 5 and Figure 6, and executed by an adversary $\mathcal{A}$. Π is a secure AKE protocol if, for all polynomial time (quantum) adversary $\mathcal{A}$, $\mathsf{adv}_{\Pi}^{\mathsf{sound}}(\mathcal{A})$, $\mathsf{adv}_{\Pi}^{\mathsf{auth}}(\mathcal{A})$, $\mathsf{adv}_{\Pi}^{\mathsf{key\text{-}ind}}(\mathcal{A})$ and $\mathsf{adv}_{\Pi}^{\mathsf{id\text{-}priv}}(\mathcal{A})$ are negligible functions of the security parameter.*

4 Building Blocks

In this section, we present the building blocks that our protocol is made of and the properties their security relies upon.

4.1 Chosen Plaintext and Ciphertext Attacks of Key Encapsulation Mechanisms

We recall the definition of a key encapsulation mechanism and the standard notion of indistinguishability under chosen-ciphertext attack. Following Jager, Kiltz, Riepel, Schäge [73], we define the collision probability of KEM public keys and ciphertexts respectively.

Definition 12 (Key Encapsulation Mechanism). *A key encapsulation mechanism (KEM)* $\mathsf{KEM} = (\mathsf{Kgen}, \mathsf{Encaps}, \mathsf{Decaps})$ *consists of three efficient algorithms defined as follows:*

- $(pk, sk) \xleftarrow{\$} \mathsf{Kgen}()$. *This probabilistic algorithm generates a public encapsulation key* pk *and a private decapsulation key* sk.
- $(ss, ct) \xleftarrow{\$} \mathsf{Encaps}(pk)$. *On input a public key* pk, *this probabilistic algorithm outputs a key* $ss \in \mathcal{K}$ *and a ciphertext* ct.
- $ss \leftarrow \mathsf{Decaps}(ct, sk)$. *On input a ciphertext* ct *and a decapsulation key* sk, *this deterministic algorithm outputs either a key (in* $\mathcal{K}$*) or a special symbol* $\perp$ *indicating rejection.*

The KEM KEM *is* δ*-correct if*

$$\Pr\Big[\mathsf{KEM.Decaps}(ct, sk) \neq ss \mid (pk, sk) \xleftarrow{\$} \mathsf{KEM.Kgen}() \wedge (ss, ct) \xleftarrow{\$} \mathsf{KEM.Encaps}(pk)\Big] \leq \delta.$$

By μ *we denote the collision probability of the key generation algorithm* Kgen. *In particular*

$$\Pr\Big[pk = pk' \mid (pk, sk) \xleftarrow{\$} \mathsf{KEM.Kgen}(), (pk', sk') \xleftarrow{\$} \mathsf{KEM.Kgen}()\Big] \leq 2^{-\mu}.$$

We denote the min-entropy of the encapsulation algorithm Encaps *by* $\gamma(pk) = -\log_2\Big(\max_{ct} \Pr[ct = \mathsf{KEM.Encaps}(pk)]\Big)$. *We say that* KEM *is* γ*-spread if for all* $(pk, sk) \xleftarrow{\$} \mathsf{KEM.Kgen}()$, $\gamma(pk) \geq \gamma$. *This implies that for all ciphertext* ct

$$\Pr[ct = \mathsf{KEM.Encaps}(pk)] \leq 2^{-\gamma}.$$

To define the security of the KEM, we rely upon the usual notion of IND-CPA security but also on IND-1CCA security introduced by Schwabe, Stebila, Wiggers [105] and further investigated by e.g. Huguenin-Dumittan, Vaudenay [67] and Jiang, Ma, Zhang [74].

In a typical AKE model, when it comes to the KEM security, the KEM public key and ciphertext are replaced by the challenge given to the reduction. However, if the honest instance receives the challenge public key, the adversary might well send its own ciphertext to the putative partner instance. Consequently, the reduction must be able to decapsulate the ciphertext without knowledge of the

(challenge) private key. Hence the need for the reduction to query a decapsulation oracle. That said, if the KEM is ephemeral, then IND-CCA may not be necessary because the putative partner instance will decapsulate at most once with the same private key. IND-CCA can therefore be restricted to at most one query to the decapsulation oracle which yields the IND-1CCA notion.

The original motive for such a notion is to avoid the cost of re-encryption due to the Fujisaki-Okamoto [55] transform that many post-quantum KEM implement to get IND-CCA security. Overall, the notion of IND-1CCA security allows using more efficient KEM schemes which remain secure in the context of deployment without demanding strong mitigations against side-channel attacks.

$G^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{A}}$

INITIALISE	FINALISE(b^*)
$b \xleftarrow{\$} \{0,1\}$	return $[\![b = b^*]\!]$
$(pk, sk) \xleftarrow{\$} \mathsf{Kgen}()$	
$(ss_0, ct) \xleftarrow{\$} \mathsf{Encaps}(pk)$	
$ss_1 \xleftarrow{\$} \mathcal{K}$	
return pk, ct, ss_b	

Fig. 10: Indistinguishability under chosen-plaintext attack (IND-CPA) of KEMs.

Definition 13 (KEM IND-CPA security). *Let* KEM *be a KEM and* $G^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{A}}$ *be the experiment for indistinguishability under chosen-plaintext attack of KEMs, described by Figure 10. For an adversary* $\mathcal{A}$ *we define*

$$\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} = \left|2 \cdot \Pr\left[G^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM},\mathcal{A}} \Rightarrow 1\right] - 1\right|$$

where $\mathcal{A}$ *runs in time at most* t*.*

Definition 14 (KEM IND-1CCA security). *Let* KEM *be a KEM and* $G^{\mathsf{IND\text{-}1CCA}}_{\mathsf{KEM},\mathcal{A}}$ *be the experiment for indistinguishability under chosen-ciphertext attack of KEMs restricted to one decapsulation query at most, described by Figure 11. For an adversary* $\mathcal{A}$ *we define*

$$\mathsf{adv}^{\mathsf{ind\text{-}1cca}}_{\mathsf{KEM}}(q_{\mathrm{D}}) = \left|2 \cdot \Pr\left[G^{\mathsf{IND\text{-}1CCA}}_{\mathsf{KEM},\mathcal{A}} \Rightarrow 1\right] - 1\right|$$

where $\mathcal{A}$ *runs in time at most* t *and makes at most* $q_{\mathrm{D}} = 1$ *query to the* DECAPS *oracle.*

4.2 Unforgeability of Signatures

We recall the definition of a signature scheme and existential unforgeability under chosen-message attack [60] (EUF-CMA).

$G^{\mathsf{IND\text{-}1CCA}}_{\mathsf{KEM},\mathcal{A}}$

Initialise	Decaps(c)	Finalise(b^*)
$b \xleftarrow{\$} \{0,1\}$	$n \leftarrow n+1$	return $[[b = b^*]]$
$n = 0$	if $(c = ct \ \vee \ n > 1)$ then	
$(pk, sk) \xleftarrow{\$} \mathsf{Kgen}()$	return $\perp$	
$(ss_0, ct) \xleftarrow{\$} \mathsf{Encaps}(pk)$	$r \leftarrow \mathsf{Decaps}(c, sk)$	
$ss_1 \xleftarrow{\$} \mathcal{K}$	return r	
return pk, ct, ss_b		

Fig. 11: Indistinguishability under chosen-ciphertext attack restricted to one decapsulation query at most (IND-1CCA) of KEMs.

Definition 15 (Signature scheme). *A signature scheme* $\mathsf{SIG} = (\mathsf{Kgen}, \mathsf{Sign}, \mathsf{Vrfy})$ *consists of three efficient algorithms defined as follows:*

- $(pk, sk) \xleftarrow{\$} \mathsf{Kgen}()$. *This probabilistic algorithm generates a public verification key* pk *and a private signing key* sk.
- $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$. *On input a signing key* sk *and a message* m*, this (possibly) probabilistic algorithm outputs a signature* σ.
- $d \leftarrow \mathsf{Vrfy}(pk, m, \sigma)$. *On input a verification key* pk*, a message* m*, and a signature* σ*, this deterministic algorithm outputs a decision bit* $d \in \{\mathtt{true}, \mathtt{false}\}$ *(where* $d = \mathtt{true}$ *indicates validity of the signature).*

$G^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG},\mathcal{A}}$

Initialise	Sign(m)	Finalise(m^*, σ^*)
$(pk, sk) \xleftarrow{\$} \mathsf{Kgen}()$	$\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$	$d^* \leftarrow \mathsf{Vrfy}(pk, m^*, \sigma^*)$
$\mathcal{Q} \leftarrow \emptyset$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$	return $[[d^* = \mathtt{true} \wedge$
return pk	return σ	$m^* \notin \mathcal{Q}]]$

Fig. 12: Existential unforgeability (EUF-CMA) of signature schemes.

Definition 16 (Signature EUF-CMA security). *Let* SIG *be a signature scheme and* $G^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG},\mathcal{A}}$ *be the experiment for signature existential unforgeability under chosen-message attack described by Figure 12. For an adversary* $\mathcal{A}$ *we define*

$$\mathsf{adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG}}(q_{\mathrm{SG}}) = \Pr\left[G^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG},\mathcal{A}} \Rightarrow 1\right]$$

where $\mathcal{A}$ *runs in time at most* t *and makes at most* q_{SG} *queries to the* Sign *oracle.*

4.3 Security Properties for Authenticated Encryption Schemes

We present the classical notions of ciphertext unforgeability for authenticated encryption (AE) schemes. We give also the notion of length-hiding AE scheme from Paterson, Ristenpart and Shrimpton [95] that we adapt to the real-or-random setting, the notion of tidy AE scheme from Namprempre, Rogaway and Shrimpton [90] that we adapt to the length-hiding setting, and the notion of key-committing AE scheme from Bellare and Hoang [13].

Definition 17 (AE scheme). *An authenticated encryption scheme (AE)* $\mathsf{AE} = (\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of three efficient algorithms defined as follows.*

- $K \xleftarrow{\$} \mathsf{Kgen}()$. *This probabilistic algorithm generates a key* K.
- $c \leftarrow \mathsf{Enc}(K, [\ell], N, H, m)$. *On input a key* K*, an (optional) length* ℓ *of the ciphertext, a nonce* N*, a header* H*, and a message* m*, this deterministic algorithm outputs a ciphertext* c.
- $m \leftarrow \mathsf{Dec}(K, N, H, c)$. *On input a key* K*, a nonce* N*, a header* H*, and a ciphertext* c*, this deterministic algorithm outputs either a plaintext message or a failure symbol* $\perp$.

The scheme AE *is correct if* $c = \mathsf{Enc}(K, [\ell], N, H, m)$ *implies that* $m = \mathsf{Dec}(K, N, H, c)$. *The scheme* AE *is tidy if* $m = \mathsf{Dec}(K, N, H, c)$ *implies that* $c = \mathsf{Enc}(K, [\ell], N, H, m)$ *(with* $\ell = |c|$*).*

Remark 4. Since Enc is a deterministic algorithm, the padding that is internally computed by Enc depends only on the inputs K, ℓ, N, H, m (i.e. the padding is a deterministic function of these inputs). We will leverage the latter in the proofs of the key indistinguishability property (case e) of predicate $\mathsf{FreshKey}$).

Definition 18 (AE $\mathsf{INT\text{-}CTXT}$ security). *Let* AE *be an AE scheme and* $G^{\mathsf{INT\text{-}CTXT}}_{\mathsf{AE},\mathcal{A}}$ *be the experiment for the AE integrity of ciphertexts described by Figure 13. For an adversary* $\mathcal{A}$*, we define*

$$\mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(q_{\mathrm{E}}, q_{\mathrm{D}}) = \Pr\left[G^{\mathsf{INT\text{-}CTXT}}_{\mathsf{AE},\mathcal{A}} \Rightarrow 1\right]$$

where $\mathcal{A}$ *runs in time at most* t *and makes at most* q_{E} *and* q_{D} *queries to the oracles* Enc *and* Dec *respectively.*

Definition 19 (AE LH security). *Let* AE *be an AE scheme and* $G^{\mathsf{LH}}_{\mathsf{AE},\mathcal{A}}$ *be the experiment for the AE length-hiding security described by Figure 14. For an adversary* $\mathcal{A}$ *we define*

$$\mathsf{adv}^{\mathsf{lh}}_{\mathsf{AE}}(q_{\mathrm{E}}, q_{\mathrm{D}}) = \left|2 \cdot \Pr\left[G^{\mathsf{LH}}_{\mathsf{AE},\mathcal{A}} \Rightarrow 1\right] - 1\right|$$

where $\mathcal{A}$ *runs in time at most* t *and makes at most* q_{E} *and* q_{D} *queries to the oracles* Enc *and* Dec *respectively.*

In the experiment described by Figure 14, the adversary can adaptively choose the ciphertext length ℓ.

$G^{\mathsf{INT\text{-}CTXT}}_{\mathsf{AE},\mathcal{A}}$

Initialise	$\textsc{Enc}([\ell], N, H, m)$	$\textsc{Dec}(N, H, c)$
$\mathcal{C} \leftarrow \emptyset$	if $N \in \mathcal{N}$ then	if $(N, H, c) \in \mathcal{C}$ then
$\mathcal{N} \leftarrow \emptyset$	return $\bot$	return $\bot$
$K \xleftarrow{\$} \mathsf{Kgen}()$	$c \leftarrow \mathsf{Enc}(K, [\ell], N, H, m)$	$m \leftarrow \mathsf{Dec}(K, N, H, c)$
	$\mathcal{C} \leftarrow \mathcal{C} \cup \{(N, H, c)\}$	return m
	$\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}$	
	return c	$\textsc{Finalise}(N^*, H^*, c^*)$
		$m^* \leftarrow \mathsf{Dec}(K, N^*, H^*, c^*)$
		return $[[m^* \neq \bot \wedge (N^*, H^*, c^*) \notin \mathcal{C}]]$

Fig. 13: Integrity of ciphertexts ($\mathsf{INT\text{-}CTXT}$) of AE schemes.

$G^{\mathsf{LH}}_{\mathsf{AE},\mathcal{A}}$

Initialise	$\textsc{Enc}(\ell, N, H, m)$	$\textsc{Dec}(N, H, c)$
$b \xleftarrow{\$} \{0,1\}$	if $N \in \mathcal{N}$ then	if $(N, H, c) \in \mathcal{C}$ then
$\mathcal{C} \leftarrow \emptyset$	return $\bot$	return $\bot$
$\mathcal{N} \leftarrow \emptyset$	$c_1 \leftarrow \mathsf{Enc}(K, \ell, N, H, m)$	$m \leftarrow \mathsf{Dec}(K, N, H, c)$
$K \xleftarrow{\$} \mathsf{Kgen}()$	$c_0 \xleftarrow{\$} \{0,1\}^{\lvert c_1 \rvert}$	return m
	$\mathcal{C} \leftarrow \mathcal{C} \cup \{(N, H, c_b)\}$	
	$\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}$	$\textsc{Finalise}(b^*)$
	return c_b	return $[[b = b^*]]$

Fig. 14: Length-hiding security (LH) of AE schemes.

$G^{\mathsf{KEY\text{-}CMT}}_{\mathsf{AE},\mathcal{A}}$

$\textsc{Finalise}(c, K, N, H, m, K', N', H', m')$
return $[[m = \mathsf{Dec}(K, N, H, c) \wedge m' = \mathsf{Dec}(K', N', H', c) \wedge m \neq \bot \wedge m' \neq \bot \wedge c \neq \bot \wedge K \neq K']]$

Fig. 15: Key-committing property ($\mathsf{KEY\text{-}CMT}$) of AE schemes.

Definition 20 (AE KEY-CMT security). *Let* AE *be an AE scheme and* $G^{\mathsf{KEY\text{-}CMT}}_{\mathsf{AE},\mathcal{A}}$ *be the experiment for the AE key-committing property described by Figure 15. For an adversary* $\mathcal{A}$*, we define*

$$\mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}} = \Pr\left[G^{\mathsf{KEY\text{-}CMT}}_{\mathsf{AE},\mathcal{A}} \Rightarrow 1\right]$$

where $\mathcal{A}$ *runs in time at most* t*.*

The experiment depicted by Figure 15 corresponds to the CMTD-1 notion of Bellare and Hoang [13]. In our context, this property is used to guarantee that a message encrypted with a key K is not valid (i.e. cannot be correctly decrypted) under a key $K' \neq K$.

4.4 PRF Security

We recall the security notion for pseudo-random functions [12].

Definition 21 (PRF security). *Let* $\mathsf{F} : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^n$*,* $k, n \in \mathbb{N}$*,* $m \in \mathbb{N} \cup \{*\}$*, be a function, and* $G^{\mathsf{PRF}}_{\mathsf{F},\mathcal{A}}$ *be the* PRF *security experiment described by Figure 16. For an adversary* $\mathcal{A}$ *we define*

$$\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{F}}(q_{\mathrm{FC}}) = \left|2 \cdot \Pr\left[G^{\mathsf{PRF}}_{\mathsf{F},\mathcal{A}} \Rightarrow 1\right] - 1\right|$$

where $\mathcal{A}$ *runs in time at most* t *and makes at most* q_{FC} *queries to the* FUNC *oracle.*

$\underline{G^{\mathsf{PRF}}_{\mathsf{F},\mathcal{A}}}$

INITIALISE	FUNC(x)	FINALISE(b^*)
$b \xleftarrow{\$} \{0,1\}$	return $f(x)$	return $[[b = b^*]]$
if $b = 1$ then		
$\quad K \xleftarrow{\$} \{0,1\}^k$		
$\quad f \leftarrow \mathsf{F}(K, \cdot)$		
else		
$\quad f \xleftarrow{\$} \mathcal{F}$		

Fig. 16: PRF security experiment. $\mathcal{F}$ is the space of all functions $\{0,1\}^m \to \{0,1\}^n$.

Below, we define the notion of *swap PRF* and *dual PRF* [10, 14, 15].

Definition 22 (swap-PRF and dual-PRF security). *Let* $\mathsf{F} : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^n$*,* $k, n \in \mathbb{N}$*,* $m \in \mathbb{N} \cup \{*\}$*, be a function family. The swap of* F*, denoted by* $\mathsf{F}^{\mathsf{swap}}$ *is the function family* $\{0,1\}^m \times \{0,1\}^k \to \{0,1\}^n$ *defined by* $\mathsf{F}^{\mathsf{swap}}(x, y) = \mathsf{F}(y, x)$*,* $\forall\ (x, y) \in \{0,1\}^m \times \{0,1\}^k$*. We say that* F *is a swap-PRF if* $\mathsf{F}^{\mathsf{swap}}$ *is a PRF. We say that* F *is a dual-PRF if* F *and* $\mathsf{F}^{\mathsf{swap}}$ *are PRFs.*

4.5 Function Collision Resistance

We define collision resistance of functions as follows.

Definition 23 (Function collision resistance). *Let* $\mathsf{F} : \{0,1\}^* \to \{0,1\}^\ell$, $\ell \in \mathbb{N}$, *be a function. For an adversary* $\mathcal{A}$ *we define*

$$\mathsf{adv}^{\mathsf{cr}}_{\mathsf{F}} = \Pr\left[(m, m') \xleftarrow{\$} \mathcal{A} \mid m \neq m' \wedge \mathsf{F}(m) = \mathsf{F}(m')\right].$$

4.6 2-universal Hash Function

Definition 24 (2-universal hash function). *Let* $\mathcal{H} = \{h_i, 0 \leq i < n\}$ *a family of hash functions where* $h_i : \{0,1\}^* \to \{0,1\}^\lambda$, $\lambda \in \mathbb{N}$, $\forall\, i \in \{0, \ldots, n-1\}$. $\mathcal{H}$ *is said to be* 2*-universal if*

$$\Pr\left[h_i(x) = h_i(x') \mid h_i \xleftarrow{\$} \mathcal{H} \ \wedge \ x \neq x'\right] \leq 2^{-\lambda}.$$

4.7 Security of the QKD Phase

The QKD phase depicted in Figure 1 corresponds to the message flows of the BB84 protocol as described by Tomamichel and Leverrier in [109].

Tomamichel and Leverrier provide a security analysis of the BB84 protocol corresponding to a detailed implementation of the protocol. Their analysis takes into account finite resource effects. Analysis leading to asymptotic security bounds do not allow computing the failure rate for specific (finite-size) parameters of the QKD protocol. Indeed, asymptotic security implies exchanging infinitely many quantum states. On the contrary, the analysis from Tomamichel and Leverrier yields finite security bounds which allow assessing concrete parameters.

However, their analysis considers that the quantum states sent through the quantum channel are single-qubit states. This does not correspond to practical implementations of the BB84 protocol based on imperfect sources of photons (used as qubits). To mitigate attacks leveraging the production of multi-photon states (e.g. the "photon number splitting" (PNS) attack [30, 81]), the default method in practical implementations consists in transmitting additional data (decoy states) to estimate the influence of an adversary on the quantum states (even though, it seems possible, at least theoretically, to defeat this countermeasure [66]).

Tomamichel and Leverrier consider a simplified implementation and therefore exclude the treatment of decoy states in their analysis. The latter is however in line with our security model which does not capture side-channel attacks. Analysis incorporating the case of decoy states, which are more convenient for QKD protocols to be implementable with imperfect hardware, can be found in e.g. Wiesemann, Krause, Tupkary, Lütkenhaus, Rusca, Walenta [115].[9]

[9] Note however that Wiesemann et al. do not consider, in their analysis, side-channel attacks and the "*complications*" they induce.

Tomamichel and Leverrier prove the security of the BB84 protocol based on the following experiment. On the one hand (real protocol), the real BB84 protocol is executed. On the other hand (ideal protocol), a BB84 run is executed and the output key is replaced with a value of same length uniformly drawn at random (more specifically, the "ideal" key is output by a measurement of a quantum state corresponding to a superposition of values with equal amplitudes). Moreover, their analysis relies upon several additional assumptions:

1. Finite-dimensional quantum systems: the quantum degrees of freedom needed by the two parties involved in the protocol can be effectively represented on a finite-dimensional Hilbert space.
2. Sealed laboratories: the two parties are in spatially separated locations and there is an exact control on what information is released by the two parties.
3. Random seeds: the two parties have access to uniformly random seeds.
4. Authenticated communication channel: the two parties share an authenticated public (classical) communication channel.
5. Initiator's preparation: a quantum state does not leak any information about its corresponding basis (i.e. the basis chosen to encode a pseudo-random classical bit into this quantum state). Moreover, the quantum states of the preparation are sufficiently complementary (which implies that any eavesdropping attempt will disturb the quantum states being transmitted).
6. Responder's measurement: the quantum system of the responder (which receives the qubits) can be decomposed into subsystems and its measurement can be represented as operators acting on the individual subsystems. In addition, when the responder makes a measurement with either of the two possible bases, the element corresponding to an inconclusive result coincides for both measurements.

To evaluate the security of the QKD phase, Tomamichel and Leverrier leverage a generalisation of the notion of uncertainty relations (difficulty to predict the outcome of incompatible measurements) formulated in terms of smooth entropies [110]. This notion is an alternative to previous security criteria proposed in the hectic history of QKD security proofs [76, 89, 119]. The distinguishing advantage between the real and ideal protocols can be given from the so-called "diamond distance" which is based on the trace distance (we refer the interested reader to [109] for more details).

In Theorem 1 below, the bound ϵ_{ec} denotes the probability that the error correction test be successful ($T = \mathsf{H}_{\mathsf{ec}}(\hat{X})$) while the two instances do not compute the same quantum key ($K_I \neq K_R$). The bound ϵ_{pe} is related to the probability that the parameter estimation test ($\epsilon_{max} < \epsilon$), based on k random bits chosen among m, be successful even though the fraction of differences between the two n-bit strings X and Y computed by the two communicating parties respectively exceeds ϵ_{max} (by a constant amount ν). The bound ϵ_{pa} caracterises the amount of randomness that can be extracted from the raw quantum key independently of side information disclosed (in particular, the syndrome Z and the digest T of the error correction step). The quantity ϵ_{ec} is the correctness component in

the upper estimate ϵ_Δ of the distance $\Delta_{pe,ec,pa}$ while $\epsilon_s = \epsilon_{pe}(\nu) + \epsilon_{pa}(\nu)$ is its secrecy component.

Parameter	Meaning
$\bar{c} < 1$	Parameter quantifying the overall (average) quality of the initiator's preparation device
m	Number of quantum states (i.e. qubits) shared by the two parties
k	Bit length of the raw quantum key used for parameter estimation (pe)
$n = m - k$	Bit length of the raw quantum key used for distillation (pa)
ϵ_{max}	Threshold for the parameter estimation test (pe)
r	Bit length of the error correction syndrome (ec)
λ	Bit length of the hash output used in the error correction scheme (ec)
ℓ	Bit length of the final quantum key output by the privacy amplification function (pa)

Table 1: Parameters used in Theorem 1

Theorem 1. *Let* $\texttt{qkd_pm}_{pe,ec,pa}$ *be the prepare-and-measure (BB84) protocol described in [109, Section 7] and* $\texttt{qkd_ideal}_{pe,ec,pa}$ *the ideal protocol corresponding to* $\texttt{qkd_pm}_{pe,ec,pa}$ *where the output keys are replaced with independent and uniformly random values of same length as the keys output by a* $\texttt{qkd_pm}_{pe,ec,pa}$ *run. Let* $\Delta_{pe,ec,pa} = \|\texttt{qkd_pm}_{pe,ec,pa} - \texttt{qkd_ideal}_{pe,ec,pa}\|_\diamond$ *be the diamond distance between* $\texttt{qkd_pm}_{pe,ec,pa}$ *and* $\texttt{qkd_ideal}_{pe,ec,pa}$*. We have that*

$$\Delta_{pe,ec,pa} \leq \epsilon_\Delta = \epsilon_{ec} + \inf_{\nu \in [0, \frac{1}{2} - \epsilon_{max}]} (\epsilon_{pe}(\nu) + \epsilon_{pa}(\nu))$$

with

$$\begin{cases} \epsilon_{ec} & = 2^{-\lambda} \\ \epsilon_{pe}(\nu) & = 2 \cdot \exp\left(-\dfrac{(m-k)k^2\nu^2}{m(k+1)}\right) \\ \epsilon_{pa}(\nu) & = \dfrac{1}{2}\sqrt{2^{(m-k)(\log_2(\bar{c}) + h(\epsilon_{max}+\nu)) + r + \lambda + \ell}} \end{cases}$$

where ϵ_{ec}*,* ϵ_{pe}*,* ϵ_{pa} *denote the error functions of the error correction, parameter estimation and privacy amplification phases respectively,* $h(x) = -x \cdot \log_2(x) - (1-x) \cdot \log_2(1-x)$ *denotes the binary entropy of* x*, and the other parameters are described in Table 1.*

Proof. Let $\texttt{qkd_eb}_{pe,ec,pa}$ be the entangled-based version of the QKD protocol described in [109, Section 3]. According to [109, Theorem 3], if $\texttt{qkd_eb}_{pe,ec,pa}$ is ϵ_Δ-secure, then $\texttt{qkd_pm}_{pe,ec,pa}$ is ϵ_Δ-secure. That is, $\Delta^{\texttt{qkd_eb}}_{pe,ec,pa} = \|\texttt{qkd_eb}_{pe,ec,pa} - \texttt{qkd_ideal}_{pe,ec,pa}\|_\diamond \leq \epsilon_\Delta$. According to [109, Lemma 1], we have $\Delta^{\texttt{qkd_eb}}_{pe,ec,pa} \leq \alpha + \beta$, with $\alpha = 2^{-\lambda}$ as per [109, Theorem 2], and $\beta = \inf_{\nu \in [0, \frac{1}{2} - \epsilon_{max}]} (\epsilon_{pe}(\nu) + \epsilon_{pa}(\nu))$ with $\epsilon_{pe}(\nu) = 2 \cdot \exp\left(-\dfrac{(m-k)k^2\nu^2}{m(k+1)}\right)$ and $\epsilon_{pa}(\nu) = \dfrac{1}{2}\sqrt{2^{(m-k)(\log_2(\bar{c}) + h(\epsilon_{max}+\nu)) + r + \lambda + \ell}}$

as per [109, Theorem 3]. Therefore, $\Delta^{\texttt{qkd_eb}}_{pe,ec,pa}$, hence $\Delta_{pe,ec,pa}$, is lower than the indicated bound in Theorem 1. □

In an ideal implementation, $\bar{c}$ is expected to be equal to $\frac{1}{2}$. In such a case, we have $\log_2(\bar{c}) + h(\epsilon_{max} + \nu) < 0$ if $\nu < \frac{1}{2} - \epsilon_{max}$.

From the distance $\Delta_{pe,ec,pa}$, one can derive the probability for an adversary to distinguish between the real and ideal protocols [61, 98]. If $\Delta_{pe,ec,pa} \leq \epsilon_\Delta$, then the maximum probability is equal to $p_\Delta = (1 + \epsilon_\Delta)/2$. In our proofs based on a sequence of games between a challenger and an adversary [17, 107], we will use the probability p_Δ in some transitions between two games (see for instance Section A.3, subcase c)2) in predicate FreshKey). More precisely, in a Game i we consider an execution of the AKE protocol with the "real" QKD phase (corresponding to protocol $\texttt{qkd_pm}_{pe,ec,pa}$). In the next game (Game $i+1$), we substitute the real output quantum key with a uniformly random key (i.e. Game $i+1$ executes $\texttt{qkd_ideal}_{pe,ec,pa}$). As explained above, this corresponds precisely to the experiment used by Tomamichel and Leverrier to prove the security of BB84 and define $\Delta_{pe,ec,pa}$. Therefore, we can upper-bound the difference of advantages related to Game i and Game $i+1$ with the probability p_Δ to distinguish the key swap. Note that, in Game i and Game $i+1$, a quantum key is actually output because the sequence of games occurs in a context where the QKD phase does not abort by assumption (e.g. case FreshKey c)2)). Moreover, the games occur when the targeted instance has a matching instance, hence satisfying assumption 4 above. The other five assumptions (1-3, 5, 6) relate essentially to the material realisation of the QKD phase. We consider that they are fulfilled during the protocol runs.

5 Security Analysis

5.1 Main Theorem

The security analysis of the protocol depicted by Figure 1 shows that the protocol is indeed a secure AKE protocol in any subcase of the predicates FreshAuth, FreshKey and FreshIdentity. That is, the explicit authentication property is ensured if at least the KEM and signature schemes remain secure, or *pss* is uncompromised; key indistinguishability: the KEM scheme is secure, or the QKD phase is safe, or *pss* is uncompromised; identity privacy: the KEM scheme is secure, or *pss* is uncompromised. In turn, *pss* must be either initialised with a value uniformly chosen at random, or computed upon completion of an uncompromised KEM or QKD exchange.

Theorem 2. *Let* Π *be the AKE protocol depicted by Figure 1.* Π *is a secure AKE protocol according to Definition 11. In particular, for any polynomial time (quantum) adversary* $\mathcal{A}$ *playing the experiments* $G^{\mathsf{sec}}_{\Pi,\mathcal{A}}$ *described by Figure 5 and Figure 6 with at most* n_P *parties that each establishes at most* n_S *sessions, each*

comprising at most n_T stages, we have

$$
\begin{aligned}
\mathsf{adv}^{\mathsf{sound}}_{\Pi}(\mathcal{A}) \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} + \mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}} \\
\mathsf{adv}^{\mathsf{auth},a)}_{\Pi}(\mathcal{A}) \leq\ & n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\
& + n_T \cdot (n_P \cdot n_S)^2 \cdot \left(\frac{1}{2^{\ell_{hk}-1}} + \frac{5}{2^{\ell_k}}\right) + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} + n_P^2 \cdot n_S \cdot n_T \cdot \mathsf{adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG}}(1) \\
& + n_T \cdot (n_P \cdot n_S)^2 \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}1cca}}_{\mathsf{KEM}}(1) + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) \\
& \qquad + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) + \ 4 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1)\Big) \\
\mathsf{adv}^{\mathsf{auth},b)2)}_{\Pi}(\mathcal{A}) \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \epsilon_s\right) \\
& + n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T-1}\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} + \mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13) \\
& \qquad + 4 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) + n_T \cdot \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) \\
& \qquad + (n_T - 2) \cdot \Big(2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big) \\
& \qquad + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)\Big)\Big) \\
\mathsf{adv}^{\mathsf{key\text{-}ind},c)1)}_{\Pi}(\mathcal{A}) \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\
& + n_T \cdot (n_P \cdot n_S)^2 \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) \\
& \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{adv}^{\mathsf{key\text{-}ind},c)2)}_{\Pi}(\mathcal{A}) \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\
& + n_T \cdot (n_P \cdot n_S)^2 \cdot \left(\epsilon_s + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)\right) \\
\mathsf{adv}^{\mathsf{id\text{-}priv},a)1)}_{\Pi}(\mathcal{A}) \leq\ & n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}} + (n_P \cdot n_S)^2 \cdot n_T \cdot \left(\frac{1}{2^{\ell_{hk}-1}} + \frac{3}{2^{\ell_k}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} \\
& + (n_P \cdot n_S)^2 \cdot n_T \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(11) \\
& \quad + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) + 3 \cdot \mathsf{adv}^{\mathsf{lh}}_{\mathsf{AE}}(1,1)\Big)
\end{aligned}
$$

where a) and b)2) refer to subcases in predicate FreshAuth*, c)1) and c)2) refer to subcases in predicate* FreshKey*, and a)1) refers to the first subcase in predicate* FreshIdentity*.*

$\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}}$*,* $\mathsf{adv}^{\mathsf{ind\text{-}1cca}}_{\mathsf{KEM}}$*,* $\mathsf{adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG}}$*,* $\mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}}$*,* $\mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}}$*,* $\mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}}$*,* $\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}$*,* $\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}$*,* $\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}$*,* $\mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}}$*,* $\mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}$*,* $\mathsf{adv}^{\mathsf{lh}}_{\mathsf{AE}}$ *denote the advantage of an adversary in breaking the IND-CPA, IND-1CCA security of the KEM* KEM*, the EUF-CMA security of the signature scheme* SIG*, the collision resistance of the hash function* H *and functions* Extract*,* Expand*, the swap-PRF security of* Extract*, the PRF security of functions* Extract*,* Expand*, and the KEY-CMT, INT-CTXT and LH security of the authenticated encryption scheme* AE *respectively.*

ℓ_{hk}*,* ℓ_k*,* ℓ_{pss} *and* λ *denote the bit length of the intermediary keys* hk*,* k_0*,* k_1*,* k_2*,* pss *and* H_{ec} *outputs respectively.* δ_{KEM}*,* μ_{KEM}*,* γ_{KEM} *denote the probability of decapsulation failure, the collision probability of public keys and the collision probability of ciphertexts respectively of the KEM* KEM*.* ϵ_s *is an upper estimate for the secrecy bound of the QKD phase (cf. Section 4.7).*

In Theorem 2, we provide security bounds corresponding to five subcases among the multiple listed in the different predicates FreshAuth, FreshKey and FreshIdentity. They exemplify the impact of the ephemeral KEM and quantum key exchanges on the protocol security.

Detailed security proofs and corresponding bounds are given in Section A.

The first bound corresponds to the soundness property. The second bound (corresponding to FreshAuth a)) is related to the explicit authentication experiment (when the targeted instance interacts with an honest instance and its long-term key is uncompromised). The security bound shows that, in this context, the protocol relies essentially upon the KEM and signature scheme security. The next bound (FreshAuth b)2)) is related to the explicit authentication experiment when the targeted instance completes a previous run with a matching instance such that the QKD phase is successful. Then, security relies upon an

uncompromised pss value (computed in particular from the output of the successful QKD run). The two next bounds (FreshKey c)1) and c)2)) are related to the key indistinguishability experiment. In this context, the tested instance has a matching instance. The session keys remain "safe" thanks to the ephemeral KEM exchange (subcase c)1)) or the quantum key exchange (subcase c)2)) that occurs during the run, and the adversary is allowed to reveal any other protocol parameter. The last bound (corresponding to FreshIdentity a)1)) is related to the identity privacy experiment (when the targeted instance interacts with an honest instance). The property is ensured essentially thanks to the security of the KEM scheme and that of the length-hiding AE function.

5.2 Discussion

Qualitative Results. The soundness property relies essentially on the KEM scheme KEM (no decapsulation failure), the 2-universal hash function H_{ec} used during the error correction step of the QKD phase (no collision) and on the symmetric-key functions Extract, Expand (no collision) and AE (key commitment) to ensure that two partnered instances, upon a successful run, share the three secret values (KEM shared secret ss, quantum key W, pss) involved in the session keys computation.

The explicit authentication property depends essentially upon the security of the signature scheme (and the IND-1CCA security of the KEM scheme) to ensure that the targeted instance (and its partner) accepts only upon reception of an honest signed message (or AE ciphertext). Alternatively, the secrecy of pss shared by two instances (and the security of the symmetric-key functions Extract, Expand and AE) enables ensuring this property. In turn, pss may depend on a previous (uncompromised) KEM exchange or QKD phase (which can be seen as an initialisation step for pss). Then, the PRF (and swap-PRF) security of the functions Extract and Expand is also involved to ensure that the intermediary keys (possibly computed among several successive stages) are indistinguishable from values uniformly chosen at random. In that case, the protocol behaves as an (evolving) shared symmetric-key AKE.

Proving the explicit authentication property implies a "bootstrap": either a secure signature scheme or an initial uniformly random pss.[10] That is, a preshared key or a (post-quantum) signature scheme is used at least during the very first run. Then, a key drawn from the QKD output (if security relies upon the QKD phase as in FreshAuth b)2) for instance) can be used to authenticate subsequent QKD runs. If the signature scheme is not broken before this first run or the preshared key (or the signature private key) is not disclosed before this first run, then the authentication key drawn from the QKD output can be safely used (if QKD is not broken and does not fail).[11]

[10] This holds also for the key indistinguishability and identity privacy properties.

[11] This is reminiscent of TLS 1.3 and the PSK/Resumption mode (with the classical or post-quantum variant): first an ephemeral ECDH or KEM exchange, authenticated with a signature scheme, that outputs a PSK used in subsequent runs.

The session keys are computed from three secret parameters: the KEM shared secret ss, the quantum key W, and pss. When the tested instance has a matching instance, then the key indistinguishability property relies essentially on the KEM scheme (IND-CPA), the QKD phase (indistinguishability of the quantum states) and the secrecy of pss. When the tested instance has no partner, then the security ensured by the signature scheme or the AE function (and also the secrecy of pss, which the AE keys depend on) forbid the adversary from computing a valid signature and AE ciphertext (consequently, the tested instance aborts the run and does not compute the session keys). Again, pss may depend on a previous (uncompromised) KEM exchange or QKD phase. Again, the protocol can then be compared as an (evolving) shared symmetric-key AKE (where the symmetric-key is safely initialised and remains uncompromised throughout the runs).

The identity privacy property relies essentially on the KEM scheme and the internal parameter pss since the AE scheme is keyed with keys computed from the KEM shared secret ss and pss (the quantum key is not involved in the computation). Alternatively, the secrecy of the AE keys may depend on a previous (uncompromised) quantum key since pss is derived from the latter. Finally, when the targeted instance has no partner, then the security ensured by the signature scheme or the AE function (and also the secrecy of pss, which the AE keys depend on) forbids the adversary from computing a valid signature and AE ciphertext (consequently, the targeted instance does not send its encrypted certificate). This property relies upon many same cryptographic components (and also same reasoning in the proofs) as key indistinguishability since the session keys derive from the same computation path as the intermediary AE keys.

As an additional note, in the proof of explicit authentication, we rely upon the IND-1CCA security property of the KEM scheme ($\mathsf{FreshAuth}$ a)). The KEM ciphertext is however protected with a MAC tag. Therefore, we should be able to leverage the EUF-CMA security ensured by the MAC scheme to relax the current IND-1CCA security of the KEM scheme and replace it with the IND-CPA property. The reason we do not so is related to our security model. The MAC key is derived from pss. Since, in our model, the adversary can reveal different parameters (including pss), the IND-1CCA property is required in the explicit authentication proof. In a weaker security model (without state reveals), the IND-CPA security of the KEM scheme would be sufficient, thanks to the MAC tag in m_2.

Quantitative Results. Our bounds are non tight. However, let us consider the two bounds for key indistinguishability given above (corresponding to subcases c)1) and c)2)). We can instantiate KEM such that the related quantities are low. For instance, with ML-KEM [92] δ_{KEM} is small [92, Table 1], the ciphertexts and public keys are pseudo-random [6, Section 4.3.1] therefore $2^{-\mu_{\mathsf{KEM}}}$ and $2^{-\gamma_{\mathsf{KEM}}}$ are negligible, and $\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}}$ [6, Table 4] is also low for the different ML-KEM parameters. Regarding the symmetric-key functions, we can instantiate $\mathsf{Extract}$ and Expand as HKDF-Extract and HKDF-Expand [77] respectively, that

is HMAC [11] in our context. We can choose parameters (e.g. 256-bit key, 512-bit hash output) such that the (swap-)PRF security bound [64] is low (2^{-131}) for reasonable adversarial time and power (at most 2^{50} queries).[12,13]

Regarding the QKD phase, n (number of bits used for distillation) is a fraction of M (the number of quantum states sent during a QKD run): $n = \beta \times m$ where β ranges roughly from 1/100 to 1/2, and $m = \eta \times M$ where $\eta < 1$ is the overall transmittance of the fiber. M ranges typically from 10^6 up to 10^{10}. We can (theoretically) choose parameters such that $\epsilon_{ec} = 2^{-\lambda}$ and ϵ_Δ are negligible. However, if we look at experimental deployments or commercially available equipment for QKD (e.g. [26,27,54,62,69,80,111]), based on the BB84 protocol or a decoy-state variant of the latter, the claimed security bound ranges from $10^{-9} \simeq 2^{-30}$ to $10^{-10} \simeq 2^{-33}$ at best, for distances from 115 to 150 Km and a quantum key rate between 1 and 300 Kb/s. Among the aforementioned references, several define the QKD security bound as the maximum probability that the final QKD key differs from an ideal key (i.e. a value of same length as the QKD key and uniformly chosen at random), as in [109]. Moreover, this bound is computed from parameters that (unsurprisingly) characterise the same steps of the QKD phase as we do (in particular, parameter estimation, error correction, privacy amplitude). Consequently, we use these bounds as concrete numerical examples to approximate our bound ϵ_Δ.

This implies that ϵ_Δ would be much higher in the inequality related to $\mathsf{adv}_\Pi^{\mathsf{key\text{-}ind},c)2)}(\mathcal{A})$ than the other terms mentioned previously and compared to what can theoretically be expected. Considering a minimum number of AKE runs of $n_P \cdot n_S \cdot n_T = 2^{35}$ this leads to non-meaningful security bounds. These bounds compare to what can be achieved with some lightweight cryptographic protocols (solely built upon symmetric-key functions and deployed in constrained environments) but not to what AKE protocols built upon classical or post-quantum cryptographic functions can provide in general.

Improving the QKD Security Bound. The quantity ϵ_{ec} (as well as the r-bit syndrome Z) impacts the quantum key rate (ℓ/M) since a lower ϵ_{ec} implies that more bits (λ) are disclosed. To compensate, one can choose a higher number of transmitted quantum states (M). But, this increases the duration of the run.

[12] The latter estimate is based on a model where the (quantum) adversary is granted queries in superposition (so-called Q2 model, where the device running the targeted cryptographic functions is a quantum device). In our work, we consider the Q1 model where the device is classical (it cannot be requested with quantum queries). Therefore, this estimate is likely higher than what could be achieved in the Q1 model (to the best of our knowledge, there exists no work analysing the security of HMAC in the latter model). If we consider classical security of HMAC, then we get a bound 2^{-156} with a 256-bit key and 256-bit hash output.

[13] Note that, if several security bounds for asymmetric and symmetric-key algorithms are asymptomatic, they are also valid for finite-size parameters. Practical cryptanalysis results (e.g. [21, 28, 97, 114]) show the relevance of these bounds to select appropriate algorithm parameters, as illustrated by several works (e.g. [44, 45]).

The same holds regarding $\epsilon_{pa}(\nu)$ which depends, in an opposite way, on r and λ on the one hand, and m (i.e. M) on the other hand. Likewise, a smaller security bound implies a longer run. Therefore, it seems that the QKD parameters are chosen in practice to achieve a trade-off (expected acceptable in practical deployments) balancing the key rate, the duration time and the security level (ϵ_{Δ}) of the QKD run.

While the loss implied by the (non-quantum) cryptographic algorithms can be decreased with an appropriate choice of parameters (e.g. bigger hash output or symmetric key size), the loss due to the QKD phase seems dependent on improvements of the (quantum) protocol implemented or the equipment deployed in practical QKD networks, unless the analyses leading to these aforementioned concrete bounds can be reassessed to achieve lower figures.

Another option to decrease the security bound ϵ_{Δ} in the proposed hybrid AKE protocol consists in relying upon the security ensured by (non-quantum) cryptographic schemes. The classical data related to the QKD phase (in particular the r-bit syndrome Z and the λ-bit digest T) are usually not encrypted (they must only be authenticated). If encrypted, then these values become opaque to an outside observer. Removing Z and T from the data accessible to the adversary allows reducing $\epsilon_{pa}(\nu)$ by a factor $\sqrt{2^{r+\lambda}}$ and the overall security bound ϵ_{Δ} could be decreased as well (see the proof of Proposition 11 in [109]). In turn, the confidentiality of Z and T depends on the IND-CPA security (in our context) or even the LH security of the AE scheme, the (swap-)PRF security of the functions $\mathsf{Extract}$ and Expand as well as the other properties that allow safely replacing the AE key with a uniformly random string. This suggestion means trading $2^{(r+\lambda)/2}$ for $\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{AE}}$ or $\mathsf{adv}^{\mathsf{lh}}_{\mathsf{AE}}$ added to other quantities, starting with $\mathsf{adv}^{\mathsf{(swap\text{-})prf}}_{\mathsf{Extract}}$ and $\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}$. Therefore, the relevance of this change (when Z and T are not encrypted) depends on the choice of the functions involved in the QKD (synd, $\mathsf{H_{ec}}$) and non-quantum (AE scheme, PRF functions in particular) phases respectively.

In the proposed hybrid AKE protocol, the confidentiality of the values Z and T (as well as other classical data related to the QKD phase) is indeed ensured. However, we do not apply the above suggestion in our proofs because, in our security model, reveal queries can be made when the QKD security is invoked, which allows the adversary to retrieve the parameters (internal state pss, KEM shared secret ss) used to compute the AE key. Consequently, we keep $\epsilon_{pa}(\nu)$ (hence ϵ_{Δ}) as expressed in Theorem 1.

6 Conclusion

In this paper, we present a two-party hybrid AKE protocol which combines classical and post-quantum algorithms, as well as a QKD sub-protocol. We do not focus on nor investigate the security of QKD technologies but, based on existing security proofs of QKD protocols, we aim at proposing a more comprehensive way to analyse such hybrid AKE protocols.

The security model that we define incorporates distinct security properties

(as opposed to "all-in-one" models) corresponding to the main expected properties for AKE protocols: soundness, explicit authentication, key indistinguishability and identity privacy. Existing security analysis of hybrid AKE protocols treat the QKD phase as a black box where the impact of establishing the quantum key appears similar as setting a pre-shared key at the two communicating parties. On the contrary, our security proofs (given for each of the security properties) make clear the security loss implied by the QKD phase.

Our security analysis, combined with the claimed characteristics of experimental deployments or commercially available equipment for QKD we are aware of, shows that the security loss induced by the QKD phase of the proposed hybrid AKE protocol is higher (although it can theoretically be negligible) than that of due to the (classical or post-quantum) cryptographic components of the protocol, and does not allow getting meaningful security bounds. The loss due to the QKD phase seems dependent on improvements of the (quantum) protocol implemented or the equipment deployed in practical QKD networks, unless the analyses leading to these bounds can be reassessed to achieve lower figures.

The AKE protocol that we present assumes a direct quantum channel established between the two communicating parties, with no entities in between. Other limitations of our security model and results inherit from those of the security analysis of the QKD phase that we use in our work. In particular, side-channel attacks against the quantum equipment are not captured (previous works have shown the feasibility of such scenarios against commercially available equipment).

As future work, side-channel attacks can be integrated in the security proof of the QKD phase and the security model for the hybrid AKE protocol. The security analysis can also be extended to take into account that multiple parties do access, in real-life deployments, the same QKD endpoint (entry point to the quantum channel), and the existence of intermediary nodes between two QKD endpoints.

Acknowledgements. The author would like to thank Pierre-Alain Fouque and André Schrottenloher for helpful discussions and, in particular, for pointing out the paper of Tomamichel and Leverrier [109]. The author thanks also anonymous CiC reviewers as well as Sebastian Verschoor, Kathrin Hövelmanns, Daan Planken, and Christian Schaffner [112] for valuable remarks and questions which helped improve the paper (see also Section B). This work has received funding from the European Union's Horizon Europe research and innovation programme under the project Quantum Security Networks Partnership (QSNP, grant agreement No 101114043).

References

1. Aguilar Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Bos, J., Deneuville, J.C., Dion, A., Gaborit, P., Lacan, J., Persichetti, E., Robert, J.M., Véron, P., Zémor, G.: Hamming Quasi-Cyclic (HQC) – Fourth round version (2025), `http://pqc-hqc.org/documentation.html`

2. Alléaume, R.: Quantum cryptography and its application frontiers. (La cryptographie quantique et ses frontières applicatives) (2021), https://tel.archives-ouvertes.fr/tel-04255325
3. ANSSI: Should Quantum Key Distribution be Used for Secure Communications? (2020), https://cyber.gouv.fr/en/publications/should-quantum-key-distribution-be-used-secure-communications
4. ANSSI: ANSSI views on the Post-Quantum Cryptography transition (2022), https://cyber.gouv.fr/en/publications/anssi-views-post-quantum-cryptography-transition
5. Aquina, N., Rommel, S., Monroy, I.T.: Quantum secure communication using hybrid post-quantum cryptography and quantum key distribution. In: 2024 24th International Conference on Transparent Optical Networks (ICTON). pp. 1–4 (2024). https://doi.org/10.1109/ICTON62926.2024.10648124
6. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber – Algorithm Specifications And Supporting Documentation (version 3.01) (2021), https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf
7. Avoine, G., Canard, S., Ferreira, L.: Symmetric-key authenticated key exchange (SAKE) with perfect forward secrecy. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 199–224. Springer, Cham (Feb 2020). https://doi.org/10.1007/978-3-030-40186-3˙10
8. Barnes, R., Bhargavan, K., Lipp, B., Wood, C.: Hybrid Public Key Encryption (2022), https://datatracker.ietf.org/doc/html/rfc9180
9. Battarbee, C., Striecks, C., Perret, L., Ramacher, S., Verhaeghe, K.: Quantum-Safe Hybrid Key Exchanges with KEM-Based Authentication. CoRR **abs/2411.04030** (2024). https://doi.org/10.48550/ARXIV.2411.04030
10. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Berlin, Heidelberg (Aug 2006). https://doi.org/10.1007/11818175˙36
11. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 1–15. Springer, Berlin, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5˙1
12. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS. pp. 394–403. IEEE Computer Society Press (Oct 1997). https://doi.org/10.1109/SFCS.1997.646128
13. Bellare, M., Hoang, V.T.: Efficient schemes for committing authenticated encryption. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 845–875. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3˙29
14. Bellare, M., Lysyanskaya, A.: Symmetric and dual PRFs from standard assumptions: A generic validation of an HMAC assumption. Cryptology ePrint Archive, Report 2015/1198 (2015), https://eprint.iacr.org/2015/1198
15. Bellare, M., Lysyanskaya, A.: Symmetric and dual PRFs from standard assumptions: A generic validation of a prevailing assumption. Journal of Cryptology **37**(4), 33 (Oct 2024). https://doi.org/10.1007/s00145-024-09513-6
16. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 232–249. Springer, Berlin, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48329-2˙21
17. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006.

LNCS, vol. 4004, pp. 409–426. Springer, Berlin, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679˙25
18. Bennett, C.H.: Quantum cryptography using any two nonorthogonal states. Phys. Rev. Lett. **68**, 3121–3124 (May 1992). https://doi.org/10.1103/PhysRevLett.68.3121
19. Bennett, C.H., Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. In: Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing. pp. 175–179 (1984). https://doi.org/10.48550/arXiv.2003.06557
20. Bennett, C.H., Brassard, G., Mermin, N.D.: Quantum cryptography without Bell's theorem. Phys. Rev. Lett. **68**, 557–559 (Feb 1992). https://doi.org/10.1103/PhysRevLett.68.557
21. Bernstein, D.J., Engels, S., Lange, T., Niederhagen, R., Paar, C., Schwabe, P., Zimmermann, R.: Faster discrete logarithms on FPGAs. Cryptology ePrint Archive, Report 2016/382 (2016), `https://eprint.iacr.org/2016/382`
22. Beullens, W.: Breaking rainbow takes a weekend on a laptop. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 464–479. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4˙16
23. Bird, R., Gopal, I.S., Herzberg, A., Janson, P.A., Kutten, S., Molva, R., Yung, M.: Systematic design of two-party authentication protocols. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 44–61. Springer, Berlin, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1˙3
24. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M. (ed.) 6th IMA International Conference on Cryptography and Coding. LNCS, vol. 1355, pp. 30–45. Springer, Berlin, Heidelberg (Dec 1997). https://doi.org/10.1007/bfb0024447
25. Blanco-Romero, J., García, P.O., Sobral-Blanco, D., Mendoza, F.A., Vilas, A.F., Díaz-Redondo, R.P.: QKD-KEM: Hybrid QKD Integration into TLS with OpenSSL Providers (2025), `https://arxiv.org/abs/2503.07196`
26. Boaron, A., Boso, G., Rusca, D., Vulliez, C., Autebert, C., Caloz, M., Perrenoud, M., Gras, G., Bussières, F., Li, M.J., Nolan, D., Martin, A., Zbinden, H.: Secure Quantum Key Distribution over 421 km of Optical fiber. Phys. Rev. Lett. **121** (November 2018). https://doi.org/10.1103/PhysRevLett.121.190502
27. Boaron, A., Korzh, B., Houlmann, R., Boso, G., Rusca, D., Gray, S., Li, M.J., Nolan, D., Martin, A., Zbinden, H.: Simple 2.5 GHz time-bin quantum key distribution. Appl. Phys. Lett. **112**(17) (April 2018). https://doi.org/10.1063/1.5027030
28. Bos, J.W., Kaihara, M.E., Kleinjung, T., Lenstra, A.K., Montgomery, P.L.: Solving a 112-bit prime elliptic curve discrete logarithm problem on game consoles using sloppy reduction. Int. J. Appl. Cryptol. **2**(3), 212–228 (Feb 2012)
29. Boyd, C., Davies, G.T., de Kock, B., Gellert, K., Jager, T., Millerjord, L.: Symmetric key exchange with full forward security and robust synchronization. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 681–710. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5˙23
30. Brassard, G., Lütkenhaus, N., Mor, T., Sanders, B.C.: Limitations on Practical Quantum Cryptography. Phys. Rev. Lett. **85**, 1330–1333 (Aug 2000). https://doi.org/10.1103/PhysRevLett.85.1330
31. Bruckner, S., Ramacher, S., Striecks, C.: Muckle+: End-to-end hybrid authenticated key exchanges. In: Johansson, T., Smith-Tone, D. (eds.) PQCrypto 2023. pp.

601–633. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-40003-2˙22
32. Buruaga, J.S., Méndez, R.B., Brito, J.P., Martin, V.: Quantum-Safe integration of TLS in SDN networks (2025), https://arxiv.org/abs/2502.17202
33. Buruaga, J., Bugler, A. Brito, J., Martin, V., Striecks, C.: Versatile quantum-safe hybrid key exchange and its application to MACsec. EPJ Quantum Technol. **12**(84) (2025). https://doi.org/https://doi.org/10.1140/epjqt/s40507-025-00382-x
34. Campagna, M., Chen, L., Özgür Dagdelen, Ding, J., Fernick, J.K., Gisin, N., Hayford, D., Jennewein, T., Lütkenhaus, N., Mosca, M., Neill, B., Pecen, M., Perlner, R., Ribordy, G., Schanck, J.M., Stebila, D., Walenta, N., Whyte, W., Zhang, Z.: Quantum Safe Cryptography and Security – An introduction, benefits, enablers and challenges (2015), https://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf, ETSI White Paper No. 8
35. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 423–447. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4˙15
36. Chen, L., Xue, K., Li, J., Li, Z., Yu, N.: Security-Enhanced WireGuard Protocol Design Using Quantum Key Distribution. In: 2024 International Conference on Computing, Networking and Communications (ICNC). pp. 718–723 (2024). https://doi.org/10.1109/ICNC59896.2024.10556292
37. Christandl, M., Renner, R., Ekert, A.: A Generic Security Proof for Quantum Key Distribution (2004), https://arxiv.org/abs/quant-ph/0402131
38. Cohn-Gordon, K., Cremers, C.J.F., Garratt, L.: On post-compromise security. In: Hicks, M., Köpf, B. (eds.) CSF 2016 Computer Security Foundations Symposium. pp. 164–178. IEEE Computer Society Press (2016). https://doi.org/10.1109/CSF.2016.19
39. Cremers, C., Dax, A., Medinger, N.: Keeping up with the KEMs: Stronger security notions for KEMs and automated analysis of KEM-based protocols. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 1046–1060. ACM Press (Oct 2024). https://doi.org/10.1145/3658644.3670283
40. Cremers, C., Jacomme, C., Naska, A.: Formal analysis of session-handling in secure messaging: Lifting security from sessions to conversations. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 1235–1252. USENIX Association (Aug 2023), https://www.usenix.org/conference/usenixsecurity23/presentation/cremers-session-handling
41. Cremers, C., Medinger, N., Naska, A.: Impossibility results for post-compromise security in real-world communication systems. Cryptology ePrint Archive, Report 2024/1886 (2024), https://eprint.iacr.org/2024/1886
42. Cremers, C.J.F.: Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. In: Atluri, V., Díaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 315–334. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23822-2˙18
43. 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. LNCS, vol. 7459, pp. 734–751. Springer, Berlin, Heidelberg (Sep 2012). https://doi.org/10.1007/978-3-642-33167-1˙42
44. Davis, H., Diemert, D., Günther, F., Jager, T.: On the concrete security of TLS 1.3 PSK mode. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 876–906. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3˙30

45. Davis, H., Günther, F.: Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. In: Sako, K., Tippenhauer, N.O. (eds.) ACNS 2021, Part II. LNCS, vol. 12727, pp. 448–479. Springer, Cham (Jun 2021). https://doi.org/10.1007/978-3-030-78375-4˙18
46. Diemert, D., Jager, T.: On the tight security of TLS 1.3: Theoretically sound cryptographic parameters for real-world deployments. Journal of Cryptology **34**(3), 30 (Jul 2021). https://doi.org/10.1007/s00145-021-09388-x
47. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. DCC **2**(2), 107–125 (1992). https://doi.org/10.1007/BF00124891
48. Dowling, B., Hansen, T.B., Paterson, K.G.: Many a mickle makes a muckle: A framework for provably quantum-secure hybrid key exchange. Cryptology ePrint Archive, Report 2020/099 (2020), `https://eprint.iacr.org/2020/099`
49. Dowling, B., Hansen, T.B., Paterson, K.G.: Many a mickle makes a muckle: A framework for provably quantum-secure hybrid key exchange. In: Ding, J., Tillich, J.P. (eds.) PQCrypto 2020. pp. 483–502. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44223-1˙26
50. Ekert, A.K.: Quantum cryptography based on Bell's theorem. Phys. Rev. Lett. **67**, 661–663 (Aug 1991). https://doi.org/10.1103/PhysRevLett.67.661
51. Ferreira, L.: AKE Protocol Combining PQC and QKD. Cryptology ePrint Archive, Report 2026/036/20260109:090307 (2026), `https://eprint.iacr.org/archive/2026/036/20260109:090307`
52. Fischlin, M., Günther, F.: Multi-stage key exchange and the case of Google's QUIC protocol. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 1193–1204. ACM Press (Nov 2014). https://doi.org/10.1145/2660267.2660308
53. French Cybersecurity Agency (ANSSI), Federal Office for Information Security (BSI), Netherlands National Communications Security Agency (NL-NCSA), Swedish National Communications Security Authority, Swedish Armed Forces: Position paper on Quantum Key Distribution (2024), `https://english.aivd.nl/publications/publications/2024/06/19/position-paper-on-quantum-key-distribution`
54. Fröhlich, B., Lucamarini, M., Dynes, J.F., Comandar, L.C., Tam, W.W.S., Plews, A., Sharpe, A.W., Yuan, Z., Shields, A.J.: Long-distance quantum key distribution secure against coherent attacks. Optica **4**(1), 163–167 (January 2017). https://doi.org/10.1364/OPTICA.4.000163
55. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Berlin, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1˙34
56. Garcia, C.R., Aguilera, A.C., Olmos, J.J.V., Monroy, I.T., Rommel, S.: Quantum-Resistant TLS 1.3: A Hybrid Solution Combining Classical, Quantum and Post-Quantum Cryptography. In: 2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). pp. 246–251 (2023). https://doi.org/10.1109/CAMAD59638.2023.10478407
57. Garms, L., Paraïso, T.K., Hanley, N., Khalid, A., Rafferty, C., Grant, J., Newman, J., Shields, A.J., Cid, C., O'Neill, M.: Experimental Integration of Quantum Key Distribution and Post-Quantum Cryptography in a Hybrid Quantum-Safe Cryptosystem. Advanced Quantum Technologies **7**(4) (2024). https://doi.org/https://doi.org/10.1002/qute.202300304

58. Garms, L., Paraïso, T.K., Hanley, N., Khalid, A., Rafferty, C., Grant, J., Newman, J., Shields, A.J., Cid, C., O'Neill, M.: Supporting Information for Experimental Integration of Quantum Key Distribution and Post-Quantum Cryptography in a Hybrid Quantum-Safe Cryptosystem. Advanced Quantum Technologies **7**(4) (2024). https://doi.org/https://doi.org/10.1002/qute.202300304
59. Geitz, M., Döring, R., Braun, R.P.: Hybrid QKD & PQC Protocols implemented in the Berlin OpenQKD testbed. In: 2023 8th International Conference on Frontiers of Signal Processing (ICFSP). pp. 69–74 (2023). https://doi.org/10.1109/ICFSP59764.2023.10372894
60. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing **17**(2), 281–308 (Apr 1988). https://doi.org/10.1137/0217017
61. Helstrom, C.W.: Quantum Detection and Estimation Theory. Mathematics in science and engineering, Academic Press (1976)
62. HEQA Security: Products, `https://heqa-sec.com/product/`
63. Hermans, J., Pashalidis, A., Vercauteren, F., Preneel, B.: A new RFID privacy model. In: Atluri, V., Díaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 568–587. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23822-2˙31
64. Hosoyamada, A., Iwata, T.: On tight quantum security of HMAC and NMAC in the quantum random oracle model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 585–615. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0˙21
65. Hövelmanns, K., Planken, D., Schaffner, C., Verschoor, S.: QKD oracles for authenticated key exchange. Cryptology ePrint Archive, Report 2025/1671 (2025), `https://eprint.iacr.org/2025/1671`
66. Huang, A., Sun, S.H., Liu, Z., Makarov, V.: Quantum key distribution with distinguishable decoy states. Phys. Rev. A **98**, 012330 (Jul 2018). https://doi.org/10.1103/PhysRevA.98.012330
67. Huguenin-Dumittan, L., Vaudenay, S.: On IND-qCCA security in the ROM and its applications - CPA security is sufficient for TLS 1.3. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 613–642. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2˙22
68. Huttner, B., Alléaume, R., Diamanti, E., Fröwis, F., Grangier, P., Hübel, H., Martin, V., Poppe, A., Slater, J.A., Spiller, T., Tittel, W., Tranier, B., Wonfor, A., Zbinden, H.: Long-range QKD without trusted nodes is not possible with current technology. npj Quantum Inf **8**(108) (September 2022). https://doi.org/10.1038/s41534-022-00613-4
69. IDQ: Clavis XG QKD System, `https://www.idquantique.com/quantum-safe-security/products/clavis-xg-qkd-system/`
70. IEEE: IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security (2018). https://doi.org/10.1109/IEEESTD.2018.8585421
71. Iyengar, J., Thomson, M.: QUIC: A UDP-Based Multiplexed and Secure Transport (2021), `https://datatracker.ietf.org/doc/html/rfc9000`
72. Jacqmin, P.A., Liénardy, J.: LP2+: a robust symmetric-key AKE protocol with perfect forward secrecy, and an advocacy for thorough security proofs. Cryptology ePrint Archive, Report 2025/983 (2025), `https://eprint.iacr.org/2025/983`
73. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: Canteaut, A., Standaert, F.X. (eds.) EURO-

CRYPT 2021, Part I. LNCS, vol. 12696, pp. 117–146. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5˙5
74. Jiang, H., Ma, Z., Zhang, Z.: Post-quantum security of key encapsulation mechanism against CCA attacks with a single decapsulation query. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part IV. LNCS, vol. 14441, pp. 434–468. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8730-6˙14
75. Kaufman, C., Hoffman, P., Nir, Y., Eronen, P.: Internet Key Exchange Protocol Version 2 (IKEv2) (2010), https://datatracker.ietf.org/doc/html/rfc5996
76. König, R., Renner, R., Bariska, A., Maurer, U.: Small Accessible Quantum Information Does Not Imply Security. Phys. Rev. Lett. **98** (Apr 2007). https://doi.org/10.1103/PhysRevLett.98.140502
77. Krawczyk, H.: Cryptographic extraction and key derivation: The HKDF scheme. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 631–648. Springer, Berlin, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7˙34
78. Krawczyk, H., Wee, H.: The OPTLS protocol and TLS 1.3. In: 2016 IEEE European Symposium on Security and Privacy. pp. 81–96. IEEE Computer Society Press (Mar 2016). https://doi.org/10.1109/EuroSP.2016.18
79. Li, Y., Schäge, S.: No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1343–1360. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3134006
80. Lucamarini, M., Patel, K.A., Dynes, J.F., Fröhlich, B., Sharpe, A.W., Dixon, A.R., Yuan, Z.L., Penty, R.V., Shields, A.J.: Efficient decoy-state quantum key distribution with quantified security. Optics Express **21**(21) (October 2013). https://doi.org/10.1364/oe.21.024550
81. Lütkenhaus, N.: Security against individual attacks for realistic quantum key distribution. Phys. Rev. A **61**, 052304 (Apr 2000). https://doi.org/10.1103/PhysRevA.61.052304
82. Lychev, R., Jero, S., Boldyreva, A., Nita-Rotaru, C.: How secure and quick is QUIC? Provable security and performance analyses. In: 2015 IEEE Symposium on Security and Privacy. pp. 214–231. IEEE Computer Society Press (May 2015). https://doi.org/10.1109/SP.2015.21
83. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 448–471. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4˙16
84. Makris, N., Papageorgopoulos, A., Tsimvrakidis, K., Konteli, P., Gautier, Y., Terenziani, M., Daudin, E., Ntoulias, D., Fragkioudakis, T., Meletios, I., Mosca, M., Hobbs, D., Rosati, T., Papastamatiou, I., Prnjat, O., Koumantaros, K., Mitropoulos, D., Morax, J.R., Huttner, B., Christodoulopoulos, O.K., Kanellos, G.T., Syvridis, D.: Field demonstration of a fully managed, L1 encrypted 3-node network with hybrid relayed-QKD and centralized symmetric classical key management (2024), https://arxiv.org/abs/2403.08526
85. Marquardt, C., Seyfarth, U., Bettendorf, S., Bohmann, M., Buchner, A., Curty, M., Elser, D., Eul, S., Gehring, T., Jain, N., Klocke, T., Reinecke, M., Sieber, N., Ursin, R., Wehling, M., Weier, H.: Implementation Attacks against QKD Systems (2023), https://www.bsi.bund.de/EN/Service-Navi/Publikationen/Studien/QKD-Systems/Implementation_Attacks_QKD_Systems_node.html
86. Mendez, R.B., Buruaga, J.S., Vicente, R.J., Mengual, L., Pastor, A., Muñiz, A., Morales, J., Canto, R., Folgueira, J., Lopez, D.R., Martin,

V., Brito, J.P.: SDN-Based Hybrid Quantum-Safe Domain Intercommunication Within MadQCI. In: 2024 International Conference on Quantum Communications, Networking, and Computing (QCNC). pp. 168–175 (2024). https://doi.org/10.1109/QCNC62729.2024.00034
87. Mosca, M., Stebila, D., Ustaoglu, B.: Quantum key distribution in the classical authenticated key exchange framework. Cryptology ePrint Archive, Report 2012/361 (2012), https://eprint.iacr.org/2012/361
88. Mosca, M., Stebila, D., Ustaoglu, B.: Quantum key distribution in the classical authenticated key exchange framework. In: Gaborit, P. (ed.) PQCrypto 2013. pp. 136–154. Springer, Berlin, Heidelberg (Jun 2013). https://doi.org/10.1007/978-3-642-38616-9˙9
89. Müller-Quade, J., Renner, R.: Composability in quantum cryptography. New Journal of Physics **11**(8) (aug 2009). https://doi.org/10.1088/1367-2630/11/8/085006
90. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering generic composition. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 257–274. Springer, Berlin, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5˙15
91. NCSC: Quantum security technologies (2020), https://www.ncsc.gov.uk/whitepaper/quantum-security-technologies
92. NIST: Module-Lattice-Based Key-Encapsulation Mechanism Standard (2024), https://csrc.nist.gov/pubs/fips/203/final
93. NSA: Quantum Key Distribution (QKD) and Quantum Cryptography (QC) (2020), https://www.nsa.gov/Cybersecurity/Quantum-Key-Distribution-QKD-and-Quantum-Cryptography-QC/
94. Ouafi, K., Phan, R.C.W.: Traceable privacy of recent provably-secure RFID protocols. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 479–489. Springer, Berlin, Heidelberg (Jun 2008). https://doi.org/10.1007/978-3-540-68914-0˙29
95. Paterson, K.G., Ristenpart, T., Shrimpton, T.: Tag size does matter: Attacks and proofs for the TLS record protocol. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 372–389. Springer, Berlin, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0˙20
96. Piétri, Y., Verdier, P.E., Lacour, B., Gautier, M., Huang, H., Camus, T., Pegon, J.S., Zuber, M., Faugère, J.C., Schiavon, M., Rhouni, A., Jaouën, Y., Fabre, N., Alléaume, R., Rivera, T., Diamanti, E.: Quantum Key Distribution with Efficient Post-Quantum Cryptography-Secured Trusted Node on a Quantum Network (2025), https://arxiv.org/abs/2504.01454
97. Pons, J.L., Zieniewicz, A.: Pollard's kangaroo for secp256k1 – bitcoin puzzle #115 (114-bit interval) solution (2020), https://github.com/JeanLucPons/Kangaroo, gitHub commit b7e0f7c
98. Portmann, C., Renner, R.: Security in quantum cryptography. Reviews of Modern Physics **94**(2) (June 2022). https://doi.org/10.1103/revmodphys.94.025008
99. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (2018), https://datatracker.ietf.org/doc/html/rfc8446
100. Ricci, S., Dobias, P., Malina, L., Hajny, J., Jedlicka, P.: Hybrid Keys in Practice: Combining Classical, Quantum and Post-Quantum Cryptography. IEEE Access **12**, 23206–23219 (2024). https://doi.org/10.1109/ACCESS.2024.3364520
101. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 472–503. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4˙17

102. Rubio Garcia, C., Rommel, S., Takarabt, S., Vegas Olmos, J., Guilley, S., Nguyen, P., Tafur Monroy, I.: Quantum-resistant Transport Layer Security. Computer Communications **213**, 345–358 (January 2024). https://doi.org/10.1016/j.comcom.2023.11.010
103. Scarani, V., Acín, A., Ribordy, G., Gisin, N.: Quantum Cryptography Protocols Robust against Photon Number Splitting Attacks for Weak Laser Pulse Implementations. Phys. Rev. Lett. **92** (Feb 2004). https://doi.org/10.1103/PhysRevLett.92.057901
104. Schäge, S., Schwenk, J., Lauer, S.: Privacy-preserving authenticated key exchange and the case of IKEv2. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 567–596. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45388-6˙20
105. 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. pp. 1461–1480. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3423350
106. Shor, P.W., Preskill, J.: Simple Proof of Security of the BB84 Quantum Key Distribution Protocol. Phys. Rev. Lett. **85**, 441–444 (Jul 2000). https://doi.org/10.1103/PhysRevLett.85.441
107. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), `https://eprint.iacr.org/2004/332`
108. Stebila, D., Mosca, M., Lütkenhaus, N.: The Case for Quantum Key Distribution. In: Sergienko, A.V., Pascazio, S., Villoresi, P. (eds.) Quantum Communication and Quantum Networking, First International Conference, QuantumComm 2009. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 36, pp. 283–296. Springer (2009). https://doi.org/10.1007/978-3-642-11731-2_35
109. Tomamichel, M., Leverrier, A.: A largely self-contained and complete security proof for quantum key distribution. Quantum (2017), `https://doi.org/10.22331/q-2017-07-14-14`
110. Tomamichel, M., Renner, R.: Uncertainty Relation for Smooth Entropies. Physical Review Letters **106**(11) (Mar 2011). https://doi.org/10.1103/physrevlett.106.110506
111. Toshiba: Quantum Key Distribution, `https://www.global.toshiba/ww/products-solutions/security-ict/qkd/products.html`
112. Verschoor, S., Hövelmanns, K., Planken, D., Schaffner, C.: Private communication (March 2026)
113. Wang, L.J., Zhang, K.Y., Wang, J.Y., Cheng, J., Yang, Y.H., Tang, S.B., Yan, D., Tang, Y.L., Liu, Z., Yu, Y., Zhang, Q., Pan, J.W.: Experimental authentication of quantum key distribution with post-quantum cryptography. npj Quantum Inf **7**(67) (May 2021). https://doi.org/10.1038/s41534-021-00400-7
114. Wenger, E., Wolfger, P.: Solving the discrete logarithm of a 113-bit Koblitz curve with an FPGA cluster. In: Joux, A., Youssef, A.M. (eds.) SAC 2014. LNCS, vol. 8781, pp. 363–379. Springer, Cham (Aug 2014). https://doi.org/10.1007/978-3-319-13051-4˙22
115. Wiesemann, J., Krause, J., Tupkary, D., Lütkenhaus, N., Rusca, D., Walenta, N.: A consolidated and accessible security proof for finite-size decoy-state quantum key distribution (2024), `https://arxiv.org/abs/2405.16578`
116. Yang, Y.H., Li, P.Y., Ma, S.Z., Qian, X.C., Zhang, K.Y., Wang, L.J., Zhang, W.L., Zhou, F., Tang, S.B., Wang, J.Y., Yu, Y., Zhang, Q., Pan,

J.W.: All optical metropolitan quantum key distribution network with post-quantum cryptography authentication. Optics Express **29**(16) (July 2021). https://doi.org/10.1364/oe.432944
117. Yehia, R., Piétri, Y., Pascual-García, C., Lefebvre, P., Centrone, F.: Energetic Analysis of Emerging Quantum Communication Protocols (2025), https://arxiv.org/abs/2410.10661
118. Yin, H.L., Zhou, M.G., Gu, J., Xie, Y.M., Lu, Y.S., Chen, Z.B.: Tight security bounds for decoy-state quantum key distribution. Scientific Reports **10**(1) (Aug 2020). https://doi.org/10.1038/s41598-020-71107-6
119. Yuen, H.P.: Security of Quantum Key Distribution. IEEE Access **4**, 724–749 (2016). https://doi.org/10.1109/ACCESS.2016.2528227
120. Zeng, P., Bandyopadhyay, D., Méndez, J.A.M., Bitner, N., Kolar, A., Solomon, M.T., Ye, Z., Rozpedek, F., Zhong, T., Heremans, F.J., Awschalom, D.D., Jiang, L., Liu, J.: Practical hybrid PQC-QKD protocols with enhanced security and performance (2024), https://arxiv.org/abs/2411.01086
121. Zimmermann, P., Johnston, A., Callas, J.: ZRTP: Media Path Key Agreement for Unicast Secure RTP (2011), https://datatracker.ietf.org/doc/html/rfc6189

A Security Proofs for the AKE Protocol

In this section we give the detailed proofs of the AKE protocol depicted by Figure 1 corresponding to the security properties defined in Section 3.2: soundness, explicit authentication, key indistinguishability and identity privacy. Our analysis is done in the standard model with polynomial-time classical or quantum adversaries and according to the security model presented in Section 3. We proceed through a sequence of games [17,107] between a challenger and an adversary $\mathcal{A}$ that aims at breaking the considered security property.

We assume that the KEM shared secret ss, the parameter pss and the ephemeral quantum key ($W \in \{K_I, K_R\}$) have distinct length. This is a technicality not to deal with the possibility that two such parameters collide when they are output by a random function. Indeed, in several games, we replace the function $\mathsf{Extract}(\cdot, x)$, $x \in \{ss, pss, W\}$, with a random function. If we consider that the parameters ss, pss, W have the same size, then this adds supplementary losses (corresponding to the probability of collision). For instance, see the difference in the first games of the proof for the explicit authentication property between cases a) and b)1).

Among the different security properties, the same set of games may appear in the proof corresponding to distinct predicate subcases. For the sake of clarity and readability, we choose to be as comprehensive as possible in each proof without need to refer to another subcase proof at the cost of some redundancy.

A.1 Soundness

Let E_i be the event that an adversary wins the soundness experiment in Game i, and $\mathsf{adv}_i = \Pr[E_i]$.

Game 0. This game corresponds to the soundness security experiment described by Figure 5 in Section 3. Therefore

$$\Pr[E_0] = \mathsf{adv}^{\mathsf{sound}}_{\Pi}(\mathcal{A}) = \mathsf{adv}_0$$

Game 1. In this game, the challenger aborts the experiment if either an ephemeral KEM public key epk or ciphertext ct is not unique among all instances and stages. There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_0 \leq \mathsf{adv}_1 + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$

Game 2. In this game, the challenger aborts the experiment if a KEM decapsulation failure occurs. That is, the challenger aborts the experiment if, on the one hand, the encapsulation done by an honest instance with input an ephemeral public key epk outputs a ciphertext ct and a shared secret ss, and, on the other hand, upon reception of ct by the honest partnered instance the decapsulation with the corresponding private key esk outputs a value $ss' \neq ss$. We have that

$$\mathsf{adv}_1 \leq \mathsf{adv}_2 + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}$$

The two partnered instances share the same sid. Therefore, they share in particular the ephemeral KEM public key and ciphertext. Hence, they share the same KEM shared secret (because decapsulation failures are excluded by Game 2) and tr_0. We want to show now that they share also the same value pss. To that purpose, we will next rely on the key-committing property of the AE scheme. We start by excluding collisions in Expand and Extract functions. This implies that, if the two partnered instances use distinct pss values to compute the key hk_R, then they yield distinct keys to compute and decrypt c_R. Finally, since the AE scheme is a key-committing function, the decryption of c_R with a key different from the encryption key must output an error.

Game 3. In this game, the challenger aborts the experiment if a collision occurs in $\mathsf{Extract}$ outputs among all instances and stages. Therefore

$$\mathsf{adv}_2 \leq \mathsf{adv}_3 + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}}$$

Game 4. In this game, the challenger aborts the experiment if a collision occurs in Expand outputs among all instances and stages. Therefore

$$\mathsf{adv}_3 \leq \mathsf{adv}_4 + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}}$$

Game 5. In this game, the challenger aborts the experiment if an honest initiator's instance decrypts correctly the ciphertext c_R with a key different from the key used by the honest responder's instance to compute c_R, when both instances are partnered. This reduces straightforwardly to the key-committing property of the AE scheme. Therefore

$$\mathsf{adv}_4 \leq \mathsf{adv}_5 + \mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}}$$

Game 6. In this game, the challenger aborts the experiment if a collision occurs in outputs of the chosen 2-universal hash function H_{ec}. These outputs are λ-bit long. Therefore, by definition of H_{ec} (cf. Definition 24), we have

$$\mathsf{adv}_5 \leq \mathsf{adv}_6 + \frac{n_P \cdot n_S \cdot n_T}{2^\lambda}$$

where the multiplicative factor $n_P \cdot n_S \cdot n_T$ comes from the fact that a new 2-universal hash function H_{ec} is chosen during each stage.

The Sound predicate is false if (i) three honest instances share the same session identifier sid, or (ii) two partnered instances compute different session keys.

Regarding the first point, the ephemeral KEM public key and ciphertext are unique (by Game 1). Therefore a sid is shared by two instances at most in a given stage t.

Regarding the second point, in Game 6 the two partnered instances π and π' in stage t share the same values tr_0, pss (by Game 5) and KEM shared secret ss (by Game 2). Therefore, they share k_0, k_1 and the key hk_R used to encrypt and decrypt c_R (which is also shared by the two instances since $\pi.\mathsf{sid}[t] = \pi'.\mathsf{sid}[t]$). Hence, they share the plaintext corresponding to c_R (hence the same tr_1) and dk_1. Consequently, they share the key hk_I, hence the plaintext corresponding to c_I (because c_I is also shared by the two instances since $\pi.\mathsf{sid}[t] = \pi'.\mathsf{sid}[t]$). Therefore, they share tr_2, the key hk'_R and the plaintext corresponding to c'_R (because c'_R is also shared by the two instances since $\pi.\mathsf{sid}[t] = \pi'.\mathsf{sid}[t]$). Consequently, they share tr_3.

In particular, the two partnered instances share the same plaintext $flag_{\mathsf{pe}}$ (encrypted under c'_R). If $flag_{\mathsf{pe}} = \texttt{false}$, then both instances use $W = 0$ in the session key computation. If $flag_{\mathsf{pe}} = \texttt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game 6).

Overall, in stage t, the two partnered instances share the same values pss, KEM shared secret ss and W (equal to 0 or to $K_I = K_R$). They compute the same transcripts (tr_1, tr_2, tr_3), intermediary keys (k_0, k_1, k_2, dk_0, dk_1), and AE keys (hk_R, hk_I, hk'_R). Consequently, they compute the same session keys tk_I, tk_R (and pss).

Therefore, in Game 6 the advantage of the adversary is $\mathsf{adv}_6 = 0$ and Sound is true.

Collecting all the probabilities from Game 0 to Game 6 we have:

$$\begin{aligned}\mathsf{adv}^{\mathsf{sound}}_{\Pi}(\mathcal{A}) \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^\lambda}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\ & + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} + \mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}}\end{aligned}$$

A.2 Explicit Authentication

We decompose the proof according to the conditions necessary for FreshAuth to be true. These conditions explicit the minimal requirements that forbid the adversary from trivially winning the explicit authentication requirement. More specifically, the three subcases that we consider are listed below. Let π the targeted instance that the adversary wants to make accept maliciously. π is in accepting state in stage t.

- Case a): conditions a)1) and a)2) are fulfilled. The long term key of π's intended partner is not revealed (before π accepts) and the ephemeral KEM private key is not revealed (before the putative partner instance accepts).
- Case b)1): pss is not revealed (before the last accepting instance accepts) where pss is "safe" thanks to a previous (uncompromised) ephemeral KEM exchange.
- Case b)2): pss is not revealed (before the last accepting instance accepts) where pss is "safe" thanks to a previous (uncompromised) QKD run.
- Case b)3): pss is not revealed (before the last accepting instance accepts) where pss is "safe" because it has been initialised with a secret value uniformly drawn at random.

In case a), the strategy followed by the proof is the following. If the targeted honest instance has no origin partner (i.e. the adversary is active and interacts with its target), then the adversary is not able to compute a valid signature (and it cannot make the honest instance accept maliciously). If the targeted instance makes an ephemeral KEM exchange with an honest instance, then the run can be successfully completed and both instances have all the necessary means to fulfil the requirements implied by ExplicitAuth and to make the latter be true (the adversary is not allowed to reveal the ephemeral KEM private key).

In case b)1), the strategy is the following. If the targeted instance shares the same parameter pss as another instance when stage t starts, then ExplicitAuth is true thanks to the pss value (that the adversary is not allowed to reveal). The latter parameter is "safe" because, during a previous stage, an ephemeral KEM shared secret (unkown to the adversary) has been involved in the computation of pss.

In case b)2), the strategy is similar to case b)1) with the following difference. In stage t, pss is "safe" because, during a previous stage, an ephemeral quantum secret value (unkown to the adversary) has been involved in the computation of the parameter.

In case b)3), we assume that pss is initialised with a secret value uniformly drawn at random. We show that ExplicitAuth is true because pss remains uncompromised when stage t begins (again, as long as the parameter has not been revealed during previous stages).

Let E_i be the event that an adversary wins the explicit authentication experiment in Game i, and $\mathsf{adv}_i = \Pr[E_i]$.

Game 0. This game corresponds to the explicit authentication security experiment described by Figure 5 in Section 3. Therefore

$$\Pr[E_0] = \mathsf{adv}^{\mathsf{auth}}_{\Pi}(\mathcal{A}) = \mathsf{adv}_0$$

We split the proof according to the three aforementioned subcases. Therefore, we have:

$$\mathsf{adv}_0 \leq \mathsf{adv}^{a)}_0 + \mathsf{adv}^{b)1)}_0 + \mathsf{adv}^{b)2)}_0 + \mathsf{adv}^{b)3)}_0$$

FreshAuth a). We start with subcase a) in predicate FreshAuth.

Game$^{a)}$ 1. In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}^{a)}_0 \leq \mathsf{adv}^{a)}_1 + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}}$$

Game$^{a)}$ 2. In this game, the challenger aborts the experiment if either an ephemeral KEM public key epk or ciphertext ct is not unique among all instances and stages. There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}^{a)}_1 \leq \mathsf{adv}^{a)}_2 + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} \right)$$

Game$^{a)}$ 3. In this game, the challenger aborts the experiment if a KEM decapsulation failure occurs. That is, the challenger aborts the experiment if, on the one hand, the encapsulation done by an honest instance with input an ephemeral public key epk outputs a ciphertext ct and a shared secret ss, and, on the other hand, upon reception of ct by the honest partnered instance the decapsulation with the corresponding private key esk outputs a value $ss' \neq ss$. We have that

$$\mathsf{adv}^{a)}_2 \leq \mathsf{adv}^{a)}_3 + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}$$

Game$^{a)}$ 4. In this game, the challenger aborts the experiment if a collision occurs in Extract outputs among all instances and stages. Therefore

$$\mathsf{adv}^{a)}_3 \leq \mathsf{adv}^{a)}_4 + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}}$$

Game$^{a)}$ 5. In this game, the challenger aborts the experiment if a collision occurs in Expand outputs among all instances and stages. Therefore

$$\mathsf{adv}^{a)}_4 \leq \mathsf{adv}^{a)}_5 + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}}$$

Game$^{a)}$ 6. In this game, the challenger tries to guess the targeted instance π, its parent and the targeted stage t, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}^{a)}_5 \leq n_P \cdot n_S \cdot n_T \cdot \mathsf{adv}^{a)}_6$$

Game$^{a)}$ *7.* In this game, the challenger tries to guess which public key (i.e. which corresponding party) the signature received by π is verified with, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_6^{a)} \leq n_P \cdot \mathsf{adv}_7^{a)}$$

Game$^{a)}$ *8.* In this game, the challenger aborts the experiment if the adversary $\mathcal{A}$ produces a valid signature under an uncorrupted key for a message which was not output by an honest instance. This reduces straightforwardly to the security of the signature scheme SIG. We show how to build an adversary $\mathcal{B}_1$ against the EUF-CMA security of the signature scheme. This adversary is given a signing oracle. It behaves as follows. $\mathcal{B}_1$ sets $lpk_w = pk^*$ for some party w, with pk^* from the signature security experiment, and generates all other protocol parameters as specified. It uses its signing oracle in order to output honest signatures valid under lpk_w. If $\mathcal{A}$ submits a valid signature under lpk_w that was not output by an honest instance, then $\mathcal{B}_1$ uses these message and signature to break the EUF-CMA security of the signature scheme. Since this message was not signed before with the corresponding private key (because $\mathcal{B}_1$ did not query its signing oracle with this message to get the signature), then it is a valid forgery (i.e. $\mathcal{B}_1$ wins).

During a given stage, $\mathcal{B}_1$ makes at most 1 call to SIGN when it needs to output a signature for an honest instance (corresponding to party w) (i.e. $q_{\mathrm{SG}} \leq 1$).

$\mathcal{B}_1$ can answer any REVEALLCK query from $\mathcal{A}$ (since $\mathcal{B}_1$ manages the simulation) but for the private key corresponding to $lpk_w = pk^*$. However, in the context where $\mathcal{A}$ tries to compute a valid signature (on behalf of an honest instance owned by some party v) in order to make an honest instance π accept, the long term (signature) key of the putative partner party v cannot be revealed before the honest targeted instance π accepts, as otherwise $\mathsf{FreshAuth}$ becomes false. Therefore, if a query REVEALLCK(w) is issued before the targeted party accepts, then the simulation is stopped.

Taking into account that there are at most n_P parties, we have

$$\mathsf{adv}_7^{a)} \leq \mathsf{adv}_8^{a)} + \mathsf{adv}_{\mathsf{SIG}}^{\mathsf{euf\text{-}cma}}(1)$$

To that point, if the adversary corrupts the long term (signature) key of the targeted instance (π)'s intended partner before π accepts in stage t, then $\mathsf{FreshAuth}(\pi, t) = \texttt{false}$. If the adversary produces a valid signature under an uncorrupted signature key for a message that was not output by an honest instance, then the challenger aborts the experiment (by Game$^{a)}$ 8) and the targeted instance does not accept. Moreover, *epk* and *ct* do not repeat by Game$^{a)}$ 2 and collisions in H outputs are excluded by Game$^{a)}$ 1. Therefore, the adversary cannot reuse a signature computed by an honest instance from another sequence of messages.

Therefore, the signature (σ_I or σ_R) received by the accepting instance π (responder's or initiator's instance) must be sent by an honest instance π'. This signature is computed in particular from a hash of the ephemeral KEM public

key epk and ciphertext ct. This implies that epk and ct are shared by π' and π (since the latter accepts the signature as valid). That is, π' is an origin instance of π and, by unicity of epk and ct, π' is the unique origin instance of π.

By assumption, $\text{REVEALECK}(\pi, t)$ and $\text{REVEALECK}(\pi', t)$ have not been queried to that point. Therefore, the KEM shared secret ss is known only to π and π' (if no decapsulation failure occurs, which is the case by Game$^{a)}$ 3) unless the IND-1CCA security of KEM can be broken. Next, we exclude such a possibility.

The adversary can execute the protocol with π (the adversary can reveal the pss parameter). However, as said above, the adversary is not able to compute a valid signature and make π accept. Therefore, in the remaining of the games, we consider that there exists an honest instance π' that interacts with π and shares in particular the same value cid as π.

Game$^{a)}$ 9. In this game, the challenger tries to guess the partner instance π' and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_8^{a)} \leq n_S \cdot \mathsf{adv}_9^{a)}$$

Game$^{a)}$ 10. In this game, we replace (at the targeted instance π) the shared secret ss with a value $\tilde{ss}$ of same size uniformly drawn at random only if there exists an honest instance that shares the same $\mathsf{cid} = epk \| ct$ as π in stage t. From now on, any honest instance that shares in stage t the same $\mathsf{cid} = epk \| ct$ as π uses this value $\tilde{ss}$ (recall that, by Game$^{a)}$ 3, decapsulation failures are excluded).

We show how any adversary $\mathcal{A}$ that can detect this change can be used to construct an adversary $\mathcal{B}_2$ against the IND-1CCA security of KEM. Let $\tilde{\pi} \in \{\pi, \pi'\}$ be the initiator's instance and $\tilde{\pi}' \in \{\pi, \pi'\} \backslash \{\tilde{\pi}\}$ be the responder's instance. Let (pk^*, ct^*, ss^*) be the IND-1CCA challenge. $\mathcal{B}_2$ sets pk^* as the ephemeral public key to $\tilde{\pi}$ and ct^* as the ciphertext sent by $\tilde{\pi}'$, and the latter uses ss^* as the shared secret. If $\tilde{\pi}$ receives ct^*, then $\mathcal{B}_2$ sets ss^* as the shared secret in $\tilde{\pi}$. If $\tilde{\pi}$ receives a ciphertext $ct \neq ct^*$, then $\mathcal{B}_2$ queries its decapsulation oracle with ct. When $\mathcal{A}$ makes a guess whether the context is the real or the random one, then $\mathcal{B}_2$ uses this to answer the IND-1CCA challenge. If ss^* is the "real" shared secret, then $\mathcal{B}_2$ simulates Game$^{a)}$ 9. If ss^* is a random value, then $\mathcal{B}_2$ simulates Game$^{a)}$ 10.

During a given stage, $\mathcal{B}_2$ needs to make at most 1 call to DECAPS (i.e. $q_{\text{D}} \leq 1$).

Therefore, we have

$$\mathsf{adv}_9^{a)} \leq \mathsf{adv}_{10}^{a)} + \mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}1cca}}(1)$$

Game$^{a)}$ 11. In this game, we replace (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot) = \mathsf{Extract}(\cdot, \tilde{ss})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot)}$. We do a similar change at π' (which shares the same cid as well as the same shared secret ss as π).

Since $\tilde{ss}$ is sampled uniformly at random by Game$^{a)}$ 10 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$

to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_3$ queries its FUNC oracle to get k_1. However, to that point we have no guarantee that the instances π and π' share the same value pss. If not, the respective inputs dk_0 to FUNC are different for π and π' (because, by Games$^{a)}$ 4 and 5, no collision occurs in $\mathsf{Extract}$ or Expand during the computation of k_0 and dk_0). Therefore, we have to count for at most 2 queries to FUNC (i.e. $q_{\text{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{10}^{a)} \leq \mathsf{adv}_{11}^{a)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2)$$

Game$^{a)}$ 12. As explained in Game$^{a)}$ 11, to that point, we cannot ensure that the two instances π and π' compute the same value dk_0, input to compute k_1. It may happen that using distinct values dk_0 as input to $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$, the outputs be equal. We want to preclude such an event. The probability that such a collision occurs at π is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_1. Therefore, in this game, the challenger aborts the experiment if such a collision occurs.

Hence,

$$\mathsf{adv}_{11}^{a)} \leq \mathsf{adv}_{12}^{a)} + \frac{1}{2^{\ell_k}}$$

Game$^{a)}$ 13. In this game, the challenger aborts the experiment if a collision occurs between one of the two keys k_0 computed by π and π' respectively, and one of the two keys k_1 computed by π and π' respectively. Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Expand is keyed with distinct keys. Then, we will be able to safely replace Expand with a random function when keyed with a specific (unique) key, and still rely upon the CR property of Expand when keyed with another specific (unique) key.

By Game$^{a)}$ 11, the ℓ_k-bit long key k_1 computed respectively by π and π' is output by a random function. Therefore, we have that

$$\mathsf{adv}_{12}^{a)} \leq \mathsf{adv}_{13}^{a)} + \frac{4}{2^{\ell_k}}$$

Game$^{a)}$ 14. In this game, we replace (at the targeted instance π) the function $\mathsf{Expand}(k_1,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this, because k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{a)}$ 11).

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_4$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_4$ makes at most 7 calls to FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1 respectively (i.e. $q_{\text{FC}} \leq 7$).

In addition, we make a similar change at π' (which does share the same cid and shared secret ss as π). To that point we do not have the guarantee that π and π' do share the same value pss. Therefore, it may be possible that π' keys

Expand (to compute the AE keys and nonces) with a different value k_1 than that of used by π (by Game$^{a)}$ 12). Still, the key k_1 used by π' is also output by the random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (but, as said, possibly from a different input). Therefore, we replace Expand keyed with the key k_1 computed by π' with a random function $\mathsf{RF}'_{\mathsf{Expand}(k_1,\cdot)}$. This implies an additional loss related to the PRF security of Expand and the reduction makes also at most 7 calls to its FUNC oracle.

Therefore

$$\mathsf{adv}^{a)}_{13} \leq \mathsf{adv}^{a)}_{14} + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)$$

Game$^{a)}$ 15. By Game$^{a)}$ 12, if π and π' use distinct inputs dk_0, then they compute distinct values k_1 (i.e. the two instances use distinct random functions $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $\mathsf{RF}'_{\mathsf{Expand}(k_1,\cdot)}$). We want to exclude the event that, while π and π' use distinct random functions, they compute the same AE key (hk_I or hk_R) needed to compute the first AE ciphertext received by π (c_I or c_R). By Game$^{a)}$ 14, the probability that π computes this same AE key as π' in such a case is $1/2^{\ell_{hk}}$, where ℓ_{hk} is the bitlength of hk_I and hk_R.

Therefore, in this game the challenger aborts the experiment if any of these two such collisions occurs.

Hence

$$\mathsf{adv}^{a)}_{14} \leq \mathsf{adv}^{a)}_{15} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{a)}$ 16. In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it. This rule is applied only if there exists an honest instance that shares the same cid as π in stage t.

By Game$^{a)}$ 14, $\mathsf{Expand}(k_1,\cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_5$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_5$ makes at most 1 call to DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_5$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}^{a)}_{15} \leq \mathsf{adv}^{a)}_{16} + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1)$$

By Game$^{a)}$ 2, epk and ct are unique. Therefore, tr_0 and tr_1 are also unique (by Game$^{a)}$ 1). Consequently π' and π are the only instances that use tr_0 or tr_1 (depending on whether π is initiator or responder) as AE header to compute

and decrypt respectively an AE ciphertext.

Let us assume that π' does not share the value pss (at beginning of stage t) with π. Then, the two instances compute distinct AE parameters (by Game$^{a)}$ 4, Game$^{a)}$ 5, Game$^{a)}$ 12 and Game$^{a)}$ 15). Therefore, no honest instance ever computes the AE ciphertext received by π. If π rejects the AE ciphertext, then π does not accept. If π deems the received AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a valid forgery. Then, the challenger aborts the experiment (by Game$^{a)}$ 16).

This means that, if the challenger does not abort the experiment and π does not abort the run to that point, then π and π' must share the same value pss at beginning of stage t.

Game$^{a)}$ 17. In this game, the challenger aborts the experiment if the instance π' receives a valid final message (m_3 or m_4 depending on whether π' is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the message as π' to decrypt it. This rule is applied only if there exists an honest instance that shares the same cid as π' in stage t.

By Game$^{a)}$ 14, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}'_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk'_R) used to decrypt the ciphertext (c_I or c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_5$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_5$ makes at most 1 call to Dec in order to decrypt c_I or c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss as π', then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_5$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}^{a)}_{16} \leq \mathsf{adv}^{a)}_{17} + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1, 1)$$

To that point, π accepts upon reception of a valid signature (σ_I or σ_R depending on π's role). Moreover, the private signature key corresponding to that signature cannot be revealed before π accepts. Therefore, by Game$^{a)}$ 8, the data protected by this signature must be chosen by an honest instance π'. Among the signed data, there is the peer's identity. Therefore, π sets accordingly (first stage) or confirms (next stages) that $\pi.\mathsf{pid} = \pi'.\mathsf{parent}$ in accordance with the identity protected with σ_I or σ_R.

The signature protects also a hash of cid, and, by Game$^{a)}$ 1, no collision occurs in H outputs. This implies that if π deems the signature is valid, then the sender π' shares the same cid as π in stage t. That is, $\pi.\mathsf{cid}[t] = \pi'.\mathsf{cid}[t]$.

If π and π' do not use the same value pss at the beginning of stage t (i.e. $\pi.\mathsf{pss}[t-1] \neq \pi'.\mathsf{pss}[t-1]$), then the two instances compute distinct values dk_0 (because, by Games$^{a)}$ 4 and 5, no collision occurs in $\mathsf{Extract}$ and Expand during the computation of k_0 and dk_0). In turn, that implies, by Games$^{a)}$ 12 and 15,

that the two instances compute distinct values k_1 and distinct AE encryption keys (corresponding to the first AE ciphertext c_R or c_I). If the corresponding AE ciphertext is valid for π, then the challenger aborts the experiment (by Game$^{a)}$ 16). Since, π does indeed accept, this implies that the two instances do share the same value pss when stage t starts (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$).

The initiator's instance sends the ephemeral KEM public key and decapsulates whereas the responder's instance encapsulates. We have shown that both instances π and π' share the same $\mathsf{cid} = epk \| ct$ in stage t. Moreover, the KEM public key and ciphertext are unique by Game$^{a)}$ 2. This implies that both instances have distinct roles and agree on their respective roles (as otherwise, the two instances would disagree on either the public key or the ciphertext).

By Game$^{a)}$ 17, if π' receives a message (m_3 or m_4 depending on the instance's role) not computed by an honest instance with the same AE key as π', then the challenger aborts the experiment. Therefore, the message (m_3 or m_4) received by π' is computed by an honest instance (with the same key as π'). This message protects in particular a hash of the KEM public key and ciphertext epk and ct, and the identity of the sender. By Game$^{a)}$ 1, no collision occurs in H outputs. Since π' accepts the message (m_3 or m_4) as valid, this implies that π' shares epk and ct with the sender. Moreover, by Game$^{a)}$ 2, the pair (epk, ct) exchanged during each stage of the protocol run is unique. Consequently, the sender of the message is π and π' sets accordingly (first stage) or confirms (next stages) that $\pi'.\mathsf{pid} = \pi.\mathsf{parent}$ as carried in the received message.

If π' is an initiator's instance, then (as already said) it accepts upon reception of a valid ciphertext $m_4 = c'_R$ sent by π. This implies that π' shares with π all the inputs used to compute the intermediary keys $k_0, dk_0, \ldots, k_2$ (because, again, no collision occurs in H outputs by Game$^{a)}$ 1). Moreover, π' shares the same value $pss = \pi'.\mathsf{pss}[t-1]$ as π, hence the same random functions used to compute these intermediary keys. Consequently, π' and π do compute the same value pss at the end of stage t (i.e. $\pi'\mathsf{pss}[t] = \pi.\mathsf{pss}[t]$).

Let us consider now that π' is a responder's instance. As said, π accepts upon reception of a signature σ_R sent by π'. We can repeat the same reasoning as above (starting with the fact that σ_R ensures that π and π' share all the inputs used to compute the intermediary keys) and conclude that $\pi.\mathsf{pss}[t] = \pi'.\mathsf{pss}[t]$.

Therefore, $\mathsf{adv}^{a)}_{17} = 0$ and $\mathsf{ExplicitAuth}$ is true in Game$^{a)}$ 17.

Collecting all the probabilities from Game$^{a)}$ 0 to Game$^{a)}$ 17, we have:

$$
\begin{aligned}
\mathsf{adv}^{a)}_0 \leq\ & n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} \right) \\
& + n_T \cdot (n_P \cdot n_S)^2 \cdot \left(\frac{1}{2^{\ell_{hk}-1}} + \frac{5}{2^{\ell_k}} \right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} + n_P^2 \cdot n_S \cdot n_T \cdot \mathsf{adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG}}(1) \\
& + n_T \cdot (n_P \cdot n_S)^2 \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}1cca}}_{\mathsf{KEM}}(1) + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \\
& \quad + 4 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) \Big)
\end{aligned}
$$

FreshAuth b)1). Now, we proceed with case b)1) in predicate FreshAuth.

We start with six of the same first games as in the proof for case a). For simplicity, we only recall the corresponding security properties at stake:

$Game^{b)1)}$ *1.* Collision resistance of hash function H.

$Game^{b)1)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

$Game^{b)1)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

$Game^{b)1)}$ *4.* Collision resistance of Extract.

$Game^{b)1)}$ *5.* Collision resistance of Expand.

$Game^{b)1)}$ *6.* In this game, the challenger aborts the experiment if the key k_0 used by any instance to key Expand in stage $t' \geq 1$ is equal to the key k_1 or k_2 used to key Expand in stage t', or is equal to the key k_0, k_1 or k_2 used to key Expand in stages u, $t' < u < t$, or is equal to the key k_0 or k_1 used to key Expand in stage t.

Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (when keyed with a pseudo-random key). That is, in the latter case, we will replace Expand with a random function. Therefore, we must be able to distinguish the keys that key Expand when we want to leverage its CR property and the keys that key Expand which will eventually be replaced with a random function.

Let us first consider one instance. If the keys k_1, k_2 in stage t', and k_0, k_1, k_2 in stages u, $t' < u < t$, and k_0 and k_1 in stage t are chosen uniformly at random, then the probability that one of these keys collide with the key k_0 used in stage t' is $1/2^{\ell_k}$. Therefore, the probability that none of these keys (in stages t', and u, $t' < u < t$, and t) collide with k_0 used in stage t' is

$$\left(1 - \frac{1}{2^{\ell_k}}\right)^2 \times \left(1 - \frac{1}{2^{\ell_k}}\right)^{3(n_T-2)} \times \left(1 - \frac{1}{2^{\ell_k}}\right)^2 = \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}$$

because each instance executes at most n_T stages. Let $A_{\tilde{\pi}}$ be such an event for a given instance $\tilde{\pi}$.

Let us now consider the event that $A_{\tilde{\pi}}$ holds for all instances $\tilde{\pi}$: $\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}$. Let us consider the opposite event that, for at least one instance $\tilde{\pi}$, the key k_0 in stage t' be equal to at least another key k_1 or k_2 in stage t', or k_0, k_1 or k_2 in a stage u, $t' < u < t$, or k_0 or k_1 in stage t. The corresponding event is $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}$.

The challenger aborts the experiment if the latter event holds.

We have $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}} = \bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}$. Therefore, the corresponding probability is

$$\begin{aligned}
\Pr\left[\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}\right] &= \Pr\left[\bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}\right] \\
&\leq \sum_{\tilde{\pi}} \Pr\left[\overline{A_{\tilde{\pi}}}\right] \\
&= \sum_{\tilde{\pi}} (1 - \Pr[A_{\tilde{\pi}}]) \\
&= \sum_{\tilde{\pi}} \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \\
&= n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right)
\end{aligned}$$

because there are at most $n_P \cdot n_S$ instances.

If $\frac{1}{2^{\ell_k}}$ is close to 0, then $1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2} \simeq \frac{3 \cdot n_T - 2}{2^{\ell_k}}$. Hence

$$n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \simeq \frac{n_P \cdot n_S \cdot (3 \cdot n_T - 2)}{2^{\ell_k}}.$$

Therefore, we have

$$\mathsf{adv}_5^{b)1)} \leq \mathsf{adv}_6^{b)1)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right)$$

To that point, we have:

$$\begin{aligned}
\mathsf{adv}_0^{b)1)} \leq\ & \mathsf{adv}_6^{b)1)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} \\
& + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}
\end{aligned}$$

Game$^{b)1)}$ 7. In this game, the challenger aborts the experiment if the parameter *pss* used by any instance in stage t' as input to the function Extract is equal to at least one of the subsequent parameters *pss* used in input to Extract in stages u, $t' < u \leq t$.

This rule follows the same rationale as that of introduced in Game$^{b)1)}$ 6. Indeed, in subsequent games we will rely upon the CR property of the function Extract but also on the PRF security of Extract (keyed with a pseudo-random

key). Therefore, we must be able to distinguish when Extract is keyed with distinct keys. Then, we will be able to safely replace Extract with a random function when keyed with a specific (unique) key, and still rely upon the CR property of Extract when keyed with another specific (unique) key.

Let us consider first one instance. If the parameters pss used in stages u, $t' < u \leq t$, are chosen uniformly at random, then the probability that one of theses values collide with pss used in stage t' is $1/2^{\ell_{pss}}$. Therefore, the probability that none of these values be equal to pss used in stage t' is $\left(1 - 2^{-\ell_{pss}}\right)^{n_T - 1}$, because an instance executes at most n_T stages. Let $A_{\tilde{\pi}}$ be such an event for an instance $\tilde{\pi}$.

Let us now consider the event that $A_{\tilde{\pi}}$ holds for all instances $\tilde{\pi}$: $\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}$. Let us consider the opposite event that, for at least one instance $\tilde{\pi}$, the parameter pss used in stage t' be equal to at least a parameter pss used in a stage u, $t' < u \leq t$. The corresponding event is $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}$. The challenger aborts the experiment if the latter event holds.

We have $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}} = \bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}$. Therefore, the corresponding probability is

$$\begin{aligned}
\Pr\left[\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}\right] &= \Pr\left[\bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}\right] \\
&\leq \sum_{\tilde{\pi}} \Pr\left[\overline{A_{\tilde{\pi}}}\right] \\
&= \sum_{\tilde{\pi}} \left(1 - \Pr[A_{\tilde{\pi}}]\right) \\
&= \sum_{\tilde{\pi}} \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \\
&= n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right)
\end{aligned}$$

because there are at most $n_P \cdot n_S$ instances.

If $\frac{1}{2^{\ell_{pss}}}$ is close to 0, then $1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1} \simeq \frac{n_T - 1}{2^{\ell_{pss}}}$. Hence

$$n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \simeq \frac{n_P \cdot n_S \cdot (n_T - 1)}{2^{\ell_{pss}}}.$$

Therefore, we have that

$$\mathsf{adv}_6^{b)1)} \leq \mathsf{adv}_7^{b)1)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right)$$

$Game^{b)1)}$ *8.* In this game, the challenger aborts the experiment if a collision occurs in outputs of the chosen 2-universal hash function H_{ec}. These outputs are λ-bit long. Therefore, by definition of H_{ec} (cf. Definition 24), we have

$$\mathsf{adv}_7^{b)1)} \leq \mathsf{adv}_8^{b)1)} + \frac{n_P \cdot n_S \cdot n_T}{2^\lambda}$$

where the multiplicative factor $n_P \cdot n_S \cdot n_T$ comes from the fact that a new 2-universal hash function H_{ec} is chosen during each stage.

$Game^{b)1)}$ *9.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, the putative party and partner instance π', and the stage t' where no query $\textsc{RevealECK}(\pi, t')$ or $\textsc{RevealECK}(\pi', t')$ is issued, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_8^{b)1)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \mathsf{adv}_9^{b)1)}$$

$Game^{b)1)}$ *10.* In this game, we replace in stage t' the shared secret ss computed by π with a value $\tilde{ss}$ of same size uniformly drawn at random. We do the same for π' in stage t'. Recall that in stage t', π and π' are matching instances. Hence, they share the same KEM public key epk and ciphertext ct. Moreover, by $\text{Game}^{b)1)}$ 3, decapsulation failures are excluded. Consequently, π and π' share the same value ss in stage t'. In addition, no query $\textsc{RevealECK}(\pi, t')$ nor $\textsc{RevealECK}(\pi', t')$ is issued. Therefore, detecting the change implies an adversary $\mathcal{B}_0$ able to break the IND-CPA security of the KEM scheme. Here we rely upon the IND-CPA notion because the two instances π and π' are matching in stage t' (in particular, the ciphertext received by one instance is the value computed by the other). Therefore, we have

$$\mathsf{adv}_9^{b)1)} \leq \mathsf{adv}_{10}^{b)1)} + \mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}}$$

$Game^{b)1)}$ *11.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot) = \mathsf{Extract}(\cdot, \tilde{ss})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$.

Since $\tilde{ss}$ is sampled uniformly at random by $\text{Game}^{b)1)}$ 10 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_1$ queries its $\textsc{Func}$ oracle in order to get k_1. However, to that point we have no guarantee that the instances π and π' share the same value pss at the beginning of stage t'. If not, the respective inputs dk_0 to $\textsc{Func}$ are different for π and π' (because no collision occurs in $\mathsf{Extract}$ and Expand by $\text{Games}^{b)1)}$ 4 and 5).

Therefore, we have to count for at most 2 queries to $\textsc{Func}$ (i.e. $q_{\text{Fc}} \leq 2$).

Therefore

$$\mathsf{adv}_{10}^{b)1)} \leq \mathsf{adv}_{11}^{b)1)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2)$$

Game$^{b)1)}$ *12.* As explained in Game$^{b)1)}$ 11, to that point, we cannot ensure that the two instances compute the same value dk_0, input to compute k_1. It may happen that using distinct values dk_0 as input to $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$, the outputs be equal. We want to preclude such an event. The probability that such a collision occurs at π is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_1. Therefore, in this game, the challenger aborts the experiment if such a collision occurs.

Hence

$$\mathsf{adv}_{11}^{b)1)} \leq \mathsf{adv}_{12}^{b)1)} + \frac{1}{2^{\ell_k}}$$

Game$^{b)1)}$ *13.* In this game, the challenger aborts the experiment if the key k_0 computed by π' is equal to the key k_1 computed by π, when the two instances do not use the same value pss at beginning of stage t'. Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Expand is keyed with distinct keys. Then, we will be able to safely replace Expand with a random function when keyed with a specific (unique) key, and still rely upon the CR property of Expand when keyed with another specific (unique) key.

By Game$^{b)1)}$ 11, the ℓ_k-bit long key k_1 computed respectively by π is output by a random function. Therefore, we have that

$$\mathsf{adv}_{12}^{b)1)} \leq \mathsf{adv}_{13}^{b)1)} + \frac{1}{2^{\ell_k}}$$

Game$^{b)1)}$ *14.* In this game, we replace in stage t' (at the targeted instance π) the function $\mathsf{Expand}(k_1,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We apply the same change at π' if the latter shares the same value pss as π. We can do this, because k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{b)1)}$ 11).

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_2$ makes at most 7 calls to $\textsc{Func}$ to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1 respectively (i.e. $q_{\textsc{Fc}} \leq 7$).

Therefore

$$\mathsf{adv}_{13}^{b)1)} \leq \mathsf{adv}_{14}^{b)1)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)$$

Game$^{b)1)}$ *15.* By Game$^{b)1)}$ 12, if π and π' use distinct inputs dk_0, then they compute distinct values k_1. We want to exclude the event that, while π and π' use distinct keys k_1, they compute the same AE keys (hk_I or hk_R) needed to compute the first AE ciphertext received by π (c_I or c_R). By Game$^{b)1)}$ 14, the probability that π computes this same AE key as π' in such a case is $1/2^{\ell_{hk}}$, where ℓ_{hk} is the bitlength of hk_I and hk_R. Therefore, in this game the challenger aborts the experiment if such a collision occurs.

Hence

$$\mathsf{adv}_{14}^{b)1)} \leq \mathsf{adv}_{15}^{b)1)} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{b)1)}$ *16.* In this game, the challenger aborts the experiment if the instance π receives, in stage t', a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it.

By Game$^{b)1)}$ 14, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_3$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_3$ makes at most 1 call to DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value *pss* as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_3$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{15}^{b)1)} \leq \mathsf{adv}_{16}^{b)1)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Let us assume that, in stage t', π and π' use distinct values *pss*. Then, they compute compute distinct values dk_0 (because, by Games$^{b)1)}$ 4 and 5, no collision occurs in $\mathsf{Extract}$ and Expand outputs during the computation of k_0 and dk_0). In turn, this implies that the two instances output distinct values k_1 and distinct AE keys to compute and decrypt the first AE ciphertext received by π (by Game$^{b)1)}$ 15). If the first AE ciphertext received by π is invalid, then it is rejected by π (hence, it does not appear in the list $\mathsf{m_r}$ of the messages received by π).

π decrypts the received AE ciphertext (c_R or c_I depending on whether π is initiator or responder) with tr_0 or tr_1 as AE header. These values are computed from *epk* and *ct* which are unique (by Game$^{b)1)}$ 2). Therefore, tr_0 and tr_1 are also unique (by Game$^{b)1)}$ 1). π and π' match in stage t'. Therefore, π' is the only honest instance able to compute an AE ciphertext with such AE headers. In addition, π' does not use the same AE key to compute the AE ciphertext as π to decrypt it. Consequently, if π deems the received AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment by Game$^{b)1)}$ 16.

Since π and π' match in stage t' (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that the AE ciphertext is not a forgery, hence both instances do share the same value *pss* at the beginning of stage t'.

Game$^{b)1)}$ *17.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}(dk_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$. We can safely apply the change at both instances because (by the preceding games) we a have excluded the possibility for the two instances to use

distinct values dk_1.

Since dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{b)1)}$ 14) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_4$ to break the PRF security of $\mathsf{Extract}$.

During the stage, $\mathcal{B}_4$ makes at most 1 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 1$). Indeed, the two instances π and π' use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and by Game$^{b)1)}$ 8, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{b)1)}$ 14 and 16, and the same transcript tr_2 since, again, they are matching).

Therefore

$$\mathsf{adv}^{b)1)}_{16} \leq \mathsf{adv}^{b)1)}_{17} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

Game$^{b)1)}$ 18. In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Expand}(k_2,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{b)1)}$ 17) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of Expand.

We can safely apply this change because the two instances π and π' do compute the same value k_2. Indeed, the two instances use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{b)1)}$ 17). Moreover, the two instances use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and use the same AE parameters hk'_R and n'_R to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{b)1)}$ 14 and 16, and the same transcripts since, again, they are matching).

This implies also that, during the stage, $\mathcal{B}_5$ makes at most 3 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$). Therefore

$$\mathsf{adv}^{b)1)}_{17} \leq \mathsf{adv}^{b)1)}_{18} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, the two partnered instances π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{b)1)}$ 14). They share also tr_0 and c_R (because the instances match). Therefore, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \texttt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \texttt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{b)1)}$ 8).

This implies that π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and

inputs to compute the new value pss, hence they share the same value which is, in addition, indistinguishable from a value uniformly drawn at random (by $\text{Game}^{b)1)}$ 18).

Moreover, recall that neither $\text{RevealPSS}(\pi, t')$ nor $\text{RevealPSS}(\pi', t')$ is queried. Therefore, at the end of stage t', pss is indistinguishable from a value chosen uniformly at random and known only to π and π'.

Next, we will rely upon the PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ (keyed with pss). The next six games are repeated for stages $t'+1, \ldots, t-1$ (in order to make this appear, we denote each of these games as "$\text{Game}^{b)1)}$ $n+$"). Since the following six games rely upon the PRF security of Expand, $\mathsf{Extract}$ and the swap-PRF security of the latter, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake. We recall also that the instances π and π' are matching in stages u (hence, the changes described below apply to π' as well as π) and neither $\text{RevealPSS}(\pi, u)$ nor $\text{RevealPSS}(\pi', u)$ is queried for stages $u \in \{t'+1, \ldots, t-1\}$.

$\text{Game}^{b)1)}$ 19+. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\text{FC}} \leq 1$).

$\text{Game}^{b)1)}$ 20+. PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\text{FC}} \leq 2$).

$\text{Game}^{b)1)}$ 21+. PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\text{FC}} \leq 1$).

$\text{Game}^{b)1)}$ 22+. PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\text{FC}} \leq 7$).

$\text{Game}^{b)1)}$ 23+. PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\text{FC}} \leq 1$).

$\text{Game}^{b)1)}$ 24+. PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\text{FC}} \leq 3$).

Note that, in this game, we also rely upon $\text{Game}^{b)1)}$ 8 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \texttt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by $\text{Game}^{b)1)}$ 8.

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π' (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$). Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{18}^{b)1)} \leq\ & \mathsf{adv}_{24+}^{b)1)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \Big)\end{aligned}$$

Now, we consider stage t. We proceed with the next four games in order to show that the AE keys hk_I and hk'_R used in stage t to encrypt (and decrypt) the messages m_3 and m_4 respectively are indistinguishable from random values. Since the reasoning relies upon the PRF security of the functions already mentioned, once again we only indicate the security properties at stake. We recall that at the beginning of stage t the instances π and π' share the same value pss.

Game$^{b)1)}$ *25.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss. We replace $\mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ at π and π'. k_0 is now output by $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ (and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)1)}$ *26.* PRF security of Expand keyed with k_0. We replace $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ at π and π'. dk_0 is now output by $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)1)}$ *27.* PRF security of $\mathsf{Extract}$ keyed with dk_0. We replace $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ at π and π'. k_1 is now output by $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$ because the simulation may have to query its FUNC oracle twice if π and π' do not use the same input ss).

Game$^{b)1)}$ *28.* PRF security of Expand keyed with k_1. We replace $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ at π and π'. This change is applied at π' if π' shares the same cid as π in stage t. In that case, one query to $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ is enough to get hk_R, n_R and dk_1 for π and π'. However, we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I, hk'_R and n'_R. Therefore, it may happen that the random function be queried twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Now, we have also to take into account the possibility that π and π' do not use the same inputs to the random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (cf. Game$^{b)1)}$ 27) and yet output the same value k_1. If π and π' do not share, in stage t, the same parameters epk, ct, then FUNC may have to be queried once per instance to compute hk_R and n_R. That is, $q_{\mathrm{FC}} \leq 1 + 2 \times 6$.

hk_I and hk'_R are now output by $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

We will use $\mathsf{cid} = epk \| ct$ (and its uniqueness, by Game$^{b)1)}$ 2) only as a "binder" between π and its partner instance in stage t (the adversary might well issue a query REVEALECK(π, t) or REVEALECK(π', t)).

For Game$^{b)1)}$ 25 up to Game$^{b)1)}$ 28, we have:

$$\mathsf{adv}^{b)1)}_{24+} \leq \mathsf{adv}^{b)1)}_{28} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13)$$

Game$^{b)1)}$ *29.* In this game, the challenger aborts the experiment if π receives a valid final message (m_3 or m_4 depending on whether π is a responder's or an initiator's instance) not computed by an honest instance.

By Game$^{b)1)}$ 28, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (for π and any honest instance that shares the same cid and pss as π). Therefore,

the AE key (hk_I or hk'_R) used to decrypt the ciphertext (c_I or c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_6$ to break the INT-CTXT security of the AE function used to decrypt the ciphertext.

During the stage, $\mathcal{B}_6$ makes at most 1 call to DEC in order to decrypt c_I or c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same cid (and pss) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_6$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{28}^{b)1)} \leq \mathsf{adv}_{29}^{b)1)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Game$^{b)1)}$ *30.* In this game, the challenger aborts the experiment if π' receives a valid final message (m_3 or m_4 depending on whether π' is a responder's or an initiator's instance) not computed by an honest instance. This rule is applied only if π shares the same cid as π' in stage t.

By Game$^{b)1)}$ 28, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (for π and π' if the latter shares the same cid, hence the same key k_1, as π). Therefore, the AE key (hk_I or hk'_R) used to decrypt the ciphertext (c_I or c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_7$ to break the INT-CTXT security of the AE function used to decrypt the ciphertext.

During the stage, $\mathcal{B}_7$ makes at most 1 call to DEC in order to decrypt c_I or c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same cid (and pss) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_7$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{29}^{b)1)} \leq \mathsf{adv}_{30}^{b)1)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

To that point, in Game$^{b)1)}$ 30, π accepts upon reception of a ciphertext c_I or c'_R depending on π's role. By Game$^{b)1)}$ 29, the ciphertext must be sent by an honest instance (as otherwise the challenger aborts the experiment). The ciphertext protects also a hash of cid, and, by Game$^{b)1)}$ 1, no collision occurs in H outputs. This implies that if π deems the ciphertext is valid, then the sender instance shares the same cid as π in stage t.

By Games$^{b)1)}$ 25 to 28, only (honest) instances that share the same value pss as π at the beginning of stage t (and the same cid in stage t) can compute such a ciphertext. Therefore, the honest sending instance must be π' (by definition of π'). This implies also that $\pi.\mathsf{cid}[t] = \pi'.\mathsf{cid}[t]$.

Among the data protected in the received ciphertext, there is the peer's identity. Therefore, π can check that $\pi.\mathsf{pid} = \pi'.\mathsf{parent}$ in accordance with the identity carried in c_I or c'_R.

The initiator's instance sends the ephemeral KEM public key and decapsulates whereas the responder's instance encapsulates. We have shown that both

instances π and π' share the same $\mathsf{cid} = epk \| ct$ in stage t. Moreover, the KEM public key and ciphertext are unique by Game$^{b)1)}$ 2. This implies that both instances have distinct roles and agree on their respective roles (as otherwise, the two instances would disagree on either the public key or the ciphertext).

In stage $t-1$ (i.e. at the beginning of stage t), π and π' share the same value pss. Moreover, the two instances share also the same cid in stage t. Hence, by Game$^{b)1)}$ 28, they use the same random function to compute the keys hk_I and hk'_R, which are indistinguishable from random and known only to instances that share the same pss and cid. By Game$^{b)1)}$ 30, if π' accepts a message (m_3 or m_4 depending on the instance's role) not computed by an honest instance, then the challenger aborts the experiment. Therefore, the message (m_3 or m_4) received by π' is computed by an honest instance. This message protects in particular a hash of the KEM public key and ciphertext epk and ct, and the identity of the sender. By Game$^{b)1)}$ 1, no collision occurs in H outputs. Since π' accepts the message (m_3 or m_4) as valid, this implies that π' shares epk and ct with the sender. Moreover, by Game$^{b)1)}$ 2, the pair (epk, ct) exchanged during each stage of the protocol run is unique. Consequently, the sender of the message is π and π' confirms that $\pi'.\mathsf{pid} = \pi.\mathsf{parent}$ as carried in the message.

Therefore, $\mathsf{adv}_{30}^{b)1)} = 0$ and $\mathsf{ExplicitAuth}$ is true in Game$^{b)1)}$ 30.

Note that the proof for case b)1) does not rely upon the security of the signature scheme because we assume that the targeted instance π has a matching instance in (some previous) stage t'. The latter is realistic as shown by the proof of case a), which in turn relies in particular upon the security of the signature scheme. Note also that, according to the $\mathsf{FreshAuth}$ predicate for case b)1), the adversary is not allowed to get pss before the *last* accepting instance has actually accepted. Otherwise, there would be no guarantee that m_4 is computed by an honest instance. For instance, if π is an initiator's instance and the adversary can query $\textsc{RevealPSS}(\pi, t-1)$ once π' accepts, then the adversary can compute the AE parameters used to output m_4.

Collecting all the probabilities from Game$^{b)1)}$ 0 to Game$^{b)1)}$ 30, we have:

$$
\begin{aligned}
\mathsf{adv}_0^{b)1)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^\lambda}\right) \\
& + n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \frac{1}{2^{\ell_{hk}-1}} + \frac{1}{2^{\ell_k - 1}}\right) \\
& + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2) + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(2) \\
& \quad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(13) + 6 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1) + (2n_T - 3) \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) \\
& \quad + (n_T - 1) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) \\
& \quad\quad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)\Big)\Big)
\end{aligned}
$$

FreshAuth b)2). Now, we proceed with case b)2) in predicate FreshAuth.

The sequence of games corresponding to this case is close to that of the previous case. The main difference lies on the fact that in this case the *pss* value used in stage t is "safe" because of an uncompromised ephemeral quantum secret computed in stage t'. As in case b)1), we will use the uniqueness of cid to bind the targeted instance and its partner instance.

The first eight games are the same as in case b)1).

$Game^{b)2)}$ *1.* Collision resistance of hash function H.

$Game^{b)2)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

$Game^{b)2)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Here, we do not rely upon the (IND-CPA or IND-CCA) security of the KEM scheme. However, its correctness is required as two instances sharing the same cid must compute the same KEM shared secret *ss* (the latter is involved in the computation of the intermediary secret values and keys).

$Game^{b)2)}$ *4.* Collision resistance of Extract.

$Game^{b)2)}$ *5.* Collision resistance of Expand.

$Game^{b)2)}$ *6.* Collision probability for the keys k_0, k_1, k_2 when pseudo-random. The challenger aborts the experiment if a parameter k_0 or k_1 used in stage t' is equal to one of the following parameters: k_2 in stage t', or k_0, k_1, k_2 in a stage u, $t' < u < t$, or k_0, k_1 in stage t. Following a reasoning similar to that of the subcase FreshAuth b)1), we have:

$$\mathsf{adv}_5^{b)2)} \leq \mathsf{adv}_6^{b)2)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)}\right)$$

$Game^{b)2)}$ *7.* Collision probability for the values *pss* when pseudo-random.

$Game^{b)2)}$ *8.* Collision resistance of H_{ec}.

To that point, we have:

$$\begin{aligned}\mathsf{adv}_0^{b)2)} \leq\ & \mathsf{adv}_8^{b)2)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} \\ & + n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T-1}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right)\end{aligned}$$

Game$^{b)2)}$ *9.* In this game, the challenger aborts the experiment if an honest instance receives an AE ciphertext valid under a key distinct from the key used by the sender to compute the ciphertext. This reduces straightforwardly to the key-committing security of the AE scheme. Therefore

$$\mathsf{adv}_8^{b)2)} \leq \mathsf{adv}_9^{b)2)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{key\text{-}cmt}}$$

Game$^{b)2)}$ *10.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, the putative party and partner instance π', and the stage t' where no query $\textsc{RevealEQK}(\pi, t')$ nor $\textsc{RevealEQK}(\pi', t')$, where $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi'.\mathsf{eqk}[t'] \neq \perp$, is issued, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_9^{b)2)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \mathsf{adv}_{10}^{b)2)}$$

To that point, in stage t', if π and π' use distinct values pss, then they compute distinct values dk_0 (because no collision occurs in $\mathsf{Extract}$ and Expand outputs during the computation of k_0 and dk_0, by Games$^{b)2)}$ 4 and 5. In turn, this implies that the two instances output distinct values k_1 and distinct AE keys to compute and decrypt the first AE ciphertext received by π. Therefore, either the first AE ciphertext received by π is invalid and rejected by π (hence, it does not appear in the list $\mathsf{m_r}$ of the messages received by π), or the ciphertext is valid but the challenger aborts the experiment by Game$^{b)2)}$ 9. Since π and π' match in stage t' (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that both instances do share the same value pss at the beginning of stage t'.

Moreover, since π and π' match in stage t', this implies that epk and ct are shared by π' and π. By unicity of epk and ct, π' is the unique origin instance of π. This implies in addition, that the plaintext data corresponding to c_R and c_I are also shared by the two instances (i.e. the (classical) data related to the quantum key agreement are exchanged through an authenticated channel). Moreover, by assumption, we have that $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi'.\mathsf{eqk}[t'] \neq \perp$ (i.e. $flag_{\mathsf{pe}} = \mathtt{true}$ in stage t'). Therefore, the two instances do compute an actual ephemeral quantum secret. Consequently, we can apply Theorem 1 and safely replace $\pi.\mathsf{eqk}[t']$ and $\pi'.\mathsf{eqk}[t']$ with a value of same size uniformly drawn at random.

Game$^{b)2)}$ *11.* In this game, we replace the ephemeral quantum secret computed in stage t' by the targeted instance π and its partner instance π' with a value $\tilde{K}$ of same size uniformly drawn at random.

Tomamichel and Leverrier [109] prove the security of the BB84 protocol based on the following experiment. On the one hand (real context), the real BB84 protocol is executed. On the other hand (ideal context), a BB84 run is executed and the output key is replaced with a value of same length uniformly drawn at random. The maximum probability of distinguishing the context is $p_\Delta = (1 + \epsilon_\Delta)/2$ (see Section 4.7). By Game$^{b)2)}$ 8, collisions in $\mathsf{H_{ec}}$ are excluded. Therefore, we can remove the corresponding probability (correctness component) $\epsilon_{ec} = 2^{-\lambda}$ from ϵ_Δ and consider instead the secrecy component only $\epsilon_s = \epsilon_\Delta - \epsilon_{ec}$, as well

as the probability $p_s = (1 + \epsilon_s)/2$.

Let F be the event that the adversary detects the change made in this game. We have

$$\begin{aligned}
\mathsf{adv}_{10}^{b)2)} &\leq \mathsf{adv}_{11}^{b)2)} + \Pr[\mathcal{A} \text{ wins} \mid \text{real QKD context}] \\
&\quad - \Pr[\mathcal{A} \text{ wins} \mid \text{ideal QKD context}] \\
&\leq \mathsf{adv}_{11}^{b)2)} + \Pr[F] \\
&\leq \mathsf{adv}_{11}^{b)2)} + 2 \cdot p_s - 1 \\
&= \mathsf{adv}_{11}^{b)2)} + \epsilon_s
\end{aligned}$$

Remark 5. The classical data related to the QKD phase (in particular the r-bit syndrome Z and the λ-bit digest T) is encrypted. Therefore, an outside observer does not have access to these values. Consequently, we could remove from the quantity $\epsilon_{pa}(\nu)$ the contribution implied by the knowledge of Z and T. That is, $\epsilon_{pa}(\nu)$ could be reduced by a factor $\sqrt{2^{r+\lambda}}$ and the overall security bound ϵ_Δ of the QKD phase decreased (see the proof of Proposition 11 in [109]). However, our security model allows the adversary to reveal in this game the parameters (internal state pss, KEM shared secret ss) used to compute the encryption key. Consequently, we keep $\epsilon_{pa}(\nu)$ (as well as ϵ_Δ and ϵ_s) as expressed in Section 4.7.

To that point, in stage t', π and π' use $\tilde{K}$ to key $\mathsf{Extract}^{\mathsf{swap}}$ in order to compute k_2.

Game$^{b)2)}$ 12. In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{K}, \cdot) = \mathsf{Extract}(\cdot, \tilde{K})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$.

Since $\tilde{K}$ is uniformly drawn at random by Game$^{b)2)}$ 11 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_0$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_0$ makes at most 1 call to Func (i.e. $q_{\mathrm{Fc}} \leq 1$).

Therefore

$$\mathsf{adv}_{11}^{b)2)} \leq \mathsf{adv}_{12}^{b)2)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1)$$

Game$^{b)2)}$ 13. In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$ (by Game$^{b)2)}$ 12) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_1$ makes at most 3 calls to Func (i.e. $q_{\mathrm{Fc}} \leq 3$).

Therefore

$$\mathsf{adv}_{12}^{b)2)} \leq \mathsf{adv}_{13}^{b)2)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3)$$

To that point, at the end of stage t', pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π'. Moreover, recall that neither RevealPSS(π, t') nor RevealPSS(π', t') is queried.

The last games of this case are exactly the same as the last games of case b)1) (i.e. Game$^{b)1)}$ 19+ to Game$^{b)1)}$ 30). They correspond to the changes applied in stages $u \in \{t'+1, \ldots, t-1\}$ (to the instances π and π' matching in stages u) and finally in stage t.

Game$^{b)2)}$ 14+. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)2)}$ 15+. PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)2)}$ 16+. PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)2)}$ 17+. PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{b)2)}$ 18+. PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)2)}$ 19+. PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{b)2)}$ 8 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \texttt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{b)2)}$ 8.

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π' (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$).

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{13}^{b)2)} \leq\ & \mathsf{adv}_{19+}^{b)2)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \Big)\end{aligned}$$

Next, we consider stage t.

Game$^{b)2)}$ 20. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss. We replace $\mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ at π and π'. k_0 is now output by $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ (and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)2)}$ *21.* PRF security of Expand keyed with k_0. We replace $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ at π and π'. dk_0 is now output by $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)2)}$ *22.* PRF security of $\mathsf{Extract}$ keyed with dk_0. We replace $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ at π and π'. k_1 is now output by $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$ because the simulation may have to query its FUNC oracle twice if π and π' do not use the same input ss).

Game$^{b)2)}$ *23.* PRF security of Expand keyed with k_1. We replace $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ at π and π'. This change is applied at π' if π' shares the same cid as π in stage t. In that case, one query to $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ is enough to get hk_R, n_R and dk_1 for π and π'. However, we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I, hk'_R, n'_R. Therefore, it may happen that the random function be queried twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Now, we have also to take into account the possibility that π and π' do not use the same inputs to the random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (cf. Game$^{b)2)}$ 22) and yet output the same value k_1. If π and π' do not share, in stage t, the same parameters epk, ct, then FUNC may have to be queried once per instance to compute hk_R and n_R. That is, $q_{\mathrm{FC}} \leq 1 + 2 \times 6$.

hk_I and hk'_R are now output by $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

We will use $\mathsf{cid} = epk \| ct$ (and its uniqueness, by Game$^{b)2)}$ 2) only as a "binder" between π and its partner instance in stage t (the adversary might well issue a query REVEALECK(π, t) or REVEALECK(π', t)).

Game$^{b)2)}$ *24.* INT-CTXT security of AE keyed with hk_I or hk'_R (abort if π receives a valid final message (m_3 or m_4) not computed by an honest instance).

Game$^{b)2)}$ *25.* INT-CTXT security of AE keyed with hk'_R or hk_I (abort if π' receives a valid final message (m_4 or m_3) not computed by an honest instance). This rule is applied only if π' shares the same cid as π in stage t.

To that point, we have

$$\begin{aligned}\mathsf{adv}_{19+}^{b)2)} \leq \mathsf{adv}_{25}^{b)2)} &+ \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(13) \\ &+ 4 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)\end{aligned}$$

The reasoning of case b)2) is the same as in case b)1) and we do not repeat it here. We can conclude that $\mathsf{adv}_{25}^{b)2)} = 0$ and $\mathsf{ExplicitAuth}$ is true in Game$^{b)2)}$ 25.

Collecting all the probabilities from Game$^{b)2)}$ 0 to Game$^{b)2)}$ 25, we have:

$$\begin{aligned}
\mathsf{adv}_0^{b)2)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \epsilon_s\right) \\
& + n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T-1}\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} + \mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13) + 4 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) \\
& \quad + n_T \cdot \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + (n_T - 2) \cdot \Big(2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big) \\
& \quad + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)\Big)\Big)
\end{aligned}$$

FreshAuth b)3). Now, we proceed with case b)3) in predicate FreshAuth.

The sequence of games corresponding to this case is close to that of case b)1). The main difference lies on the fact that in this case the *pss* value used in stage t is "safe" because the initial value of *pss* is uncompromised and *pss* remains uncompromised when stage t begins.

Several first games are similar to case b)1). Consequently, we do not expand.

Game$^{b)3)}$ *1.* Collision resistance of hash function H.

Game$^{b)3)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

Game$^{b)3)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Here, we do not rely upon the (IND-CPA or IND-CCA) security of the KEM scheme. However, its correctness is required as two instances sharing the same $\mathsf{cid} = epk \| ct$ must compute the same KEM shared secret *ss* (the latter is involved in the computation of the intermediary secret values and keys).

Game$^{b)3)}$ *4.* Collision resistance of H_{ec}.

To that point, we have

$$\begin{aligned}
\mathsf{adv}_0^{b)3)} \leq\ & \mathsf{adv}_4^{b)3)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\
& + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right)
\end{aligned}$$

$Game^{b)3)}$ *5.* In this game, the challenger tries to guess the targeted instance π, its parent, the putative partner party and instance π', and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_4^{b)3)} \leq (n_P \cdot n_S)^2 \cdot \mathsf{adv}_5^{b)3)}$$

$Game^{b)3)}$ *6.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(pss, \cdot) = \mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$.

Since pss is initialised with a value uniformly drawn at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_0$ to break the swap-PRF security of $\mathsf{Extract}$. During a given stage, $\mathcal{B}_0$ queries its FUNC oracle in order to compute k_0 (i.e. $q_{\mathrm{FC}} \leq 1$).

We do a similar change at π' if the latter use the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_5^{b)3)} \leq \mathsf{adv}_6^{b)3)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1)$$

$Game^{b)3)}$ *7.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$. We can do this because k_0 computed by π is now uniformly drawn at random by $\mathrm{Game}^{b)3)}$ 6. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the PRF security of Expand. During a given stage, $\mathcal{B}_1$ makes at most two queries to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 2$).

We apply a similar change at π' if the latter use the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_6^{b)3)} \leq \mathsf{adv}_7^{b)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2)$$

$Game^{b)3)}$ *8.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$. We can do this because dk_0 computed by π is now uniformly drawn at random by $\mathrm{Game}^{b)3)}$ 7. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the PRF security of $\mathsf{Extract}$. During a given stage, $\mathcal{B}_2$ makes at most one query to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 1$).

We apply a similar change at π' if the latter use the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_7^{b)3)} \leq \mathsf{adv}_8^{b)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

$Game^{b)3)}$ *9.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this because k_1 computed by π is now uniformly drawn at random by $\mathrm{Game}^{b)3)}$ 8. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand. During a given stage, $\mathcal{B}_3$ makes at most 7 queries to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 7$).

We apply a similar change at π' if the latter use the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_8^{b)3)} \leq \mathsf{adv}_9^{b)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)$$

Game$^{b)3)}$ *10.* If π and π' do not share the same value pss, they compute distinct keys k_0, dk_0, k_1, and hk_R and hk_I unless a collision occurs when π and π' choose each of the latter parameters. We want to exclude such events. Therefore, the challenger aborts the experiment if any such collision occurs.

When π and π' do not share pss, the probability that π computes the same value k_0 (or k_1) as π' is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_0 and k_1. The probability that π computes the same value dk_0 as π' is $1/2^{\ell_{dk}}$ where ℓ_{dk} is the bitlength of dk_0. The probability that π computes the same value hk_R (or hk_I, both values being uniformly drawn at random by Game$^{b)3)}$ 9) as π' is $1/2^{\ell_{hk}}$ where ℓ_{hk} is the bitlength of hk_R and hk_I.

Therefore

$$\mathsf{adv}_9^{b)3)} \leq \mathsf{adv}_{10}^{b)3)} + \frac{1}{2^{\ell_k-1}} + \frac{1}{2^{\ell_{dk}}} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{b)3)}$ *11.* In this game, the challenger aborts the experiment if the instance π receives during the first stage a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it.

By Game$^{b)3)}$ 9, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_4$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_4$ makes at most 1 call to Dec in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_4$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{10}^{b)3)} \leq \mathsf{adv}_{11}^{b)3)} + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1)$$

To that point, if π and π' do not share the same value pss at the beginning of the first stage, then they compute distinct AE keys corresponding to the first AE ciphertext received by π (which is sent by π' since both instances match in stage 1), unless a collision occurs in the computation of one of the intermediary keys or in the computation of the AE keys. Such events are excluded by Game$^{b)3)}$ 10. Therefore, either π and π' compute the same AE key (which implies that they share the same value pss), or the first AE ciphertext received by π is invalid and rejected by π (hence, it does not appear in the list $\mathsf{m_r}$ of the messages

received by π), or the ciphertext is valid but the challenger aborts the experiment by Game$^{b)3)}$ 11. Indeed, the AE header (tr_0 or tr_1) used by π to decrypt the AE ciphertext is computed from epk and ct which are unique (by Game$^{b)3)}$ 2). Therefore, the AE header is also unique (by Game$^{b)3)}$ 1). This means that this AE header is never used by any other honest instance (but π') to compute an AE ciphertext (and, as said, π' uses an AE key different from that of used by π). This allows leveraging the INT-CTXT security of the AE function.

Since π and π' match in stage 1 (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that both instances do share the same value pss at the beginning of stage 1.

Game$^{b)3)}$ 12. In this game, we replace in stage 1 (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}(dk_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$.

Since dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{b)3)}$ 9) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of $\mathsf{Extract}$.

During the stage, $\mathcal{B}_5$ makes at most 1 call to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 1$). Indeed, the two instances π and π' use the same input W because they share the ciphertext c'_R (since they are matching in stage 1 by assumption) and by Game$^{b)3)}$ 4, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{b)3)}$ 9 and 11, and the same transcript tr_2 since, again, they are matching).

Therefore

$$\mathsf{adv}_{11}^{b)3)} \leq \mathsf{adv}_{12}^{b)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

Game$^{b)3)}$ 13. In this game, we replace in stage 1 (at the targeted instance π and its matching instance π') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{b)3)}$ 12) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_6$ to break the PRF security of Expand.

We can safely apply this change because the two instances π and π' do compute the same value k_2. Indeed, the two instances use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{b)3)}$ 12). Moreover, the two instances use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and by Game$^{b)3)}$ 4, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{b)3)}$ 9 and 11, and the same transcripts since, again, they are matching).

This implies also that, during the stage, $\mathcal{B}_6$ makes at most 3 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$).

Therefore

$$\mathsf{adv}_{12}^{b)3)} \leq \mathsf{adv}_{13}^{b)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, the two partnered instances π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{b)3)}$ 9). They share also tr_0 and c_R (because the instances match). Therefore, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \mathtt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \mathtt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{b)3)}$ 4).

This implies that π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ (by Game$^{b)3)}$ 13) known only to them, and the same inputs to compute the new value pss. Hence they share this value. Moreover, neither RevealPSS$(\pi, 0)$ nor RevealPSS$(\pi', 0)$ is queried, and RevealPSS$(\pi, 1)$ and RevealPSS$(\pi', 1)$ are not queried either. Therefore, at the end of stage 1, pss is indistinguishable from a value chosen uniformly at random and known only to π and π'.

Now, we guess the stage t.

Game$^{b)3)}$ 14. In this game, the challenger tries to guess the stage t and aborts if it does not succeed. Therefore

$$\mathsf{adv}^{b)3)}_{13} \leq n_T \cdot \mathsf{adv}^{b)3)}_{14}$$

In the subsequent games, we use the swap-PRF security of Extract and the PRF security of Extract and Expand to eventually argue that the value pss at the beginning of stage t is random and known only to π and π'. Since the reasoning relies upon security properties already mentioned, once again we only indicate the properties at stake.

The next six games are repeated for stages u, $1 < u < t$ (in order to make this appear, we denote each of these games as "Game$^{b)3)}$ n+"). The corresponding changes are done at π and its matching instance π' in stages u.

In stages u, $1 < u < t$, π and π' share the inputs to the different random functions (replacing Extract and Expand) because they match, compute the same intermediary keys (hence use the same random functions), and there is no collision in H_{ec} outputs (by Game$^{b)3)}$ 4). Moreover, recall that RevealPSS(π, u) and RevealPSS(π', u) are not queried where π and π' match in stages u, $1 < u < t$.

Game$^{b)3)}$ 15+. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)3)}$ 16+. PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)3)}$ *17+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)3)}$ *18+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{b)3)}$ *19+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)3)}$ *20+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π' (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$).

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{14}^{b)3)} \leq\ & \mathsf{adv}_{20+}^{b)3)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + 2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) \\ & \qquad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)\Big)\end{aligned}$$

Next, we consider stage t. The corresponding games are the same as those made in case b)2) for stage t.

Game$^{b)3)}$ *21.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss. We replace $\mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ at π and π'. k_0 is now output by $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ (and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)3)}$ *22.* PRF security of Expand keyed with k_0. We replace $\mathsf{Expand}(k_0,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ at π and π'. dk_0 is now output by $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)3)}$ *23.* PRF security of $\mathsf{Extract}$ keyed with dk_0. We replace $\mathsf{Extract}(dk_0,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ at π and π'. k_1 is now output by $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$ because the simulation may have to query its FUNC oracle twice if π and π' do not use the same input ss).

Game$^{b)3)}$ *24.* PRF security of Expand keyed with k_1. We replace $\mathsf{Expand}(k_1,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ at π and π'. This change is applied at π' if π' shares the same cid as π in stage t. In that case, one query to $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ is enough to get hk_R, n_R and dk_1 for π and π'. However, we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I, hk'_R, n'_R. Therefore, it may happen that the random function be queried twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Now, we have also to take into account the possibility that π and π' do not

use the same inputs to the random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (cf. Game$^{b)3)}$ 23) and yet output the same value k_1. If π and π' do not share, in stage t, the same parameters epk, ct, then FUNC may have to be queried once per instance to compute hk_R and n_R. That is, $q_{\mathrm{FC}} \leq 1 + 2 \times 6$.

hk_I and hk'_R are now output by $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

We will use $\mathsf{cid} = epk \| ct$ (and its uniqueness, by Game$^{b)3)}$ 2) only as a "binder" between π and its partner instance in stage t (the adversary might well issue a query REVEALECK(π, t) or REVEALECK(π', t)).

Game$^{b)3)}$ 25. INT-CTXT security of AE keyed with hk_I or hk'_R (abort if π receives a valid final message (m_3 or m_4) not computed by an honest instance).

Game$^{b)3)}$ 26. INT-CTXT security of AE keyed with hk'_R or hk_I (abort if π' receives a valid final message (m_4 or m_3) not computed by an honest instance). This rule is applied only if π' shares the same cid as π in stage t.

To that point, we have

$$\begin{aligned}\mathsf{adv}^{b)3)}_{20+} \leq \mathsf{adv}^{b)3)}_{26} &+ \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13) \\ &+ 4 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1)\end{aligned}$$

The reasoning of case b)3) is the same as in case b)2) and we do not repeat it here. We can conclude that $\mathsf{adv}^{b)3)}_{26} = 0$ and $\mathsf{ExplicitAuth}$ is true in Game$^{b)3)}$ 26.

Collecting all the probabilities from Game$^{b)3)}$ 0 to Game$^{b)3)}$ 26, we have:

$$\begin{aligned}\mathsf{adv}^{b)3)}_0 \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) + (n_P \cdot n_S)^2 \cdot \left(\frac{1}{2^{\ell_k - 1}} + \frac{1}{2^{\ell_{dk}}} + \frac{1}{2^{\ell_{hk}-1}}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} \\ & + (n_P \cdot n_S)^2 \cdot \Big(n_T \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13)\Big) \\ & \quad + (4 \cdot n_T + 2) \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) \\ & \quad + (n_T(n_T - 1) + 1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2)\Big) \\ & \quad + (2 \cdot n_T(n_T - 2) + 2) \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) \\ & \quad + (n_T(n_T - 2) + 1) \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\Big)\end{aligned}$$

A.3 Key Indistinguishability

In order to prove that this property is ensured, we consider the different conditions related to the $\mathsf{FreshKey}$ predicate. That is, we prove that each of these conditions are sufficient to forbid the adversary from (trivially) winning the experiment. Let π be the accepting instance that the adversary targets in stage t.

Conditions a) and b) allow the adversary to get the session key computed in

stage t by π or by a matching instance π' (if the latter exists). Consequently, the adversary wins straightforwardly (with overwhelming probability) the experiment. As thwarting such scenarios does not depend upon a security property ensured by one of the underlying cryptographic functions, we do not elaborate more on these cases.

We focus on the three last conditions.

- Case c): we consider the event that the tested instance in stage t has a matching instance (i.e. a partner). At least one of the conditions c)1) to c)5) is fulfilled. We decompose the security proof into as many corresponding subcases.
  - Case c)1): the ephemeral KEM private key is not revealed in stage t.
  - Case c)2): the ephemeral quantum secret value is not revealed in stage t.
  - Case c)3): pss is not revealed where pss is "safe" thanks to a previous (uncompromised) ephemeral KEM exchange.
  - Case c)4): pss is not revealed where pss is "safe" thanks to a previous (uncompromised) QKD phase.
  - Case c)5): pss is not revealed where pss is "safe" because it has been initialised with a secret value uniformly drawn at random.
- Case d): we consider the event that the tested instance in stage t has no origin instance. At least one of the conditions d)1) to d)5) holds. We decompose the security proof into as many corresponding subcases.
  - Case d)1): if the tested instance does not share its cid (in stage t) with any other honest instance, then the asymmetric long-term key of the peer is not revealed (before the tested instance accepts in stage t).
  - Case d)2): if the tested instance shares its cid (in stage t) with an honest instance, then the ephemeral KEM private key is not revealed (before the tested instance accepts).
  - Case d)3): pss is not revealed (until the tested instance accepts) where pss is "safe" thanks to a previous (uncompromised) ephemeral KEM exchange.
  - Case d)4): pss is not revealed (until the tested instance accepts) where pss is "safe" thanks to a previous (uncompromised) QKD phase.
  - Case d)5): pss is not revealed where pss is "safe" because it has been initialised with a secret value uniformly drawn at random.
- Case e): we consider the event that the tested instance π has an origin instance π' in stage t but is not itself an origin instance of π'.
  - Case e)1): essentially, we forbid the adversary from revealing the private signature key of the tested instance's putative partner (before the tested instance accepts), and when the tested instance is responder we incorporate a feature related to the fact that the tested instance computes the last message of the run.
  - Case e)2): the adversary is not allowed to reveal the KEM shared secret in stage t.
  - Case e)3): the ephemeral quantum secret value is not revealed in stage t.
  - Case e)4): pss is not revealed where pss is "safe" thanks to a previous (uncompromised) ephemeral KEM exchange.

- Case e)5): *pss* is not revealed where *pss* is "safe" thanks to a previous (uncompromised) QKD phase.
- Case e)6): *pss* is not revealed where *pss* is "safe" because it has been initialised with a secret value uniformly drawn at random.

In case c), the strategy is the following. First we do not have to deal with a possible impersonation since the tested instance has a partner instance (more precisely, it may be possible that m_4, sent by the tested (responder's) instance, is not received by its origin instance, but we neglect this possibility as it does not change the reasoning, and we consider that π has indeed a matching instance). The three secret values involved in the session keys computation are: the ephemeral KEM secret value, the ephemeral quantum secret value and *pss*. As long as at least one of these values remains uncompromised, then the session keys are indistinguishable from random values. Here, we essentially rely upon the IND-CPA security of the KEM scheme and the indistinguishability provided by the quantum key exchange (in addition to the PRF and swap-PRF security of $\mathsf{Extract}$ and Expand).

In case d), we consider first the possibility of an impersonation. This is precluded by the security of the signature scheme when the adversary must take part in the KEM exchange (case d)1)). If the peer's asymmetric long-term key is revealed, then we rely upon the fact that either the ephemeral KEM shared secret *ss* (case d)2)) or *pss* (cases d)3), d)4) and d)5)) remains uncompromised to forbid the adversary from deceiving the tested instance. Then, we rely essentially upon the INT-CTXT security of AE functions (the latter are keyed with a key computed from *pss* and *ss*). That is, in such a case, the tested instance does not accept because it rejects the last AE ciphertext it receives.

In case e), we consider the possibility of an impersonation but not to the tested instance π (the latter has an origin instance π') but from the adversary to π'. We show that, if the adversary alters the messages intended to π', then eventually π does not accept.

Let E_i be the event that the adversary wins the key indistinguishability experiment in Game i, and $\mathsf{adv}_i = \Pr[E_i] - 1/2$.

Game 0. This game corresponds to the key indistinguishability experiment described by Figure 5 in Section 3. Therefore

$$\Pr[E_0] = \mathsf{adv}_{\Pi}^{\mathsf{key\text{-}ind}}(\mathcal{A}) + \frac{1}{2} = \mathsf{adv}_0 + \frac{1}{2}$$

We split the proof according to the aforementioned subcases. Therefore, we have:

$$\begin{aligned}
\mathsf{adv}_0 &\leq \max\left(\mathsf{adv}_0^{c)}, \mathsf{adv}_0^{d)}, \mathsf{adv}_0^{e)}\right) \\
&\leq \mathsf{adv}_0^{c)} + \mathsf{adv}_0^{d)} + \mathsf{adv}_0^{e)} \\
&\leq \mathsf{adv}_0^{c)1)} + \mathsf{adv}_0^{c)2)} + \mathsf{adv}_0^{c)3)} + \mathsf{adv}_0^{c)4)} + \mathsf{adv}_0^{c)5)} \\
&\quad + \mathsf{adv}_0^{d)1)} + \mathsf{adv}_0^{d)2)} + \mathsf{adv}_0^{d)3)} + \mathsf{adv}_0^{d)4)} + \mathsf{adv}_0^{d)5)} \\
&\quad + \mathsf{adv}_0^{e)1)} + \mathsf{adv}_0^{e)2)} + \mathsf{adv}_0^{e)4)} + \mathsf{adv}_0^{e)5)} + \mathsf{adv}_0^{e)6)}
\end{aligned}$$

FreshKey c)1). We start with subcase c)1) in predicate FreshKey.

Game$^{c)1)}$ *1.* In this game, the challenger aborts the experiment if either an ephemeral KEM public key *epk* or ciphertext *ct* is not unique among all instances and stages. This ensures that a matching instance has a unique partner.

There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_0^{c)1)} \leq \mathsf{adv}_1^{c)1)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$

Game$^{c)1)}$ *2.* In this game, the challenger aborts the experiment if a KEM decapsulation failure occurs. That is, the challenger aborts the experiment if, on the one hand, the encapsulation done by an honest instance with input an ephemeral public key *epk* outputs a ciphertext *ct* and a shared secret *ss*, and, on the other hand, upon reception of *ct* by the honest partnered instance the decapsulation with the corresponding private key *esk* outputs a value $ss' \neq ss$. We have that

$$\mathsf{adv}_1^{c)1)} \leq \mathsf{adv}_2^{c)1)} + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}$$

Game$^{c)1)}$ *3.* In this game, the challenger aborts the experiment if a collision occurs in outputs of the chosen 2-universal hash function H_{ec}. These outputs are λ-bit long. Therefore, by definition of H_{ec} (cf. Definition 24), we have

$$\mathsf{adv}_2^{c)1)} \leq \mathsf{adv}_3^{c)1)} + \frac{n_P \cdot n_S \cdot n_T}{2^\lambda}$$

where the multiplicative factor $n_P \cdot n_S \cdot n_T$ comes from the fact that a new 2-universal hash function H_{ec} is chosen during each stage.

Game$^{c)1)}$ *4.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, and the matching instance π' and its parent, and it aborts if the guess is incorrect. Recall that, by Game$^{c)1)}$ 1, the partner instance π' is unique. Therefore we have:

$$\mathsf{adv}_3^{c)1)} \leq (n_P \cdot n_S)^2 \cdot n_T \cdot \mathsf{adv}_4^{c)1)}$$

Game$^{c)1)}$ *5.* In this game, we replace in stage t the shared secret ss computed by π with a value $\tilde{ss}$ of same size uniformly drawn at random. We do the same for π' in stage t. Recall that in stage t, π and π' are matching instances. Hence, they share the same KEM public key epk and ciphertext ct. Moreover, by Game$^{c)1)}$ 2, decapsulation failures are excluded. Consequently, π and π' share the same value ss in stage t. In addition, no query $\textsc{RevealECK}(\pi, t)$ nor $\textsc{RevealECK}(\pi', t)$ is issued. Therefore, detecting the change implies an adversary $\mathcal{B}_0$ able to break the IND-CPA security of the KEM scheme. Here we rely upon the IND-CPA notion because the two instances π and π' are matching in stage t (in particular, the ciphertext received by one instance is the value computed by the other). Therefore, we have

$$\mathsf{adv}_4^{c)1)} \leq \mathsf{adv}_5^{c)1)} + \mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}}$$

Game$^{c)1)}$ *6.* In this game, we replace in stage t (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot) = \mathsf{Extract}(\cdot, \tilde{ss})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$.

Since $\tilde{ss}$ is sampled uniformly at random by Game$^{c)1)}$ 5 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_1$ queries its $\textsc{Func}$ oracle in order to get k_1. However, to that point we have no guarantee that the instances π and π' share the same value pss at the beginning of stage t. If not, the respective inputs dk_0 to $\textsc{Func}$ may be different for π and π' (if no collision occurs in $\mathsf{Extract}$ and Expand).

Therefore, we have to count for at most 2 queries to $\textsc{Func}$ (i.e. $q_{\textsc{Fc}} \leq 2$).

Therefore

$$\mathsf{adv}_5^{c)1)} \leq \mathsf{adv}_6^{c)1)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2)$$

Game$^{c)1)}$ *7.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We apply the same change at π' if the latter shares the same value pss as π. We can do this, because k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{c)1)}$ 6).

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_2$ makes at most 7 calls to its $\textsc{Func}$ oracle to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1 respectively (i.e. $q_{\textsc{Fc}} \leq 7$). 7 calls to $\textsc{Func}$ are enough because the two instances π and π' match in stage t. Therefore, they use the same inputs to compute the latter four parameters (each instance can copy the parameter already computed by its partner instead of computing it anew).

Therefore

$$\mathsf{adv}_6^{c)1)} \leq \mathsf{adv}_7^{c)1)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)$$

Game$^{c)1)}$ *8.* In this game, we replace in stage t at the targeted instance π the function $\mathsf{Extract}(dk_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$. We apply the same change at π' if the latter shares the same value pss as π.

Since dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{c)1)}$ 7) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of $\mathsf{Extract}$.

During the stage, $\mathcal{B}_3$ makes at most 1 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 1$). Indeed, if the two instances π and π' share the same value pss, then they use the same input W because they share the ciphertext c'_R (since they are matching in stage t by assumption) and by Game$^{c)1)}$ 3, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{c)1)}$ 7, and the same transcript tr_2 since, again, they are matching).

Therefore

$$\mathsf{adv}_7^{c)1)} \leq \mathsf{adv}_8^{c)1)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

Game$^{c)1)}$ 9. In this game, we replace in stage t at the targeted instance π the function $\mathsf{Expand}(k_2,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$. We apply the same change at π' if the latter shares the same value pss as π.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{c)1)}$ 8) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_4$ to break the PRF security of Expand.

If the two instances share the same value pss, then they use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{c)1)}$ 8). Moreover, the two instances use then the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and by Game$^{c)1)}$ 3, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{c)1)}$ 7, and the same transcripts since, again, they are matching). Therefore, they use then the same function $\mathsf{Expand}(k_2,\cdot)$. This implies that, during the stage, $\mathcal{B}_4$ makes at most 3 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$).

Therefore

$$\mathsf{adv}_8^{c)1)} \leq \mathsf{adv}_9^{c)1)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, if the two partnered instances π and π' share the same value pss, then they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{c)1)}$ 7). They share also tr_0 and c_R (because the instances match). Therefore, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \texttt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \texttt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{c)1)}$ 3).

This implies that, if π and π' share pss, then they use the same random

function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and inputs to compute the session keys. In addition, RevealSK(π, t) is not queried, as otherwise the stage is unfresh. RevealSK(π', t) is not queried either. Consequently, the session keys are indistinguishable from values uniformly drawn at random (by Game$^{c)1)}$ 9), and can safely be replaced with the latter. This implies that the adversary has no better probability of guessing b_{key} then 1/2 and its advantage in Game$^{c)1)}$ 9 is $\mathsf{adv}_9^{c)1)} = 0$.

Collecting all the probabilities from Game$^{c)1)}$ 0 to Game$^{c)1)}$ 9 we have:

$$\begin{aligned}\mathsf{adv}_0^{c)1)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^\lambda}\right) + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\ & + (n_P \cdot n_S)^2 \cdot n_T \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) \\ & + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\end{aligned}$$

FreshKey c)2). Now, we proceed with case c)2) in predicate FreshKey.

The sequence of games corresponding to this case is very close to that of case c)1). The main difference lies on the fact that the session keys are "safe" because of an uncompromised QKD phase.

Game$^{c)2)}$ 1. In this game, the challenger aborts the experiment if either an ephemeral KEM public key *epk* or ciphertext *ct* is not unique among all instances and stages. This ensures that a matching instance has a unique partner.

There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_0^{c)2)} \leq \mathsf{adv}_1^{c)2)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$

Game$^{c)2)}$ 2. In this game, the challenger aborts the experiment if a KEM decapsulation failure occurs. That is, the challenger aborts the experiment if, on the one hand, the encapsulation done by an honest instance with input an ephemeral public key *epk* outputs a ciphertext *ct* and a shared secret *ss*, and, on the other hand, upon reception of *ct* by the honest partnered instance the decapsulation with the corresponding private key *esk* outputs a value $ss' \neq ss$. We have that

$$\mathsf{adv}_1^{c)2)} \leq \mathsf{adv}_2^{c)2)} + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}$$

Game$^{c)2)}$ 3. In this game, the challenger aborts the experiment if a collision occurs in outputs of the chosen 2-universal hash function H_{ec}. These outputs are λ-bit long. Therefore, by definition of H_{ec} (cf. Definition 24), we have

$$\mathsf{adv}_2^{c)2)} \leq \mathsf{adv}_3^{c)2)} + \frac{n_P \cdot n_S \cdot n_T}{2^\lambda}$$

where the multiplicative factor $n_P \cdot n_S \cdot n_T$ comes from the fact that a new 2-universal hash function H_{ec} is chosen during each stage.

Game$^{c)2)}$ *4.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, and the matching instance π' and its parent, and it aborts if the guess is incorrect. Recall that, by Game$^{c)2)}$ 1, the partner instance π' is unique. Therefore we have:

$$\mathsf{adv}_3^{c)2)} \leq (n_P \cdot n_S)^2 \cdot n_T \cdot \mathsf{adv}_4^{c)2)}$$

To that point, if the two partnered instances π and π' share the same value pss, then they share also the plaintext corresponding to the AE ciphertext c_R and c_I, since they are matching (hence agree on the ciphertexts and on the inputs used to compute the AE keys, including the KEM shared secret by Game$^{c)2)}$ 2). This implies in particular that π and π' share the (classical) data related to the QKD phase. The data sent by one instance is received unmodified by the other since the two instances are matching. Moreover, by assumption, we have that $\pi.\mathsf{eqk}[t] \neq \perp$ and $\pi'.\mathsf{eqk}[t] \neq \perp$ (i.e. $flag_{\mathsf{pe}} = \mathtt{true}$ in stage t). Therefore, the two instances do compute an actual ephemeral quantum secret. Consequently, we can apply Theorem 1 and safely replace $\pi.\mathsf{eqk}[t]$ and $\pi'.\mathsf{eqk}[t]$ with a value of same size uniformly drawn at random.

Game$^{c)2)}$ *5.* In this game, we replace the ephemeral quantum secret computed in stage t by the tested instance π with a value $\tilde{K}$ of same size uniformly drawn at random. We apply the same change with $\tilde{K}$ at π' if the latter shares the same value pss as π in stage t.

Tomamichel and Leverrier [109] prove the security of the BB84 protocol based on the following experiment. On the one hand (real context), the real BB84 protocol is executed. On the other hand (ideal context), a BB84 run is executed and the output key is replaced with a value of same length uniformly drawn at random. The maximum probability of distinguishing the context is $p_\Delta = (1 + \epsilon_\Delta)/2$ (see Section 4.7). By Game$^{c)2)}$ 3, collisions in H_{ec} are excluded. Therefore, we can remove the corresponding probability (correctness component) $\epsilon_{ec} = 2^{-\lambda}$ from ϵ_Δ and consider instead the secrecy component only $\epsilon_s = \epsilon_\Delta - \epsilon_{ec}$, as well as the probability $p_s = (1 + \epsilon_s)/2$.

Let F be the event that the adversary detects the change made in this game. We have

$$\begin{aligned}
\mathsf{adv}_4^{c)2)} &\leq \mathsf{adv}_5^{c)2)} + \Pr[\mathcal{A} \text{ wins} \mid \text{real QKD context}] \\
&\qquad - \Pr[\mathcal{A} \text{ wins} \mid \text{ideal QKD context}] \\
&\leq \mathsf{adv}_5^{c)2)} + \Pr[F] \\
&\leq \mathsf{adv}_5^{c)2)} + 2 \cdot p_s - 1 \\
&= \mathsf{adv}_5^{c)2)} + \epsilon_s
\end{aligned}$$

Remark 6. The classical data related to the QKD phase (in particular the r-bit syndrome Z and the λ-bit digest T) is encrypted. Therefore, an outside observer does not have access to these values. Consequently, we could remove from the quantity $\epsilon_{pa}(\nu)$ the contribution implied by the knowledge of Z and T. That is,

$\epsilon_{pa}(\nu)$ could be reduced by a factor $\sqrt{2^{r+\lambda}}$ and the overall security bound ϵ_Δ of the QKD phase decreased (see the proof of Proposition 11 in [109]). However, our security model allows the adversary to reveal in this game the parameters (internal state pss, KEM shared secret ss) used to compute the encryption key. Consequently, we keep $\epsilon_{pa}(\nu)$ (as well as ϵ_Δ and ϵ_s) as expressed in Section 4.7.

To that point, in stage t, π and π' use $\tilde{K}$ to key $\mathsf{Extract}^{\mathsf{swap}}$ in order to compute k_2.

Game$^{c)2)}$ *6.* In this game, we replace in stage t at the targeted instance π the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot) = \mathsf{Extract}(\cdot,\tilde{K})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$. We apply the same change at π' if it shares the same value pss as π in stage t.

Since $\tilde{K}$ is uniformly drawn at random by Game$^{c)2)}$ 5 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_0$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_0$ makes at most 1 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_5^{c)2)} \leq \mathsf{adv}_6^{c)2)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1)$$

Game$^{c)2)}$ *7.* In this game, we replace in stage t the function $\mathsf{Expand}(k_2,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$. We apply the same change at π' if it shares the same value pss as π in stage t.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$ (by Game$^{c)2)}$ 6) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_1$ makes at most 3 calls to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$).

Therefore

$$\mathsf{adv}_6^{c)2)} \leq \mathsf{adv}_7^{c)2)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, π and π' share tr_0 and c_R (because the instances match). They share also the same KEM shared secret (by Game$^{c)2)}$ 2). If they share the same value pss, then they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

By Game$^{c)2)}$ 7, π and π' share the same function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$. Therefore, they compute the same session keys. In addition, REVEALSK(π,t) is not queried, as otherwise the stage is unfresh. REVEALSK(π',t) is not queried either. Consequently, the session keys are indistinguishable from values uniformly drawn at random (by Game$^{c)2)}$ 7), and can safely be replaced with the latter. This implies that the adversary has no better probability of guessing b_{key} then $1/2$ and its advantage in Game$^{c)2)}$ 7 is $\mathsf{adv}_7^{c)2)} = 0$.

Collecting all the probabilities from Game$^{c)2)}$ 0 to Game$^{c)2)}$ 7 we have:

$$\mathsf{adv}_0^{c)2)} \leq n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^\lambda}\right) + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$
$$+ n_T \cdot (n_P \cdot n_S)^2 \cdot \left(\epsilon_s + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)\right)$$

FreshKey c)3). Now, we proceed with case c)3) in predicate FreshKey.

The sequence of games corresponding to this case is close to that of case b)1) in FreshAuth (cf. Definition 4). However, here we do not have to prove that pss is shared by the tested instance π and its matching instance. We will only consider what happens when pss is indeed shared by the matching instances.

More specifically, the way we treat stage t' here is the same as the way we treat stage t in case c)1) of FreshKey. The only difference is that we must guess the stage t' in addition to stage t (hence a factor $(n_P \cdot n_S \cdot n_T)^2$ instead of $(n_P \cdot n_S)^2 \cdot n_T$). Consequently, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake.

Game$^{c)3)}$ *1.* Collision of the ephemeral KEM public key epk and ciphertext ct values.

Game$^{c)3)}$ *2.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Game$^{c)3)}$ *3.* Collision resistance of H_{ec}.

Game$^{c)3)}$ *4.* Guess of tested instance π and its parent, matching instance π' and its parent, and stages t' and t.

Game$^{c)3)}$ *5.* IND-CPA security of the ephemeral KEM scheme.

Game$^{c)3)}$ *6.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with $\tilde{ss}$ (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)3)}$ *7.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

7 calls to the random function are enough because the two instances π and π' match in stage t. Therefore (if the two instances share pss at beginning of the stage), they use the same inputs to compute the latter four parameters (each instance can copy the parameter already computed by its partner instead of computing it anew).

Game$^{c)3)}$ *8.* PRF security of Extract keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)3)}$ *9.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Collecting all probabilities from Game$^{c)3)}$ 0 to Game$^{c)3)}$ 9, we have

$$\begin{aligned}\mathsf{adv}_0^{c)3)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\ + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \\ + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}_9^{c)3)}\Big)\end{aligned}$$

To that point, the two partnered instances π and π' share tr_0 and c_R (because the instances match). If they share the same value pss, then they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{c)3)}$ 7 and also by Game$^{c)3)}$ 2). In such a case, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \texttt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \texttt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{c)3)}$ 3).

This implies that π and π' (if they share pss) use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and inputs to compute the new value pss, hence they share the same value which is, in addition, indistinguishable from a value uniformly drawn at random (by Game$^{c)3)}$ 9).

Moreover, recall that neither RevealPSS(π, t') nor RevealPSS(π', t') is queried. Therefore, at the end of stage t', pss is indistinguishable from a value chosen uniformly at random and known only to π and π'.

Next, we will rely upon the PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ (keyed with pss). The next six games are repeated for stages $t'+1, \ldots, t$ (in order to make this appear, we denote each of these games as "Game$^{c)3)}$ $n+$"). Since the following six games rely upon the PRF security of Expand, $\mathsf{Extract}$ and the swap-PRF security of the latter, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake. We recall also that the instances π and π' are matching in stages u and neither RevealPSS(π, u) nor RevealPSS(π', u) is queried for stages $u \in \{t'+1, \ldots, t-1\}$. In addition, π and π' match also in stage t. The changes described below apply to π. They apply to π' as well if the latter shares the same value pss as π at the beginning of stage t' (we have shown that both instances share then the same value pss at the end of stage t').

Game$^{c)3)}$ *10+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)3)}$ *11+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{c)3)}$ *12+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)3)}$ *13+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{c)3)}$ *14+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)3)}$ *15+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{c)3)}$ 3 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \mathtt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{c)3)}$ 3.

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{9}^{c)3)} \leq\ & \mathsf{adv}_{15+}^{c)3)} \\ & + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\end{aligned}$$

To that point, at the end of stage t, the session keys are output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π' (if π and π' share pss at the beginning of stage t'). In addition, $\textsc{RevealSK}(\pi, t)$ is not queried, as otherwise the stage is unfresh. $\textsc{RevealSK}(\pi', t)$ is not queried either. Consequently, the session keys are indistinguishable from values uniformly drawn at random and can safely be replaced with the latter. This implies that the adversary has no better probability of guessing b_{key} then 1/2 and its advantage in Game$^{c)3)}$ 15+ is $\mathsf{adv}_{15+}^{c)3)} = 0$.

Collecting all the probabilities from Game$^{c)3)}$ 0 to Game$^{c)3)}$ 15+ we have:

$$\begin{aligned}\mathsf{adv}_{0}^{c)3)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + (2n_T - 1) \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) \\ & \quad + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2)\Big) \\ & \quad + n_T \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\Big)\end{aligned}$$

FreshKey c)4). Now, we proceed with case c)4) in predicate $\mathsf{FreshKey}$.

The way we treat stage t' is almost the same as the way we treat stage t in case c)2) of $\mathsf{FreshKey}$. The only differences is that we must guess the stage t' in addition to stage t (hence a factor $(n_P \cdot n_S \cdot n_T)^2$ instead of $(n_P \cdot n_S)^2 \cdot n_T$), and we rely in addition upon the collision resistance of H_{ec} (to ensure that the matching instances use the same value W). The way we treat the stages $t'+1$ up to stage t is the same as we do for the corresponding stages in case c)3) of $\mathsf{FreshKey}$. Consequently, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake.

$Game^{c)4)}$ *1.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

$Game^{c)4)}$ *2.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

$Game^{c)4)}$ *3.* Collision resistance of H_{ec}.

$Game^{c)4)}$ *4.* Guess of tested instance π and its parent, matching instance π' and its parent, and stages t' and t.

$Game^{c)4)}$ *5.* Indistinguishability of the ephemeral quantum secret (it is is now replaced with a value $\tilde{K}$ uniformly chosen at random). In addition, recall that neither $\textsc{RevealEQK}(\pi, t')$ nor $\textsc{RevealEQK}(\pi', t')$ is queried.

$Game^{c)4)}$ *6.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with $\tilde{K}$ (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

$Game^{c)4)}$ *7.* PRF security of Expand keyed with k_2 (*pss* is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Collecting the probabilities from $\mathrm{Game}^{c)4)}$ 0 to $\mathrm{Game}^{c)4)}$ 7, we have:

$$\begin{aligned}\mathsf{adv}_0^{c)4)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \epsilon_s\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}_7^{c)4)}\right)\end{aligned}$$

To that point, if the matching instances in stage t' π and π' share the same value *pss* at the beginning of stage t', then they share the same value *pss* at the end of stage t', which is indistinguishable from a value uniformly chosen at random. Moreover, neither $\textsc{RevealPSS}(\pi, t')$ nor $\textsc{RevealPSS}(\pi', t')$ is queried. Therefore, at the end of stage t', *pss* is indistinguishable from a value chosen uniformly at random and known only to π and π'.

Next, we will rely upon the PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ (keyed with pss). The next six games are repeated for stages $t'+1,\ldots,t$ (in order to make this appear, we denote each of these games as "Game$^{c)4)}$ $n+$"). Since the following six games rely upon the PRF security of Expand, $\mathsf{Extract}$ and the swap-PRF security of the latter, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake. We recall also that the instances π and π' are matching in stages u and neither $\textsc{RevealPSS}(\pi,u)$ nor $\textsc{RevealPSS}(\pi',u)$ is queried for stages $u \in \{t'+1,\ldots,t-1\}$. In addition, π and π' match also in stage t. The changes described below apply to π. They apply to π' as well if the latter shares the same value pss as π at the beginning of stage t' (we have shown that both instances share then the same value pss at the end of stage t').

Game$^{c)4)}$ 8+. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)4)}$ 9+. PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{c)4)}$ 10+. PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)4)}$ 11+. PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

7 calls to the random function are enough because the two instances π and π' match in stage t. Therefore (if the two instances shares pss at beginning of the stage), they use the same inputs to compute the latter four parameters (each instance can copy the parameter already computed by its partner instead of computing it anew).

Game$^{c)4)}$ 12+. PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)4)}$ 13+. PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{c)4)}$ 3 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \mathtt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{c)4)}$ 3.

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{7}^{c)4)} \leq\ & \mathsf{adv}_{13+}^{c)4)} \\ & + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \Big)\end{aligned}$$

To that point, at the end of stage t, the session keys are output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π' (if π and π' share pss at the beginning of stage t'). In addition, $\textsc{RevealSK}(\pi, t)$ is not queried, as otherwise the stage is unfresh. $\textsc{RevealSK}(\pi', t)$ is not queried either. Consequently, the session keys are indistinguishable from values uniformly drawn at random and can safely be replaced with the latter. This implies that the adversary has no better probability of guessing b_{key} then $1/2$ and its advantage in Game$^{c)4)}$ 13+ is $\mathsf{adv}^{c)4)}_{13+} = 0$.

Collecting all the probabilities from Game$^{c)4)}$ 0 to Game$^{c)4)}$ 13+ we have:

$$\begin{aligned}\mathsf{adv}^{c)4)}_0 \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \epsilon_s\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(n_T \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)\Big) \\ & \quad + (n_T - 1) \cdot \Big(2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\Big)\end{aligned}$$

FreshKey c)5). Now, we proceed with case c)5) in predicate FreshKey.

This case is close to case c)3) in FreshKey but, here, we rely upon the fact that pss is initialised with a (uniform) random value instead of being "fed" with an uncompromised ephemeral KEM secret value. Since we start from the very first stage, we do not have to guess a stage t' either (see case b)3) in FreshAuth).

The changes implied by each successive games are done at the tested instance π. They are also done at the matching instance π' if the latter shares the same initial pss value as π (i.e. $\pi.\mathsf{pss}[0] = \pi'.\mathsf{pss}[0]$).

Game$^{c)5)}$ *1.* Collision of the ephemeral KEM public key epk and ciphertext ct values.

Game$^{c)5)}$ *2.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Game$^{c)5)}$ *3.* Collision resistance of H_{ec}.

Game$^{c)5)}$ *4.* Guess of tested instance π and its parent, matching instance π' and its parent.

Now, we consider the very first stage (stage 1).

Game$^{c)5)}$ *5.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)5)}$ *6.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{c)5)}$ *7.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)5)}$ *8.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

7 calls to the random function are enough because the two instances π and π' match in stage t. Therefore (if the two instances shares pss at beginning of the stage), they use the same inputs to compute the latter four parameters (each instance can copy the parameter already computed by its partner instead of computing it anew).

Game$^{c)5)}$ *9.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)5)}$ *10.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Collecting all probabilities from Game$^{c)5)}$ 0 to Game$^{c)5)}$ 10, we have

$$\begin{aligned}\mathsf{adv}_0^{c)5)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\ + (n_P \cdot n_S)^2 \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) + \mathsf{adv}^{c)5)}_{10}\Big)\end{aligned}$$

To that point, at the end of stage 1, the two partnered instances π and π' share tr_0 and c_R (because the instances match in stage 1). If they share the same value pss at the beginning of stage 1, then they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{c)5)}$ 8 and also by Game$^{c)5)}$ 2). In such a case, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \texttt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \texttt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{c)5)}$ 3).

This implies that π and π' (if they share pss at beginning of stage 1) use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ (by Game$^{c)5)}$ 10) known only to them, and the same inputs to compute the new value pss. Hence they share this value. Moreover, neither RevealPSS$(\pi, 0)$ nor RevealPSS$(\pi', 0)$ is queried,

and RevealPSS$(\pi, 1)$ and RevealPSS$(\pi', 1)$ are not queried either. Therefore, at the end of stage 1, pss is indistinguishable from a value chosen uniformly at random and known only to π and π'.

Now, we guess the stage t.

Game$^{c)5)}$ *11.* In this game, the challenger tries to guess the stage t and aborts if it does not succeed. Therefore

$$\mathsf{adv}_{10}^{c)5)} \leq n_T \cdot \mathsf{adv}_{11}^{c)5)}$$

In the subsequent games, we use the swap-PRF security of Extract and the PRF security of Extract and Expand to eventually argue that the value pss at the beginning of stage t is random and known only to π and π'. Since the reasoning relies upon security properties already mentioned, once again we only indicate the properties at stake.

The next six games are repeated for stages u, $1 < u < t$, and stage t (in order to make this appear, we denote each of these games as "Game$^{c)5)}$ n+"). The corresponding changes are done at π in stages u and t. They are done also at its matching instance π' in stages u and t if π' shares the same pss value as π at the beginning of stage 1 (recall that π and π' match in stages u, $1 < u < t$, and t).

If π and π' share the same pss value at the beginning of stage 1, then, in stages u, $1 < u < t$, and t, π and π' share the inputs to the different random functions (replacing Extract and Expand) because they match, compute the same intermediary keys (hence use the same random functions), and there is no collision in $\mathsf{H_{ec}}$ outputs (by Game$^{c)5)}$ 3). Moreover, recall that RevealPSS(π, u) and RevealPSS(π', u) are not queried where π and π' match in stages u, $1 < u < t$.

Game$^{c)5)}$ *12+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)5)}$ *13+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{c)5)}$ *14+.* PRF security of Extract keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)5)}$ *15+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{c)5)}$ *16+.* PRF security of Extract keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{c)5)}$ *17+.* PRF security of Expand keyed with k_2 (*pss* is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{c)5)}$ 3 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \texttt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{c)5)}$ 3.

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{11}^{c)5)} \leq\ & \mathsf{adv}_{17+}^{c)5)} \\ & + (n_T - 1) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + 2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) \\ & \qquad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)\Big)\end{aligned}$$

To that point, at the end of stage t, the session keys are output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π' (if π and π' share *pss* at the beginning of stage 1). In addition, $\textsc{RevealSK}(\pi, t)$ is not queried, as otherwise the stage is unfresh. $\textsc{RevealSK}(\pi', t)$ is not queried either. Consequently, the session keys are indistinguishable from values uniformly drawn at random and can safely be replaced with the latter. This implies that the adversary has no better probability of guessing b_{key} then 1/2 and its advantage in Game$^{c)5)}$ 17+ is $\mathsf{adv}_{17+}^{c)5)} = 0$.

Collecting all the probabilities from Game$^{c)5)}$ 0 to Game$^{c)5)}$ 17+ we have:

$$\begin{aligned}\mathsf{adv}_{0}^{c)5)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\ & + (n_P \cdot n_S)^2 \cdot (n_T(n_T-1)+1) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + 2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) \\ & + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)\Big)\end{aligned}$$

FreshKey d)1). Now, we proceed with case d)1) in predicate $\mathsf{FreshKey}$.

Here, no honest instance shares in stage t the same cid as the tested instance π. This implies in particular that the adversary must be active during the ephemeral KEM exchange (consequently, it knows the KEM shared secret in stage t) and must also compute a valid signature (as otherwise the tested instance does not accept). We show in the games below that the security of the signature scheme forbids eventually the adversary from winning the experiment.

Game$^{d)1)}$ *1.* In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}_{0}^{d)1)} \leq \mathsf{adv}_{1}^{d)1)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}}$$

$Game^{d)1)}$ *2.* In this game, the challenger aborts the experiment if either an ephemeral KEM public key epk or ciphertext ct is not unique among all instances and stages. There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_1^{d)1)} \leq \mathsf{adv}_2^{d)1)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$

$Game^{d)1)}$ *3.* In this game, the challenger tries to guess the targeted instance π, its parent and the targeted stage t, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_2^{d)1)} \leq n_P \cdot n_S \cdot n_T \cdot \mathsf{adv}_3^{d)1)}$$

$Game^{d)1)}$ *4.* In this game, the challenger tries to guess which public key (i.e. which corresponding party) the signature received by π is verified with, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_3^{d)1)} \leq n_P \cdot \mathsf{adv}_4^{d)1)}$$

$Game^{d)1)}$ *5.* In this game, the challenger aborts the experiment if the adversary $\mathcal{A}$ produces a valid signature under an uncorrupted key for a message which was not output by an honest instance. This reduces straightforwardly to the security of the signature scheme SIG. We show how to build an adversary $\mathcal{B}_1$ against the EUF-CMA security of the signature scheme. This adversary is given a signing oracle. It behaves as follows. $\mathcal{B}_1$ sets $lpk_w = pk^*$ for some party w, with pk^* from the signature security experiment, and generates all other protocol parameters as specified. It uses its signing oracle in order to output honest signatures valid under lpk_w. If $\mathcal{A}$ submits a valid signature under lpk_w that was not output by an honest instance, then $\mathcal{B}_1$ uses these message and signature to break the EUF-CMA security of the signature scheme. Since this message was not signed before with the corresponding private key (because $\mathcal{B}_1$ did not query its signing oracle with this message to get the signature), then it is a valid forgery (i.e. $\mathcal{B}_1$ wins).

During a given stage, $\mathcal{B}_1$ makes at most 1 call to SIGN when it needs to output a signature for an honest instance (corresponding to party w) (i.e. $q_{\mathrm{SG}} \leq 1$).

$\mathcal{B}_1$ can answer any REVEALLCK query from $\mathcal{A}$ (since $\mathcal{B}_1$ manages the simulation) but for the private key corresponding to $lpk_w = pk^*$. However, in the context where $\mathcal{A}$ tries to compute a valid signature (on behalf of an honest instance owned by some party v) in order to make an honest instance π accept, the long term (signature) key of the putative partner party v cannot be revealed before the honest targeted instance π accepts, as otherwise $\mathsf{FreshAuth}$ becomes false. Therefore, if a query REVEALLCK(w) is issued before the targeted party accepts, then the simulation is stopped.

Taking into account that there are at most n_P parties, we have

$$\mathsf{adv}_4^{d)1)} \leq \mathsf{adv}_5^{d)1)} + \mathsf{adv}_{\mathsf{SIG}}^{\mathsf{euf\text{-}cma}}(1)$$

To that point, the tested instance π accepts upon reception of a valid signature (σ_I or σ_R). By Game$^{d)1)}$ 2, the ephemeral KEM public key epk and ciphertext ct are unique. By assumption of case d)1), no honest instance shares the same cid as π in stage t. Moreover, no collision occurs in H outputs (by Game$^{d)1)}$ 1). Therefore, no honest instance signs the same transcript (tr_1 or tr_2) as that of computed by π. If the adversary reveals a long-term key to compute a valid signature, then $\mathsf{FreshKey}$ is false and a query to $\textsc{Test}$ returns $\perp$. If the adversary produces a valid signature (which protects in particular a transcript (tr_1 or tr_2) that no honest instance has signed), then the challenger aborts the experiment and π does not accept (hence $\mathsf{FreshKey}$ is false). If the challenger sends an invalid signature, then π rejects it and does not accept either. Therefore, a query $\textsc{Test}(\pi, t)$ returns $\perp$. Hence, in Game$^{d)1)}$ 5 the advantage of the adversary is $\mathsf{adv}_5^{d)1)} = 0$.

Collecting all the probabilities from Game$^{d)1)}$ 0 to Game$^{d)1)}$ 5 we have:

$$\mathsf{adv}_0^{d)1)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} \right) + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + n_P^2 \cdot n_S \cdot n_T \cdot \mathsf{adv}_{\mathsf{SIG}}^{\mathsf{euf\text{-}cma}} \quad (1)$$

FreshKey d)2). Now, we proceed with case d)2) in predicate $\mathsf{FreshKey}$.

Since we are in case d)2), the tested instance π has no origin instance in stage t but shares its contributive identifier with some honest instance π'' (i.e. $\pi.\mathsf{cid}[t] = \pi''.\mathsf{cid}[t]$). If π is a responder's instance, then this implies that m_3 must not come from π'' (as otherwise π'' is an origin instance for π). If π is an initiator's instance, then this implies that either the AE ciphertext c_R or the MAC tag τ_R in m_2, or $m_4 = c'_R$ must not come from π''. The last games described below incorporate these different possibilities and show how the INT-CTXT security of the AE scheme forbids eventually the adversary from winning the experiment.

Game$^{d)2)}$ 1. In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}_0^{d)2)} \leq \mathsf{adv}_1^{d)2)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}}$$

Game$^{d)2)}$ 2. In this game, the challenger aborts the experiment if either an ephemeral KEM public key epk or ciphertext ct is not unique among all instances and stages. There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_1^{d)2)} \leq \mathsf{adv}_2^{d)2)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} \right)$$

Game$^{d)2)}$ 3. In this game, the challenger aborts the experiment if a KEM decapsulation failure occurs. That is, the challenger aborts the experiment if, on the one hand, the encapsulation done by an honest instance with input an ephemeral

public key epk outputs a ciphertext ct and a shared secret ss, and, on the other hand, upon reception of ct by the honest partnered instance the decapsulation with the corresponding private key esk outputs a value $ss' \neq ss$. We have that

$$\mathsf{adv}_2^{d)2)} \leq \mathsf{adv}_3^{d)2)} + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}$$

Game$^{d)2)}$ *4.* In this game, the challenger aborts the experiment if a collision occurs in $\mathsf{Extract}$ outputs among all instances and stages. Therefore

$$\mathsf{adv}_3^{d)2)} \leq \mathsf{adv}_4^{d)2)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}}$$

Game$^{d)2)}$ *5.* In this game, the challenger aborts the experiment if a collision occurs in Expand outputs among all instances and stages. Therefore

$$\mathsf{adv}_4^{d)2)} \leq \mathsf{adv}_5^{d)2)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}}$$

Game$^{d)2)}$ *6.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, and the putative peer party and instance π'' where no query RevealECK(π, t) or RevealECK(π'', t) is issued, and it aborts if the guess is incorrect. By Game$^{d)2)}$ 2, there can be only one such instance π''. Therefore we have:

$$\mathsf{adv}_5^{d)2)} \leq (n_P \cdot n_S)^2 \cdot n_T \cdot \mathsf{adv}_6^{d)2)}$$

Game$^{d)2)}$ *7.* In this game, we replace in stage t the shared secret ss computed by π with a value $\tilde{ss}$ of same size uniformly drawn at random. We do the same for π'' in stage t. Recall that in stage t, π and π'' share the same cid. Hence, they share the same KEM public key epk and ciphertext ct. Moreover, by Game$^{d)2)}$ 3, decapsulation failures are excluded. Consequently, π and π'' share the same value ss in stage t. In addition, no query RevealECK(π, t) nor RevealECK(π'', t) is issued. Therefore, detecting the change implies an adversary $\mathcal{B}_1$ able to break the IND-CPA security of the KEM scheme. Here we rely upon the IND-CPA notion because the two instances π and π'' share the same cid in stage t (i.e. the ciphertext received by one instance is the value computed by the other). Therefore, we have

$$\mathsf{adv}_6^{d)2)} \leq \mathsf{adv}_7^{d)2)} + \mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}}$$

Game$^{d)2)}$ *8.* In this game, we replace in stage t at the tested instance π the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot) = \mathsf{Extract}(\cdot, \tilde{ss})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot)}$. We do a similar change at π'' (which shares the same cid as π).

Since $\tilde{ss}$ is sampled uniformly at random by Game$^{d)2)}$ 7 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_2$ queries its Func oracle in order to get k_1. However, to that point we have no guarantee that the instances π and π'' share the same value pss at the beginning of stage t. If not, the respective inputs dk_0 to Func

are different for π and π'' (by Game$^{d)2)}$ 4 and Game$^{d)2)}$ 5).

Therefore, we have to count for at most 2 queries to FUNC (i.e. $q_{\text{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_7^{d)2)} \leq \mathsf{adv}_8^{d)2)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2)$$

Game$^{d)2)}$ 9. As explained in Game$^{d)2)}$ 8, to that point, we cannot ensure that the two instances π and π'' compute the same value dk_0, input to compute k_1. It may happen that using distinct values dk_0 as input to $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$, the outputs be equal. We want to preclude such an event. The probability that such a collision occurs at π is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_1. Therefore, in this game, the challenger aborts the experiment if such a collision occurs.

Hence,

$$\mathsf{adv}_8^{d)2)} \leq \mathsf{adv}_9^{d)2)} + \frac{1}{2^{\ell_k}}$$

Game$^{d)2)}$ 10. In this game, the challenger aborts the experiment if the key k_1 computed by π is equal to k_0 computed by π or to k_0 computed by π'', and used to key Expand. Indeed, in subsequent games we will rely upon the CR property of the function Expand (keyed with k_0) but also on the PRF security of Expand (keyed with k_1). Therefore, we must be able to distinguish when Expand is keyed with distinct keys. Then, we will be able to safely replace Expand with a random function when keyed with a specific (unique) key, and still rely upon the CR property of Expand when keyed with another specific (unique) key.

By Game$^{d)2)}$ 8, the ℓ_k-bit long key k_1 computed by π is output by a random function. Therefore, we have that

$$\mathsf{adv}_9^{d)2)} \leq \mathsf{adv}_{10}^{d)2)} + \frac{2}{2^{\ell_k}}$$

Game$^{d)2)}$ 11. In this game, we replace (at the tested instance π) the function $\mathsf{Expand}(k_1,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this, because k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{d)2)}$ 8). We apply a similar change at π'' if the latter shares the same value pss as π.

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand.

During the stage, $\mathcal{B}_3$ queries its oracle FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1. Since π and π'' share cid, one query to FUNC is enough to get hk_R, n_R and dk_1 for π and π'' (if the latter uses the same value pss as π). However, we have no guarantee that π'' would use the same inputs as π to compute the other parameters. Therefore, 2×4 calls to FUNC may be necessary (if π'' uses the same value pss as π). Consequently, the oracle FUNC is queried at most $3+2\times 4$ times (i.e. $q_{\text{FC}} \leq 11$).

Therefore

$$\mathsf{adv}_{10}^{d)2)} \leq \mathsf{adv}_{11}^{d)2)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(11)$$

Game$^{d)2)}$ *12.* By Game$^{d)2)}$ 9, if π and π'' use distinct inputs dk_0, then they compute distinct values k_1. We want to exclude the event that, while π (as initiator's instance) and π'' use distinct parameters k_1, they compute the same AE key hk_R needed to compute and decrypt the AE ciphertext c_R received by π. By Game$^{d)2)}$ 11, the probability that π computes this same AE key as π'' in such a case is $1/2^{\ell_{hk}}$, where ℓ_{hk} is the bitlength of hk_R.

Therefore, in this game the challenger aborts the experiment if such a collision occurs.

Hence

$$\mathsf{adv}_{11}^{d)2)} \leq \mathsf{adv}_{12}^{d)2)} + \frac{1}{2^{\ell_{hk}}}$$

Game$^{d)2)}$ *13.* In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it.

By Game$^{d)2)}$ 11, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_4$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_4$ makes at most 1 call to Dec in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value *pss* as π, then both instances use the same random function to compute the AE key. If this instance uses the same inputs to the random function as π, then the two instances compute the same AE key. In such a case, $\mathcal{B}_4$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$). Note that this message must come from the adversary (and not from an honest instance) as otherwise π has an origin instance (which could then be π'' since the latter shares cid with π). However, even if π does not receive the honest message, then $\mathcal{B}_4$ may have to compute such a message for an honest "sending" instance (which would then be "replaced" by the adversary with a message of its choice).

Consequently, we have

$$\mathsf{adv}_{12}^{d)2)} \leq \mathsf{adv}_{13}^{d)2)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Game$^{d)2)}$ *14.* In this game, the challenger aborts the experiment if the instance π, when it is an initiator's instance, receives a valid second AE ciphertext (c'_R) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it.

By Game$^{d)2)}$ 11, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk'_R) used to decrypt the ciphertext (c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_5$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_5$ makes at most 1 call to DEC in order to decrypt c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss as π, then both instances use the same random function to compute the AE key. If this instance uses the same inputs to the random function as π, then the two instances compute the same AE key. In such a case, $\mathcal{B}_5$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$). Note that this message must come from the adversary (and not from an honest instance) as otherwise π may have an origin instance (which could then be π'' since the latter shares cid with π). However, even if π does not receive the honest message, then $\mathcal{B}_5$ may have to compute such a message for an honest "sending" instance (which would then be "replaced" by the adversary with a message of its choice).

Consequently, we have

$$\mathsf{adv}_{13}^{d)2)} \leq \mathsf{adv}_{14}^{d)2)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

To that point, in order for π to accept in stage t, it must receive a valid last AE ciphertext.

Let us consider first that π is a responder's instance. A responder's instance (during a complete run) receives only one AE ciphertext (c_I). The ciphertext received by π cannot come from the honest instance π'' (which shares the same cid as π in stage t) as, otherwise, π'' becomes an origin instance of π in stage t. π decrypts the AE ciphertext with tr_1 as AE header. The latter is computed from epk and ct which are unique (by Game$^{d)2)}$ 2). Therefore, tr_1 is also unique (by Game$^{d)2)}$ 1). This means that no honest instance (besides π'') can compute such an AE ciphertext (and the received AE ciphertext does not come from π''). Consequently, if the received AE ciphertext is not valid, then π does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment (by Game$^{d)2)}$ 13).

To sum up, (if the challenger does not abort the experiment) π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$.

Let us consider now that π is an initiator's instance. The adversary must compute either m_2 or m_4.

- Let us assume that the adversary computes m_2. It must compute either the AE ciphertext or the MAC tag (since ct is shared by π and π'').
  - Let us assume that the first AE ciphertext (c_R) received by π does not come from the honest instance π''. π decrypts c_R with tr_0 as AE header. The latter is computed from epk and ct which are unique (by Game$^{d)2)}$ 2). Therefore, tr_0 is also unique (by Game$^{d)2)}$ 1). This means that no honest instance (besides π'') can compute such an AE ciphertext (and the received AE ciphertext does not come from π''). Consequently, if the received AE ciphertext is not valid, then π does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment (by Game$^{d)2)}$ 13).

- Let us assume that the MAC tag τ_R does not come from π'' (and c_R is received unmodified).
  - Let us assume that π and π'' share pss at beginning of stage t. π and π'' share also cid in stage t (i.e. epk and ct). Therefore, the two instances compute the same MAC key and tag τ_R. If the adversary sends $\tilde{\tau_R} \neq \tau_R$, then the received MAC tag is invalid (because we assume that $\mathsf{MAC.Tag}$ is deterministic) and π aborts the run.
  - Let us assume that π and π'' do not share pss at beginning of stage t. Then, they compute distinct values k_0 (by Game$^{d)2)}$ 4), dk_0 (by Game$^{d)2)}$ 5), k_1 (by Game$^{d)2)}$ 9) and hk_R (by Game$^{d)2)}$ 12). This means that π decrypts c_R with a key distinct from that of used by π'' to compute the AE ciphertext. Moreover, π and π'' use tr_0 as AE header. The latter is computed from epk and ct which are unique (by Game$^{d)2)}$ 2). Consequently, the AE header tr_0 is also unique (by Game$^{d)2)}$ 1). This means that no honest instance (besides π'') ever uses tr_0 as AE header to compute an AE ciphertext. If c_R is invalid, then π aborts the run (hence, it does not accept). If c_R is deemed valid, then we can leverage the INT-CTXT security of the AE function to claim that it is a forgery. Hence, the challenger aborts the experiment (by Game$^{d)2)}$ 13).

– Let us assume that the adversary computes m_4. π decrypts c'_R with tr_2 as AE header. The latter is computed from epk and ct. We can apply a similar reasoning as previously and conclude that either π rejects c'_R because it is invalid (and π aborts the run) or c'_R is valid and is a forgery. Indeed, the only honest instance that could use the same AE header (tr_2) to compute an AE ciphertext is π'' and the latter does not compute the AE ciphertext received by π. Consequently, the challenger aborts the experiment (by Game$^{d)2)}$ 14).

To sum up, (if the challenger does not abort the experiment) π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$.

Hence, the advantage of the adversary in Game$^{d)2)}$ 14 is $\mathsf{adv}_{14}^{d)2)} = 0$.

Collecting all the probabilities from Game$^{d)2)}$ 0 to Game$^{d)2)}$ 14 we have:

$$\begin{aligned}\mathsf{adv}_0^{d)2)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\ & + (n_P \cdot n_S)^2 \cdot n_T \cdot \left(\frac{3}{2^{\ell_k}} + \frac{1}{2^{\ell_{hk}}}\right) \\ & + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} \\ & + (n_P \cdot n_S)^2 \cdot n_T \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(11) \\ & \qquad + 3 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1)\Big)\end{aligned}$$

FreshKey d)3). Now, we proceed with case d)3) in predicate $\mathsf{FreshKey}$.

In this case, we consider an instance π'' that matches π in stages u, $t' \leq u < t$. We show that the pss value computed by π (and π'') is indistinguishable from a

value uniformly drawn at random thanks to an uncompromised ephemeral KEM exchange done in stage t'. Then, we show that the AE key computed in stage t from the resulting *pss* value allows relying upon the INT-CTXT security of the AE function used to compute the AE ciphertexts received by the tested instance. We conclude that the tested instance π rejects the ciphertext and never accepts. We follow a strategy similar to that of case b)1) in FreshAuth.

The changes described in the subsequent games apply to the tested instance π. They apply also the matching instance π'' in stages u, $t' \leq u < t$, if π'' shares the same value *pss* as π at the beginning of stage t'.

$Game^{d)3)}$ *1.* Collision resistance of hash function H.

$Game^{d)3)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

$Game^{d)3)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

$Game^{d)3)}$ *4.* Collision resistance of Extract.

$Game^{d)3)}$ *5.* Collision resistance of Expand.

$Game^{d)3)}$ *6.* In this game, the challenger aborts the experiment if the key k_0 used by any instance to key Expand in stage $t' \geq 1$ is equal to the key k_1 or k_2 used to key Expand in stage t', or is equal to the key k_0, k_1 or k_2 used to key Expand in stages u, $t' < u < t$, or is equal to the key k_0 or k_1 used to key Expand in stage t.

Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (when keyed with a pseudo-random key). That is, in the latter case, we will replace Expand with a random function. Therefore, we must be able to distinguish the keys that key Expand when we want to leverage its CR property and the keys that key Expand which will eventually be replaced with a random function.

Let us first consider one instance. If the keys k_1, k_2 in stage t', and k_0, k_1, k_2 in stages u, $t' < u < t$, and k_0 and k_1 in stage t are chosen uniformly at random, then the probability that one of these keys collide with the key k_0 used in stage t' is $1/2^{\ell_k}$. Therefore, the probability that none of these keys (in stages t', and u, $t' < u < t$, and t) collide with k_0 used in stage t' is

$$\left(1 - \frac{1}{2^{\ell_k}}\right)^2 \times \left(1 - \frac{1}{2^{\ell_k}}\right)^{3(n_T-2)} \times \left(1 - \frac{1}{2^{\ell_k}}\right)^2 = \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}$$

because each instance executes at most n_T stages. Let $A_{\tilde{\pi}}$ be such an event for a given instance $\tilde{\pi}$.

Let us now consider the event that $A_{\tilde{\pi}}$ holds for all instances $\tilde{\pi}$: $\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}$. Let

us consider the opposite event that, for at least one instance $\tilde{\pi}$, the key k_0 in stage t' be equal to at least another key k_1 or k_2 in stage t', or k_0, k_1 or k_2 in a stage u, $t' < u < t$, or k_0 or k_1 in stage t. The corresponding event is $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}$. The challenger aborts the experiment if the latter event holds.

We have $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}} = \bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}$. Therefore, the corresponding probability is

$$\begin{aligned}
\Pr\left[\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}\right] &= \Pr\left[\bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}\right] \\
&\leq \sum_{\tilde{\pi}} \Pr\left[\overline{A_{\tilde{\pi}}}\right] \\
&= \sum_{\tilde{\pi}} (1 - \Pr[A_{\tilde{\pi}}]) \\
&= \sum_{\tilde{\pi}} \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \\
&= n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right)
\end{aligned}$$

because there are at most $n_P \cdot n_S$ instances.

If $\frac{1}{2^{\ell_k}}$ is close to 0, then $1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2} \simeq \frac{3 \cdot n_T - 2}{2^{\ell_k}}$. Hence

$$n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \simeq \frac{n_P \cdot n_S \cdot (3 \cdot n_T - 2)}{2^{\ell_k}}.$$

Therefore, we have that

$$\mathsf{adv}_5^{d)3)} \leq \mathsf{adv}_6^{d)3)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right)$$

To that point, we have:

$$\begin{aligned}
\mathsf{adv}_0^{d)3)} \leq\ & \mathsf{adv}_6^{d)3)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} \\
& + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}
\end{aligned}$$

Game$^{d)3)}$ 7. In this game, the challenger aborts the experiment if the parameter *pss* used by any instance in stage t' as input to the function Extract is equal to at least one of the subsequent parameters *pss* used in input to Extract in stages

u, $t' < u \leq t$.

This rule follows the same rationale as that of introduced in Game$^{d)3)}$ 6. Indeed, in subsequent games we will rely upon the CR property of the function Extract but also on the PRF security of Extract (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Extract is keyed with distinct keys. Then, we will be able to safely replace Extract with a random function when keyed with a specific (unique) key, and still rely upon the CR property of Extract when keyed with another specific (unique) key.

Let us consider first one instance. If the parameters pss used in stages u, $t' < u \leq t$, are chosen uniformly at random, then the probability that one of theses values collide with pss used in stage t' is $1/2^{\ell_{pss}}$. Therefore, the probability that none of these values be equal to pss used in stage t' is $\left(1 - 2^{-\ell_{pss}}\right)^{n_T - 1}$, because an instance executes at most n_T stages. Let $A_{\tilde{\pi}}$ be such an event for an instance $\tilde{\pi}$.

Let us now consider the event that $A_{\tilde{\pi}}$ holds for all instances $\tilde{\pi}$: $\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}$. Let us consider the opposite event that, for at least one instance $\tilde{\pi}$, the parameter pss used in stage t' be equal to at least a parameter pss used in a stage u, $t' < u \leq t$. The corresponding event is $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}$. The challenger aborts the experiment if the latter event holds.

We have $\bigcap_{\tilde{\pi}} A_{\tilde{\pi}} = \bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}$. Therefore, the corresponding probability is

$$\begin{aligned}
\Pr\left[\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}\right] &= \Pr\left[\bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}\right] \\
&\leq \sum_{\tilde{\pi}} \Pr\left[\overline{A_{\tilde{\pi}}}\right] \\
&= \sum_{\tilde{\pi}} \left(1 - \Pr[A_{\tilde{\pi}}]\right) \\
&= \sum_{\tilde{\pi}} \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \\
&= n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right)
\end{aligned}$$

because there are at most $n_P \cdot n_S$ instances.

If $\frac{1}{2^{\ell_{pss}}}$ is close to 0, then $1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1} \simeq \frac{n_T - 1}{2^{\ell_{pss}}}$. Hence

$$n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \simeq \frac{n_P \cdot n_S \cdot (n_T - 1)}{2^{\ell_{pss}}}.$$

Therefore, we have that

$$\mathsf{adv}_6^{d)3)} \leq \mathsf{adv}_7^{d)3)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right)$$

Game$^{d)3)}$ *8.* In this game, the challenger aborts the experiment if a collision occurs in outputs of the chosen 2-universal hash function H_{ec}. These outputs are λ-bit long. Therefore, by definition of H_{ec} (cf. Definition 24), we have

$$\mathsf{adv}_7^{d)3)} \leq \mathsf{adv}_8^{d)3)} + \frac{n_P \cdot n_S \cdot n_T}{2^{\lambda}}$$

where the multiplicative factor $n_P \cdot n_S \cdot n_T$ comes from the fact that a new 2-universal hash function H_{ec} is chosen during each stage.

Game$^{d)3)}$ *9.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, the matching instance π'' and its parent, and the stage t' where no query RevealECK(π, t') or RevealECK(π'', t') is issued, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_8^{d)3)} \leq \left(n_P \cdot n_S \cdot n_T\right)^2 \cdot \mathsf{adv}_9^{d)3)}$$

Game$^{d)3)}$ *10.* In this game, we replace in stage t' the shared secret ss computed by π with a value $\tilde{ss}$ of same size uniformly drawn at random. We do the same for π'' in stage t'. Recall that in stage t', π and π'' are matching instances. Hence, they share the same KEM public key epk and ciphertext ct. Moreover, by Game$^{d)3)}$ 3, decapsulation failures are excluded. Consequently, π and π'' share the same value ss in stage t'. In addition, no query RevealECK(π, t') nor RevealECK(π'', t') is issued. Therefore, detecting the change implies an adversary $\mathcal{B}_0$ able to break the IND-CPA security of the KEM scheme. Here we rely upon the IND-CPA notion because the two instances π and π'' are matching in stage t' (in particular, the ciphertext received by one instance is the value computed by the other). Therefore, we have

$$\mathsf{adv}_9^{d)3)} \leq \mathsf{adv}_{10}^{d)3)} + \mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}}$$

Game$^{d)3)}$ *11.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π'') the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot) = \mathsf{Extract}(\cdot, \tilde{ss})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot)}$.

Since $\tilde{ss}$ is sampled uniformly at random by Game$^{d)3)}$ 10 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the swap-PRF security of Extract.

During a given stage, $\mathcal{B}_1$ queries its Func oracle in order to get k_1. However, to that point we have no guarantee that the instances π and π'' share the same value pss at the beginning of stage t'. If not, the respective inputs dk_0 to Func are different for π and π'' (because no collision occurs in Extract and Expand by

Games$^{d)3)}$ 4 and 5).

Therefore, we have to count for at most 2 queries to FUNC (i.e. $q_{\text{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{10}^{d)3)} \leq \mathsf{adv}_{11}^{d)3)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2)$$

Game$^{d)3)}$ 12. As explained in Game$^{d)3)}$ 11, to that point, we cannot ensure that the two instances compute the same value dk_0, input to compute k_1. It may happen that using distinct values dk_0 as input to $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$, the outputs be equal. We want to preclude such an event. The probability that such a collision occurs at π is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_1. Therefore, in this game, the challenger aborts the experiment if such a collision occurs.

Hence

$$\mathsf{adv}_{11}^{d)3)} \leq \mathsf{adv}_{12}^{d)3)} + \frac{1}{2^{\ell_k}}$$

Game$^{d)3)}$ 13. In this game, the challenger aborts the experiment if the key k_0 computed by π'' is equal to the key k_1 computed by π, when the two instances do not use the same value pss at beginning of stage t'. Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Expand is keyed with distinct keys. Then, we will be able to safely replace Expand with a random function when keyed with a specific (unique) key, and still rely upon the CR property of Expand when keyed with another specific (unique) key.

By Game$^{d)3)}$ 11, the ℓ_k-bit long key k_1 computed respectively by π is output by a random function. Therefore, we have that

$$\mathsf{adv}_{12}^{d)3)} \leq \mathsf{adv}_{13}^{d)3)} + \frac{1}{2^{\ell_k}}$$

Game$^{d)3)}$ 14. In this game, we replace in stage t' (at the targeted instance π) the function $\mathsf{Expand}(k_1,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We apply the same change at π'' if the latter shares the same value pss as π. We can do this, because k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{d)3)}$ 11).

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_2$ makes at most 7 calls to FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1 respectively (i.e. $q_{\text{FC}} \leq 7$).

Therefore

$$\mathsf{adv}_{13}^{d)3)} \leq \mathsf{adv}_{14}^{d)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)$$

Game$^{d)3)}$ 15. By Game$^{d)3)}$ 12, if π and π'' use distinct inputs dk_0, then they compute distinct values k_1. We want to exclude the event that, while π and π'' use distinct keys k_1, they compute the same AE keys (hk_I or hk_R) needed to compute the first AE ciphertext received by π (c_I or c_R). By Game$^{d)3)}$ 14, the probability that π computes this same AE key as π'' in such a case is $1/2^{\ell_{hk}}$,

where ℓ_{hk} is the bitlength of hk_I and hk_R. Therefore, in this game the challenger aborts the experiment if such a collision occurs.

Hence

$$\mathsf{adv}_{14}^{d)3)} \leq \mathsf{adv}_{15}^{d)3)} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{d)3)}$ *16.* In this game, the challenger aborts the experiment if the instance π receives, in stage t', a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it.

By Game$^{d)3)}$ 14, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_3$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_3$ makes at most 1 call to DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value *pss* as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_3$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{15}^{d)3)} \leq \mathsf{adv}_{16}^{d)3)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Let us assume that, in stage t', π and π'' use distinct values *pss*. Then, they compute compute distinct values dk_0 (because, by Games$^{d)3)}$ 4 and 5, no collision occurs in $\mathsf{Extract}$ and Expand outputs during the computation of k_0 and dk_0). In turn, this implies that the two instances output distinct values k_1 and distinct AE keys to compute and decrypt the first AE ciphertext received by π (by Game$^{d)3)}$ 15). If the first AE ciphertext received by π is invalid, then it is rejected by π (hence, it does not appear in the list $\mathsf{m_r}$ of the messages received by π).

π decrypts the received AE ciphertext (c_R or c_I depending on whether π is initiator or responder) with tr_0 or tr_1 as AE header. These values are computed from *epk* and *ct* which are unique (by Game$^{d)3)}$ 2). Therefore, tr_0 and tr_1 are also unique (by Game$^{d)3)}$ 1). π and π'' match in stage t'. Therefore, π'' is the only honest instance able to compute an AE ciphertext with such AE headers. In addition, π'' does not use the same AE key to compute the AE ciphertext as π to decrypt it. Consequently, if π deems the received AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment by Game$^{d)3)}$ 16.

Since π and π'' match in stage t' (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that the AE ciphertext is not a forgery, hence both instances do share the same value *pss* at the beginning of stage t'.

Game$^{d)3)}$ *17.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π'') the function $\mathsf{Extract}(dk_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$. We can safely apply the change at both instances because (by the preceding games) we a have excluded the possibility for the two instances to use distinct values dk_1.

Since dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{d)3)}$ 14) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_4$ to break the PRF security of $\mathsf{Extract}$.

During the stage, $\mathcal{B}_4$ makes at most 1 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 1$). Indeed, the two instances π and π'' use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and by Game$^{d)3)}$ 8, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{d)3)}$ 14 and 16, and the same transcript tr_2 since, again, they are matching).

Therefore

$$\mathsf{adv}_{16}^{d)3)} \leq \mathsf{adv}_{17}^{d)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

Game$^{d)3)}$ *18.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π'') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{d)3)}$ 17) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of Expand.

We can safely apply this change because the two instances π and π'' do compute the same value k_2. Indeed, the two instances use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{d)3)}$ 17). Moreover, the two instances use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and use the same AE parameters hk'_R and n'_R to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{d)3)}$ 14 and 16, and the same transcripts since, again, they are matching).

This implies also that, during the stage, $\mathcal{B}_5$ makes at most 3 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$). Therefore

$$\mathsf{adv}_{17}^{d)3)} \leq \mathsf{adv}_{18}^{d)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, the two partnered instances π and π'' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{d)3)}$ 14). They share also tr_0 and c_R (because the instances match). Therefore, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π'' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \mathtt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \mathtt{true}$, then the initiator's instance and the responder's instance use

$W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{d)3)}$ 8).

This implies that π and π'' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and inputs to compute the new value pss, hence they share the same value which is, in addition, indistinguishable from a value uniformly drawn at random (by Game$^{d)3)}$ 18).

Moreover, recall that neither RevealPSS(π, t') nor RevealPSS(π'', t') is queried. Therefore, at the end of stage t', pss is indistinguishable from a value chosen uniformly at random and known only to π and π''.

Next, we will rely upon the PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ (keyed with pss). The next six games are repeated for stages $t'+1, \ldots, t-1$ (in order to make this appear, we denote each of these games as "Game$^{d)3)}$ $n+$"). Since the following six games rely upon the PRF security of Expand, Extract and the swap-PRF security of the latter, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake. We recall also that the instances π and π'' are matching in stages u (hence, the changes described below apply to π'' as well as π) and neither RevealPSS(π, u) nor RevealPSS(π'', u) is queried for stages $u \in \{t'+1, \ldots, t-1\}$.

Game$^{d)3)}$ 19+. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{d)3)}$ 20+. PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{d)3)}$ 21+. PRF security of Extract keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{d)3)}$ 22+. PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{d)3)}$ 23+. PRF security of Extract keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{d)3)}$ 24+. PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{d)3)}$ 8 to ensure that the matching instances π and π'' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \texttt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{d)3)}$ 8.

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π'' (i.e. $\pi.\mathsf{pss}[t-1] = \pi''.\mathsf{pss}[t-1]$). Taking

into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{18}^{d)3)} \leq\ & \mathsf{adv}_{24+}^{d)3)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + 2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) \\ & \qquad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7) \Big)\end{aligned}$$

Now, we consider stage t. We proceed with the next four games in order to show that the AE keys used in stage t to encrypt (and decrypt) the AE ciphertexts are indistinguishable from random values. Since the reasoning relies upon the PRF security of the functions already mentioned, once again we only indicate the security properties at stake. We recall that at the beginning of stage t the instances π and π'' share the same value pss.

Game$^{d)3)}$ *25.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss. We replace $\mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ at π and π''. k_0 is now output by $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ (and $q_{\mathrm{FC}} \leq 1$).

Game$^{d)3)}$ *26.* PRF security of Expand keyed with k_0. We replace $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ at π and π''. dk_0 is now output by $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$).

Game$^{d)3)}$ *27.* PRF security of $\mathsf{Extract}$ keyed with dk_0. We replace $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ at π and π''. k_1 is now output by $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$ because the simulation may have to query its FUNC oracle twice if π and π'' do not use the same input ss).

Game$^{d)3)}$ *28.* PRF security of Expand keyed with k_1. We replace $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ at π and π''. This change is applied at π'' if π'' shares the same cid as π in stage t. In that case, one query to $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ is enough to get hk_R, n_R and dk_1 for π and π''. However, we cannot ensure that π'' uses the same inputs as π to compute hk_I, n_I, hk'_R and n'_R. Therefore, it may happen that the random function be queried twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Now, we have also to take into account the possibility that π and π'' do not use the same inputs to the random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (cf. Game$^{d)3)}$ 27) and yet output the same value k_1. If π and π'' do not share, in stage t, the same parameters epk, ct, then FUNC may have to be queried once per instance to compute hk_R and n_R. That is, $q_{\mathrm{FC}} \leq 1 + 2 \times 6$.

hk_I and hk'_R are now output by $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

We will use $\mathsf{cid} = epk \| ct$ (and its uniqueness, by Game$^{d)3)}$ 2) only as a "binder" between π and its partner instance in stage t (the adversary might well issue a query REVEALECK(π, t) or REVEALECK(π'', t)).

For Game$^{d)3)}$ 25 up to Game$^{d)3)}$ 28, we have:

$$\mathsf{adv}_{24+}^{d)3)} \leq \mathsf{adv}_{28}^{d)3)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(13)$$

Game$^{d)3)}$ *29.* In this game, the challenger aborts the experiment if π (as initiator's instance) receives a valid first AE ciphertext (c_R in m_2) not computed by an honest instance.

By Game$^{d)3)}$ 28, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (for π and any honest instance that shares the same cid and pss as π). Therefore, the AE key (hk_R) used to decrypt the ciphertext (c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_6$ to break the INT-CTXT security of the AE function used to decrypt the ciphertext.

During the stage, $\mathcal{B}_6$ makes at most 1 call to Dec in order to decrypt c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same cid (and pss) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_6$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{28}^{d)3)} \leq \mathsf{adv}_{29}^{d)3)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Game$^{d)3)}$ *30.* In this game, the challenger aborts the experiment if π receives a valid final message (m_3 or m_4 depending on whether π is a responder's or an initiator's instance) not computed by an honest instance.

By Game$^{d)3)}$ 28, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (for π and any honest instance that shares the same cid and pss as π). Therefore, the AE key (hk_I or hk'_R) used to decrypt the ciphertext (c_I or c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_7$ to break the INT-CTXT security of the AE function used to decrypt the ciphertext.

During the stage, $\mathcal{B}_7$ makes at most 1 call to Dec in order to decrypt c_I or c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same cid (and pss) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_7$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{29}^{d)3)} \leq \mathsf{adv}_{30}^{d)3)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Let us consider first that π is a responder's instance. The adversary must compute either m_1 or m_3 in stage t, as otherwise π'' is an origin instance of π.

- Let us assume that the adversary computes m_1. The KEM ciphertext ct computed by π is unique (by Game$^{d)3)}$ 2). Therefore, no honest instance (but, possibly, π'') uses ct to compute tr_1. This implies that tr_1 is also unique (by Game$^{d)3)}$ 1). In addition, since $m_1 = epk$ is computed by the adversary, π'' does not compute its own value tr_1 from the KEM public key sent by the adversary. Therefore, no honest instance uses the value tr_1 computed by π as AE header to compute an AE ciphertext. Consequently, if the AE ciphertext received by π is invalid, then π aborts the run and does not accept. If the

AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. In the latter case, the challenger aborts the experiment (by $\text{Game}^{d)3)}$ 30).

– Let us assume that the adversary computes m_3 (and does not modify m_1). π decrypts the AE ciphertext with tr_1 as AE header. The latter is computed from epk and ct which are unique (by $\text{Game}^{d)3)}$ 2). Therefore, tr_1 is also unique (by $\text{Game}^{d)3)}$ 1). This means that no honest instance (besides, possibly, π'') can compute such an AE ciphertext (and the received AE ciphertext does not come from π''). Consequently, if the received AE ciphertext is not valid, then π does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment (by $\text{Game}^{d)3)}$ 30).

To sum up, (if the challenger does not abort the experiment) π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$.

Let us consider now that π is an initiator's instance. The adversary must compute either m_2 or m_4 in stage t.

– Let us assume that the adversary computes m_2. It must compute either the KEM ciphertext or the AE ciphertext or the MAC tag.
  - Let us assume that the KEM ciphertext is computed by the adversary. π decrypts c_R with tr_0 as AE header. The latter is computed from epk (computed by π) and the received KEM ciphertext. epk is unique (by $\text{Game}^{d)3)}$ 2). Therefore, tr_0 is also unique (by $\text{Game}^{d)3)}$ 1). Since the adversary computes the KEM ciphertext, π'' does not use the same value ct to compute its own parameter tr_0. Therefore, no honest instance ever uses the value tr_0 computed by π as AE header to compute an AE ciphertext. Consequently, if the AE ciphertext received by π is invalid, then π aborts the run and does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that it is a forgery. Then, the challenger aborts the run (by $\text{Game}^{d)3)}$ 29).
  - Let us assume that the first AE ciphertext (c_R) received by π does not come from the honest instance π'' (but ct does). π decrypts c_R with tr_0 as AE header. The latter is computed from epk and ct which are unique (by $\text{Game}^{d)3)}$ 2). Therefore, tr_0 is also unique (by $\text{Game}^{d)3)}$ 1). This means that no honest instance (besides π'') can compute such an AE ciphertext (and the received AE ciphertext does not come from π''). Consequently, if the received AE ciphertext is not valid, then π does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment (by $\text{Game}^{d)3)}$ 29).
  - Let us assume that the MAC tag τ_R does not come from π'' (and ct and c_R are received unmodified).
    * Let us assume that the KEM public key received by π'' is not the KEM public key computed by π. The KEM public key and ciphertext are unique (by $\text{Game}^{d)3)}$ 2). Therefore, tr_0 computed by π (from its

own KEM public key and the received KEM ciphertext, computed by π'') is also unique (by Game$^{d)3)}$ 1). This means that no honest instance uses the value tr_0 computed by π as AE header to compute an AE ciphertext. If the received AE ciphertext (c_R is invalid, then π aborts the run and does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that it is a forgery. Then, the challenger aborts the experiment (by Game$^{d)3)}$ 29).

* Let us assume that the KEM public key received by π'' is computed by π. π and π'' share the same value pss at beginning of stage t because they share the same value pss at beginning of stage t' (by Game$^{d)3)}$ 16). π and π'' share also cid in stage t (i.e. epk and ct). Therefore, the two instances compute the same MAC key and tag τ_R. If the adversary sends $\tilde{\tau_R} \neq \tau_R$, then the received MAC tag is invalid (because we assume that $\mathsf{MAC.Tag}$ is deterministic) and π aborts the run.

– Let us assume that the adversary computes m_4. π decrypts c'_R with tr_2 as AE header. The latter is computed from epk and ct. If epk and ct are shared by π and π'', then no other honest instance but π'' computes the same value tr_2 as π (by Game$^{d)3)}$ 2 and Game$^{d)3)}$ 1). If one of these values is not shared by π and π'', then no honest instance ever computes the same value tr_2 as π. This implies that no honest instance (but, possibly, π'') ever uses tr_2 as AE header to compute an AE ciphertext (but the AE ciphertext received by π is not computed by π''. We can apply a similar reasoning as previously and conclude that either π rejects c'_R because it is invalid (and π aborts the run) or c'_R is valid and is a forgery. Indeed, the only honest instance that could use the same AE header (tr_2) to compute an AE ciphertext is π'' and the latter does not compute the AE ciphertext received by π. Consequently, the challenger aborts the experiment (by Game$^{d)3)}$ 30).

To sum up, (if the challenger does not abort the experiment) π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$.

Hence, the advantage of the adversary in Game$^{d)3)}$ 30 is $\mathsf{adv}_{30}^{d)3)} = 0$.

Collecting all the probabilities from Game$^{d)3)}$ 0 to Game$^{d)3)}$ 30 we have:

$$\begin{aligned}
\mathsf{adv}_0^{d)3)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^\lambda}\right) \\
& + n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \frac{1}{2^{\ell_{hk}-1}} + \frac{1}{2^{\ell_k - 1}}\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) \\
& \quad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13) + 5 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) + (2n_T - 3) \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) \\
& \quad + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) \\
& \quad\quad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\Big)
\end{aligned}$$

FreshKey d)4). Now, we proceed with case d)4) in predicate FreshKey.

The reasoning here is close to that of case d)3) in FreshKey. The main difference is that we rely upon the security provided by an uncompromised QKD phase rather than an uncompromised ephemeral KEM exchange.

More precisely, in this case, we consider an instance π'' that matches π in stages u, $t' \leq u < t$. We show that the *pss* value computed by π (and π'') is indistinguishable from a value uniformly drawn at random thanks to an uncompromised QKD phase done in stage t'. Then, we show that the AE key computed in stage t from the resulting *pss* value allows relying upon the INT-CTXT security of the AE function used to compute the last AE ciphertext received by the tested instance. We conclude that the tested instance π rejects the ciphertext and never accepts. We follow essentially a strategy similar to that of subcase b)2) in FreshAuth.

The changes described in the subsequent games apply to the tested instance π. They apply also the matching instance π'' in stages u, $t' \leq u < t$, if π'' shares the same value *pss* as π at the beginning of stage t'.

Game$^{d)4)}$ *1.* Collision resistance of hash function H.

Game$^{d)4)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

Game$^{d)4)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Here, we do not rely upon the (IND-CPA or IND-CCA) security of the KEM scheme. However, its correctness is required as two instances sharing the same cid must compute the same KEM shared secret *ss* (the latter is involved in the computation of the intermediary secret values and keys).

$Game^{d)4)}$ *4.* Collision resistance of $\mathsf{Extract}$.

$Game^{d)4)}$ *5.* Collision resistance of Expand.

$Game^{d)4)}$ *6.* Collision probability for the keys k_0, k_1, k_2 when pseudo-random. The challenger aborts the experiment if a parameter k_0 or k_1 used in stage t' is equal to one of the following parameters: k_2 in stage t', or k_0, k_1, k_2 in a stage u, $t' < u < t$, or k_0, k_1 in stage t. Following a reasoning similar to that of the subcase $\mathsf{FreshAuth}$ b)1), we have:

$$\mathsf{adv}_0^{d)4)} \leq \mathsf{adv}_6^{d)4)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)}\right)$$

$Game^{d)4)}$ *7.* Collision probability for the values pss when pseudo-random.

$Game^{d)4)}$ *8.* Collision resistance of H_{ec}.

To that point, we have:

$$\begin{aligned}\mathsf{adv}_0^{d)4)} \leq\ & \mathsf{adv}_8^{d)4)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} \\ & + n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T-1}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right)\end{aligned}$$

$Game^{d)4)}$ *9.* In this game, the challenger aborts the experiment if an honest instance receives an AE ciphertext valid under a key distinct from the key used by the sender to compute the ciphertext. This reduces straightforwardly to the key-committing security of the AE scheme. Therefore

$$\mathsf{adv}_8^{d)4)} \leq \mathsf{adv}_9^{d)4)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{key\text{-}cmt}}$$

$Game^{d)4)}$ *10.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, the putative party and partner instance π'', and the stage t' where no query $\textsc{RevealEQK}(\pi, t')$ nor $\textsc{RevealEQK}(\pi'', t')$, where $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi''.\mathsf{eqk}[t'] \neq \perp$, is issued, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_9^{d)4)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \mathsf{adv}_{10}^{d)4)}$$

To that point, in stage t', if π and π'' use distinct values pss, then they compute distinct values dk_0 (because no collision occurs in $\mathsf{Extract}$ and Expand outputs during the computation of k_0 and dk_0, by $\text{Games}^{d)4)}$ 4 and 5. In turn, this implies that the two instances output distinct values k_1 and distinct AE keys to compute and decrypt the first AE ciphertext received by π. Therefore,

either the first AE ciphertext received by π is invalid and rejected by π (hence, it does not appear in the list $\mathsf{m_r}$ of the messages received by π), or the ciphertext is valid but the challenger aborts the experiment by Game$^{d)4)}$ 9. Since π and π'' match in stage t' (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that both instances do share the same value pss at the beginning of stage t'.

Moreover, since π and π'' match in stage t', this implies that epk and ct are shared by π'' and π. By unicity of epk and ct, π'' is the unique origin instance of π. This implies in addition, that the plaintext data corresponding to c_R and c_I are also shared by the two instances (i.e. the (classical) data related to the quantum key agreement are exchanged through an authenticated channel). Moreover, by assumption, we have that $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi''.\mathsf{eqk}[t'] \neq \perp$ (i.e. $flag_{\mathsf{pe}} = \mathtt{true}$ in stage t'). Therefore, the two instances do compute an actual ephemeral quantum secret. Consequently, we can apply Theorem 1 and safely replace $\pi.\mathsf{eqk}[t']$ and $\pi''.\mathsf{eqk}[t']$ with a value of same size uniformly drawn at random.

Game$^{d)4)}$ 11. In this game, we replace the ephemeral quantum secret computed in stage t' by the targeted instance π and its partner instance π'' with a value $\tilde{K}$ of same size uniformly drawn at random.

Tomamichel and Leverrier [109] prove the security of the BB84 protocol based on the following experiment. On the one hand (real context), the real BB84 protocol is executed. On the other hand (ideal context), a BB84 run is executed and the output key is replaced with a value of same length uniformly drawn at random. The maximum probability of distinguishing the context is $p_\Delta = (1 + \epsilon_\Delta)/2$ (see Section 4.7). By Game$^{d)4)}$ 8, collisions in $\mathsf{H_{ec}}$ are excluded. Therefore, we can remove the corresponding probability (correctness component) $\epsilon_{ec} = 2^{-\lambda}$ from ϵ_Δ and consider instead the secrecy component only $\epsilon_s = \epsilon_\Delta - \epsilon_{ec}$, as well as the probability $p_s = (1 + \epsilon_s)/2$.

Let F be the event that the adversary detects the change made in this game. We have

$$\begin{aligned}
\mathsf{adv}_{10}^{d)4)} &\leq \mathsf{adv}_{11}^{d)4)} + \Pr[\mathcal{A} \text{ wins} \mid \text{real QKD context}] \\
&\qquad - \Pr[\mathcal{A} \text{ wins} \mid \text{ideal QKD context}] \\
&\leq \mathsf{adv}_{11}^{d)4)} + \Pr[F] \\
&\leq \mathsf{adv}_{11}^{d)4)} + 2 \cdot p_s - 1 \\
&= \mathsf{adv}_{11}^{d)4)} + \epsilon_s
\end{aligned}$$

To that point, in stage t', π and π'' use $\tilde{K}$ to key $\mathsf{Extract}^{\mathsf{swap}}$ in order to compute k_2.

Game$^{d)4)}$ 12. In this game, we replace in stage t' (at the targeted instance π and its matching instance π'') the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{K}, \cdot) = \mathsf{Extract}(\cdot, \tilde{K})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$.

Since $\tilde{K}$ is uniformly drawn at random by Game$^{d)4)}$ 11 the probability for $\mathcal{A}$

to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_0$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_0$ makes at most 1 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{11}^{d)4)} \leq \mathsf{adv}_{12}^{d)4)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1)$$

Game$^{d)4)}$ *13.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π'') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$ (by Game$^{d)4)}$ 12) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_1$ makes at most 3 calls to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$).

Therefore

$$\mathsf{adv}_{12}^{d)4)} \leq \mathsf{adv}_{13}^{d)4)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, at the end of stage t', pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π''. Moreover, recall that neither REVEALPSS(π, t') nor REVEALPSS(π'', t') is queried.

The last games of this case are exactly the same as the last games of case b)1) (i.e. Game$^{b)1)}$ 19+ to Game$^{b)1)}$ 30). They correspond to the changes applied in stages $u \in \{t'+1, \ldots, t-1\}$ (to the instances π and π'' matching in stages u) and finally in stage t.

Game$^{d)4)}$ *14+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{d)4)}$ *15+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{d)4)}$ *16+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{d)4)}$ *17+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{d)4)}$ *18+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{d)4)}$ *19+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{d)4)}$ 8 to ensure that the matching instances π and π'' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \mathtt{false}$ then they both use $W = 0$, otherwise one

instance uses K_I and the other K_R which must be equal by Game$^{d)4)}$ 8.

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π'' (i.e. $\pi.\mathsf{pss}[t-1] = \pi''.\mathsf{pss}[t-1]$).

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{13}^{d)4)} \leq\ & \mathsf{adv}_{19+}^{d)4)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \Big)\end{aligned}$$

Next, we consider stage t.

Game$^{d)4)}$ 20. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss. We replace $\mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ at π and π''. k_0 is now output by $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ (and $q_{\mathrm{FC}} \leq 1$).

Game$^{d)4)}$ 21. PRF security of Expand keyed with k_0. We replace $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ at π and π''. dk_0 is now output by $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$).

Game$^{d)4)}$ 22. PRF security of $\mathsf{Extract}$ keyed with dk_0. We replace $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ at π and π''. k_1 is now output by $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$ because the simulation may have to query its FUNC oracle twice if π and π'' do not use the same input ss).

Game$^{d)4)}$ 23. PRF security of Expand keyed with k_1. We replace $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ at π and π''. This change is applied at π'' if π'' shares the same cid as π in stage t. In that case, one query to $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ is enough to get hk_R, n_R and dk_1 for π and π''. However, we cannot ensure that π'' uses the same inputs as π to compute hk_I, n_I, hk'_R, n'_R. Therefore, it may happen that the random function be queried twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Now, we have also to take into account the possibility that π and π'' do not use the same inputs to the random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (cf. Game$^{d)4)}$ 22) and yet output the same value k_1. If π and π'' do not share, in stage t, the same parameters epk, ct, then FUNC may have to be queried once per instance to compute hk_R and n_R. That is, $q_{\mathrm{FC}} \leq 1 + 2 \times 6$.

hk_I and hk'_R are now output by $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

We will use $\mathsf{cid} = epk \| ct$ (and its uniqueness, by Game$^{d)4)}$ 2) only as a "binder" between π and its partner instance in stage t (the adversary might well issue a query REVEALECK(π, t) or REVEALECK(π'', t)).

Game$^{d)4)}$ *24.* In this game, the challenger aborts the experiment if π (as initiator's instance) receives a valid first AE ciphertext (c_R in m_2) not computed by an honest instance.

By Game$^{d)4)}$ 23, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (for π and any honest instance that shares the same cid and pss as π). Therefore, the AE key (hk_R) used to decrypt the ciphertext (c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_2$ to break the INT-CTXT security of the AE function used to decrypt the ciphertext.

During the stage, $\mathcal{B}_2$ makes at most 1 call to Dec in order to decrypt c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same cid (and pss) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_2$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{23}^{d)4)} \leq \mathsf{adv}_{24}^{d)4)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Game$^{d)4)}$ *25.* In this game, the challenger aborts the experiment if π receives a valid final message (m_3 or m_4 depending on whether π is a responder's or an initiator's instance) not computed by an honest instance.

By Game$^{d)4)}$ 23, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (for π and any honest instance that shares the same cid and pss as π). Therefore, the AE key (hk_I or hk'_R) used to decrypt the ciphertext (c_I or c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_3$ to break the INT-CTXT security of the AE function used to decrypt the ciphertext.

During the stage, $\mathcal{B}_3$ makes at most 1 call to Dec in order to decrypt c_I or c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same cid (and pss) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_3$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{24}^{d)4)} \leq \mathsf{adv}_{25}^{d)4)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

To that point, we have

$$\begin{aligned}\mathsf{adv}_{19+}^{d)4)} \leq \mathsf{adv}_{25}^{d)4)} &+ \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(13) \\ &+ 3 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)\end{aligned}$$

Let us consider first that π is a responder's instance. The adversary must compute either m_1 or m_3 in stage t, as otherwise π'' is an origin instance of π.

– Let us assume that the adversary computes m_1. The KEM ciphertext ct computed by π is unique (by Game$^{d)4)}$ 2). Therefore, no honest instance (but, possibly, π'') uses ct to compute tr_1. This implies that tr_1 is also unique

(by Game$^{d)4)}$ 1). In addition, since $m_1 = epk$ is computed by the adversary, π'' does not compute its own value tr_1 from the KEM public key sent by the adversary. Therefore, no honest instance uses the value tr_1 computed by π as AE header to compute an AE ciphertext. Consequently, if the AE ciphertext received by π is invalid, then π aborts the run and does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. In the latter case, the challenger aborts the experiment (by Game$^{d)4)}$ 25).

– Let us assume that the adversary computes m_3 (and does not modify m_1). π decrypts the AE ciphertext with tr_1 as AE header. The latter is computed from epk and ct which are unique (by Game$^{d)4)}$ 2). Therefore, tr_1 is also unique (by Game$^{d)4)}$ 1). This means that no honest instance (besides, possibly, π'') can compute such an AE ciphertext (and the received AE ciphertext does not come from π''). Consequently, if the received AE ciphertext is not valid, then π does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment (by Game$^{d)4)}$ 25).

To sum up, (if the challenger does not abort the experiment) π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$.

Let us consider now that π is an initiator's instance. The adversary must compute either m_2 or m_4 in stage t.

– Let us assume that the adversary computes m_2. It must compute either the KEM ciphertext or the AE ciphertext or the MAC tag.
  - Let us assume that the KEM ciphertext is computed by the adversary. π decrypts c_R with tr_0 as AE header. The latter is computed from epk (computed by π) and the received KEM ciphertext. epk is unique (by Game$^{d)4)}$ 2). Therefore, tr_0 is also unique (by Game$^{d)4)}$ 1). Since the adversary computes the KEM ciphertext, π'' does not use the same value ct to compute its own parameter tr_0. Therefore, no honest instance ever uses the value tr_0 computed by π as AE header to compute an AE ciphertext. Consequently, if the AE ciphertext received by π is invalid, then π aborts the run and does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that it is a forgery. Then, the challenger aborts the run (by Game$^{d)4)}$ 24).
  - Let us assume that the first AE ciphertext (c_R) received by π does not come from the honest instance π'' (but ct does). π decrypts c_R with tr_0 as AE header. The latter is computed from epk and ct which are unique (by Game$^{d)4)}$ 2). Therefore, tr_0 is also unique (by Game$^{d)4)}$ 1). This means that no honest instance (besides π'') can compute such an AE ciphertext (and the received AE ciphertext does not come from π''). Consequently, if the received AE ciphertext is not valid, then π does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment (by Game$^{d)4)}$ 24).

- Let us assume that the MAC tag τ_R does not come from π'' (and ct and c_R are received unmodified).

  * Let us assume that the KEM public key received by π'' is not the KEM public key computed by π. The KEM public key and ciphertext are unique (by Game$^{d)4)}$ 2). Therefore, tr_0 computed by π (from its own KEM public key and the received KEM ciphertext, computed by π'') is also unique (by Game$^{d)4)}$ 1). This means that no honest instance uses the value tr_0 computed by π as AE header to compute an AE ciphertext. If the received AE ciphertext (c_R is invalid, then π aborts the run and does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that it is a forgery. Then, the challenger aborts the experiment (by Game$^{d)4)}$ 24).
  * Let us assume that the KEM public key received by π'' is computed by π. π and π'' share the same value pss at beginning of stage t because they share the same value pss at beginning of stage t' (by Game$^{d)4)}$ 9). π and π'' share also cid in stage t (i.e. epk and ct). Therefore, the two instances compute the same MAC key and tag τ_R. If the adversary sends $\tilde{\tau_R} \neq \tau_R$, then the received MAC tag is invalid (because we assume that $\mathsf{MAC.Tag}$ is deterministic) and π aborts the run.

– Let us assume that the adversary computes m_4. π decrypts c'_R with tr_2 as AE header. The latter is computed from epk and ct. If epk and ct are shared by π and π'', then no other honest instance but π'' computes the same value tr_2 as π (by Game$^{d)4)}$ 2 and Game$^{d)4)}$ 1). If one of these values is not shared by π and π'', then no honest instance ever computes the same value tr_2 as π. This implies that no honest instance (but, possibly, π'') ever uses tr_2 as AE header to compute an AE ciphertext (but the AE ciphertext received by π is not computed by π''. We can apply a similar reasoning as previously and conclude that either π rejects c'_R because it is invalid (and π aborts the run) or c'_R is valid and is a forgery. Indeed, the only honest instance that could use the same AE header (tr_2) to compute an AE ciphertext is π'' and the latter does not compute the AE ciphertext received by π. Consequently, the challenger aborts the experiment (by Game$^{d)4)}$ 25).

To sum up, (if the challenger does not abort the experiment) π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$.

Hence, the advantage of the adversary in Game$^{d)4)}$ 25 is $\mathsf{adv}_{25}^{d)4)} = 0$.

Collecting all the probabilities from Game$^{d)4)}$ 0 to Game$^{d)4)}$ 25, we have:

$$\begin{aligned}
\mathsf{adv}_0^{d)4)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^\lambda}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \epsilon_s\right) \\
& + n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T-1}\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} + \mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13) + 3 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) \\
& \quad + n_T \cdot \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + (n_T - 2) \cdot \Big(2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big) \\
& \quad + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)\Big)\Big)
\end{aligned}$$

FreshKey d)5). Now, we proceed with case d)5) in predicate FreshKey.

The reasoning here is close to that of cases d)3) and d)4) in FreshKey. The main difference is that we rely upon the security provided by the fact that the initial value of *pss* is uniformly chosen at random.

More precisely, in this case, we consider an instance π'' that matches π in stages u, $0 < u < t$. We show that the *pss* value computed by π (and π'') is indistinguishable from a value uniformly drawn at random if $\pi.\mathsf{pss}[0]$ (and $\pi''.\mathsf{pss}[0]$) is uniformly drawn at random. Then, we show that the AE key computed in stage t from the resulting *pss* value allows relying upon the INT-CTXT security of the AE function used to compute the last AE ciphertext received by the tested instance. We conclude that the tested instance π rejects the ciphertext and never accepts. The strategy is essentially similar to that of subcase b)3) in FreshAuth.

The changes described in the subsequent games apply to the tested instance π. They apply also to the matching instance π'' in stages u, $0 < u < t$, if π'' shares the same initial value *pss* as π.

Game$^{d)5)}$ *1.* Collision resistance of hash function H.

Game$^{d)5)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

Game$^{d)5)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Here, we do not rely upon the (IND-CPA or IND-CCA) security of the KEM scheme. However, its correctness is required as two instances sharing the same $\mathsf{cid} = epk \| ct$ must compute the same KEM shared secret *ss* (the latter is involved in the computation of the intermediary secret values and keys).

Game$^{d)5)}$ *4.* Collision resistance of H_{ec}.

To that point, we have

$$\mathsf{adv}_0^{d)5)} \leq \mathsf{adv}_4^{d)5)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right)$$

Game$^{d)5)}$ *5.* In this game, the challenger tries to guess the targeted instance π, its parent, the putative partner party and instance π'', and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_4^{d)5)} \leq (n_P \cdot n_S)^2 \cdot \mathsf{adv}_5^{d)5)}$$

Game$^{d)5)}$ *6.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(pss, \cdot) = \mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$.

Since pss is initialised with a value uniformly drawn at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_0$ to break the swap-PRF security of $\mathsf{Extract}$. During a given stage, $\mathcal{B}_0$ queries its FUNC oracle in order to compute k_0 (i.e. $q_{\mathrm{FC}} \leq 1$).

We do a similar change at π'' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_5^{d)5)} \leq \mathsf{adv}_6^{d)5)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1)$$

Game$^{d)5)}$ *7.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$. We can do this because k_0 computed by π is now uniformly drawn at random by Game$^{d)5)}$ 6. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the PRF security of Expand. During a given stage, $\mathcal{B}_1$ makes at most two queries to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 2$).

We apply a similar change at π'' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_6^{d)5)} \leq \mathsf{adv}_7^{d)5)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2)$$

Game$^{d)5)}$ *8.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$. We can do this because dk_0 computed by π is now uniformly drawn at random by Game$^{d)5)}$ 7. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the PRF security of $\mathsf{Extract}$. During a given stage, $\mathcal{B}_2$ makes at most one query to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 1$).

We apply a similar change at π'' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_7^{d)5)} \leq \mathsf{adv}_8^{d)5)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1)$$

Game$^{d)5)}$ *9.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this because k_1 computed by π is now uniformly drawn at random by Game$^{d)5)}$ 8. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand. During a given stage, $\mathcal{B}_3$ makes at most 7 queries to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 7$).

We apply a similar change at π'' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_8^{d)5)} \leq \mathsf{adv}_9^{d)5)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)$$

Game$^{d)5)}$ *10.* If π and π'' do not share the same value pss, they compute distinct keys k_0, dk_0, k_1, and hk_R and hk_I unless a collision occurs when π and π'' choose each of the latter parameters. We want to exclude such events. Therefore, the challenger aborts the experiment if any such collision occurs.

When π and π'' do not share pss, the probability that π computes the same value k_0 (or k_1) as π'' is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_0 and k_1. The probability that π computes the same value dk_0 as π'' is $1/2^{\ell_{dk}}$ where ℓ_{dk} is the bitlength of dk_0. The probability that π computes the same value hk_R (or hk_I, both values being uniformly drawn at random by Game$^{d)5)}$ 9) as π'' is $1/2^{\ell_{hk}}$ where ℓ_{hk} is the bitlength of hk_R and hk_I.

Therefore

$$\mathsf{adv}_9^{d)5)} \leq \mathsf{adv}_{10}^{d)5)} + \frac{1}{2^{\ell_k - 1}} + \frac{1}{2^{\ell_{dk}}} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{d)5)}$ *11.* In this game, the challenger aborts the experiment if the instance π receives during the first stage a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it.

By Game$^{d)5)}$ 9, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_4$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_4$ makes at most 1 call to DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_4$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{10}^{d)5)} \leq \mathsf{adv}_{11}^{d)5)} + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1, 1)$$

To that point, if π and π'' do not share the same value pss at the beginning of the first stage, then they compute distinct AE keys corresponding to the first AE

ciphertext received by π (which is sent by π'' since both instances match in stage 1), unless a collision occurs in the computation of one of the intermediary keys or in the computation of the AE keys. Such events are excluded by Game$^{d)5)}$ 10. Therefore, either π and π'' compute the same AE key (which implies that they share the same value pss), or the first AE ciphertext received by π is invalid and rejected by π (hence, it does not appear in the list $\mathsf{m_r}$ of the messages received by π), or the ciphertext is valid but the challenger aborts the experiment by Game$^{d)5)}$ 11. Indeed, the AE header (tr_0 or tr_1) used by π to decrypt the AE ciphertext is computed from epk and ct which are unique (by Game$^{d)5)}$ 2). Therefore, the AE header is also unique (by Game$^{d)5)}$ 1). This means that this AE header is never used by any other honest instance (but π'') to compute an AE ciphertext (and, as said, π'' uses an AE key different from that of used by π). This allows leveraging the INT-CTXT security of the AE function.

Since π and π'' match in stage 1 (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that both instances do share the same value pss at the beginning of stage 1.

Game$^{d)5)}$ 12. In this game, we replace in stage 1 (at the targeted instance π and its matching instance π'') the function $\mathsf{Extract}(dk_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$.

Since dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{d)5)}$ 9) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of $\mathsf{Extract}$.

During the stage, $\mathcal{B}_5$ makes at most 1 call to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 1$). Indeed, the two instances π and π'' use the same input W because they share the ciphertext c'_R (since they are matching in stage 1 by assumption) and by Game$^{d)5)}$ 4, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{d)5)}$ 9 and 11, and the same transcript tr_2 since, again, they are matching).

Therefore

$$\mathsf{adv}_{11}^{d)5)} \leq \mathsf{adv}_{12}^{d)5)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

Game$^{d)5)}$ 13. In this game, we replace in stage 1 (at the targeted instance π and its matching instance π'') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{d)5)}$ 12) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_6$ to break the PRF security of Expand.

We can safely apply this change because the two instances π and π'' do compute the same value k_2. Indeed, the two instances use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{d)5)}$ 12). Moreover, the two instances use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and by Game$^{d)5)}$ 4, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{d)5)}$ 9 and 11, and the same transcripts since, again, they

are matching).

This implies also that, during the stage, $\mathcal{B}_6$ makes at most 3 call to FUNC (i.e. $q_{\text{FC}} \leq 3$).

Therefore

$$\mathsf{adv}_{12}^{d)5)} \leq \mathsf{adv}_{13}^{d)5)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, the two partnered instances π and π'' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{d)5)}$ 9). They share also tr_0 and c_R (because the instances match). Therefore, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π'' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \mathtt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \mathtt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{d)5)}$ 4).

This implies that π and π'' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ (by Game$^{d)5)}$ 13) known only to them, and the same inputs to compute the new value pss. Hence they share this value. Moreover, neither REVEALPSS$(\pi, 0)$ nor REVEALPSS$(\pi'', 0)$ is queried, and REVEALPSS$(\pi, 1)$ and REVEALPSS$(\pi'', 1)$ are not queried either. Therefore, at the end of stage 1, pss is indistinguishable from a value chosen uniformly at random and known only to π and π''.

Now, we guess the stage t.

Game$^{d)5)}$ 14. In this game, the challenger tries to guess the stage t and aborts if it does not succeed. Therefore

$$\mathsf{adv}_{13}^{d)5)} \leq n_T \cdot \mathsf{adv}_{14}^{d)5)}$$

In the subsequent games, we use the swap-PRF security of Extract and the PRF security of Extract and Expand to eventually argue that the value pss at the beginning of stage t is random and known only to π and π''. Since the reasoning relies upon security properties already mentioned, once again we only indicate the properties at stake.

The next six games are repeated for stages u, $1 < u < t$ (in order to make this appear, we denote each of these games as "Game$^{d)5)}$ $n+$"). The corresponding changes are done at π and its matching instance π'' in stages u.

In stages u, $1 < u < t$, π and π'' share the inputs to the different random functions (replacing Extract and Expand) because they match, compute the same intermediary keys (hence use the same random functions), and there is no collision in H_{ec} outputs (by Game$^{d)5)}$ 4). Moreover, recall that REVEALPSS(π, u) and REVEALPSS(π'', u) are not queried where π and π'' match in stages u, $1 < u < t$.

$Game^{d)5)}$ *15+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

$Game^{d)5)}$ *16+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

$Game^{d)5)}$ *17+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

$Game^{d)5)}$ *18+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

$Game^{d)5)}$ *19+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

$Game^{d)5)}$ *20+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π'' (i.e. $\pi.\mathsf{pss}[t-1] = \pi''.\mathsf{pss}[t-1]$).

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{14}^{d)5)} \leq\ & \mathsf{adv}_{20+}^{d)5)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + 2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) \\ & \qquad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7) \Big)\end{aligned}$$

Next, we consider stage t. The corresponding games are the same as those made in case b)2) for stage t.

$Game^{d)5)}$ *21.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss. We replace $\mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ at π and π''. k_0 is now output by $\mathsf{RF}_{\mathsf{Extract}(\cdot,pss)}$ (and $q_{\mathrm{FC}} \leq 1$).

$Game^{d)5)}$ *22.* PRF security of Expand keyed with k_0. We replace $\mathsf{Expand}(k_0,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ at π and π''. dk_0 is now output by $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$).

$Game^{d)5)}$ *23.* PRF security of $\mathsf{Extract}$ keyed with dk_0. We replace $\mathsf{Extract}(dk_0,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ at π and π''. k_1 is now output by $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (and $q_{\mathrm{FC}} \leq 2$ because the simulation may have to query its FUNC oracle twice if π and π'' do not use the same input ss).

Game$^{d)5)}$ *24.* PRF security of Expand keyed with k_1. We replace $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ at π and π''. This change is applied at π'' if π'' shares the same cid as π in stage t. In that case, one query to $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ is enough to get hk_R, n_R and dk_1 for π and π''. However, we cannot ensure that π'' uses the same inputs as π to compute hk_I, n_I, hk'_R, n'_R. Therefore, it may happen that the random function be queried twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Now, we have also to take into account the possibility that π and π'' do not use the same inputs to the random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (cf. Game$^{d)5)}$ 23) and yet output the same value k_1. If π and π'' do not share, in stage t, the same parameters epk, ct, then FUNC may have to be queried once per instance to compute hk_R and n_R. That is, $q_{\mathrm{FC}} \leq 1 + 2 \times 6$.

hk_I and hk'_R are now output by $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

We will use $\mathsf{cid} = epk \| ct$ (and its uniqueness, by Game$^{d)5)}$ 2) only as a "binder" between π and its partner instance in stage t (the adversary might well issue a query REVEALECK(π, t) or REVEALECK(π'', t)).

Game$^{d)5)}$ *25.* In this game, the challenger aborts the experiment if π (as initiator's instance) receives a valid first AE ciphertext (c_R in m_2) not computed by an honest instance.

By Game$^{d)5)}$ 24, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (for π and any honest instance that shares the same cid and pss as π). Therefore, the AE key (hk_R) used to decrypt the ciphertext (c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_7$ to break the INT-CTXT security of the AE function used to decrypt the ciphertext.

During the stage, $\mathcal{B}_7$ makes at most 1 call to DEC in order to decrypt c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same cid (and pss) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_7$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{24}^{d)5)} \leq \mathsf{adv}_{25}^{d)5)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1, 1)$$

Game$^{d)5)}$ *26.* In this game, the challenger aborts the experiment if π receives a valid final message (m_3 or m_4 depending on whether π is a responder's or an initiator's instance) not computed by an honest instance.

By Game$^{d)5)}$ 24, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (for π and any honest instance that shares the same cid and pss as π). Therefore, the AE key (hk_I or hk'_R) used to decrypt the ciphertext (c_I or c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_8$ to break the INT-CTXT security of the AE function used to decrypt the ciphertext.

During the stage, $\mathcal{B}_8$ makes at most 1 call to DEC in order to decrypt c_I or c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the

same cid (and pss) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_8$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{25}^{d)5)} \leq \mathsf{adv}_{26}^{d)5)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

To that point, we have

$$\begin{aligned}\mathsf{adv}_{20+}^{d)5)} \leq \mathsf{adv}_{26}^{d)5)} &+ \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(13) \\ &+ 3 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)\end{aligned}$$

Let us consider first that π is a responder's instance. The adversary must compute either m_1 or m_3 in stage t, as otherwise π'' is an origin instance of π.

- Let us assume that the adversary computes m_1. The KEM ciphertext ct computed by π is unique (by Game$^{d)5)}$ 2). Therefore, no honest instance (but, possibly, π'') uses ct to compute tr_1. This implies that tr_1 is also unique (by Game$^{d)5)}$ 1). In addition, since $m_1 = epk$ is computed by the adversary, π'' does not compute its own value tr_1 from the KEM public key sent by the adversary. Therefore, no honest instance uses the value tr_1 computed by π as AE header to compute an AE ciphertext. Consequently, if the AE ciphertext received by π is invalid, then π aborts the run and does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. In the latter case, the challenger aborts the experiment (by Game$^{d)5)}$ 26).
- Let us assume that the adversary computes m_3 (and does not modify m_1). π decrypts the AE ciphertext with tr_1 as AE header. The latter is computed from epk and ct which are unique (by Game$^{d)5)}$ 2). Therefore, tr_1 is also unique (by Game$^{d)5)}$ 1). This means that no honest instance (besides, possibly, π'') can compute such an AE ciphertext (and the received AE ciphertext does not come from π''). Consequently, if the received AE ciphertext is not valid, then π does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment (by Game$^{d)5)}$ 26).

To sum up, (if the challenger does not abort the experiment) π never accepts and a query TEST(π, t) returns $\perp$.

Let us consider now that π is an initiator's instance. The adversary must compute either m_2 or m_4 in stage t.

- Let us assume that the adversary computes m_2. It must compute either the KEM ciphertext or the AE ciphertext or the MAC tag.
  - Let us assume that the KEM ciphertext is computed by the adversary. π decrypts c_R with tr_0 as AE header. The latter is computed from epk (computed by π) and the received KEM ciphertext. epk is unique (by

$\text{Game}^{d)5)}$ 2). Therefore, tr_0 is also unique (by $\text{Game}^{d)5)}$ 1). Since the adversary computes the KEM ciphertext, π'' does not use the same value ct to compute its own parameter tr_0. Therefore, no honest instance ever uses the value tr_0 computed by π as AE header to compute an AE ciphertext. Consequently, if the AE ciphertext received by π is invalid, then π aborts the run and does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that it is a forgery. Then, the challenger aborts the run (by $\text{Game}^{d)5)}$ 25).

- Let us assume that the first AE ciphertext (c_R) received by π does not come from the honest instance π'' (but ct does). π decrypts c_R with tr_0 as AE header. The latter is computed from epk and ct which are unique (by $\text{Game}^{d)5)}$ 2). Therefore, tr_0 is also unique (by $\text{Game}^{d)5)}$ 1). This means that no honest instance (besides π'') can compute such an AE ciphertext (and the received AE ciphertext does not come from π''). Consequently, if the received AE ciphertext is not valid, then π does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext is a forgery. Then, the challenger aborts the experiment (by $\text{Game}^{d)5)}$ 25).
- Let us assume that the MAC tag τ_R does not come from π'' (and ct and c_R are received unmodified).
  - Let us assume that the KEM public key received by π'' is not the KEM public key computed by π. The KEM public key and ciphertext are unique (by $\text{Game}^{d)5)}$ 2). Therefore, tr_0 computed by π (from its own KEM public key and the received KEM ciphertext, computed by π'') is also unique (by $\text{Game}^{d)5)}$ 1). This means that no honest instance uses the value tr_0 computed by π as AE header to compute an AE ciphertext. If the received AE ciphertext (c_R is invalid, then π aborts the run and does not accept. If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that it is a forgery. Then, the challenger aborts the experiment (by $\text{Game}^{d)5)}$ 25).
  - Let us assume that the KEM public key received by π'' is computed by π. π and π'' share the same value pss at beginning of stage t because they share the same value pss at beginning of stage t' (by $\text{Game}^{d)5)}$ 11). π and π'' share also cid in stage t (i.e. epk and ct). Therefore, the two instances compute the same MAC key and tag τ_R. If the adversary sends $\tilde{\tau_R} \neq \tau_R$, then the received MAC tag is invalid (because we assume that $\mathsf{MAC.Tag}$ is deterministic) and π aborts the run.

– Let us assume that the adversary computes m_4. π decrypts c'_R with tr_2 as AE header. The latter is computed from epk and ct. If epk and ct are shared by π and π'', then no other honest instance but π'' computes the same value tr_2 as π (by $\text{Game}^{d)5)}$ 2 and $\text{Game}^{d)5)}$ 1). If one of these values is not shared by π and π'', then no honest instance ever computes the same value tr_2 as π. This implies that no honest instance (but, possibly, π'') ever uses tr_2 as AE header to compute an AE ciphertext (but the AE ciphertext received by π

is not computed by π''. We can apply a similar reasoning as previously and conclude that either π rejects c'_R because it is invalid (and π aborts the run) or c'_R is valid and is a forgery. Indeed, the only honest instance that could use the same AE header (tr_2) to compute an AE ciphertext is π'' and the latter does not compute the AE ciphertext received by π. Consequently, the challenger aborts the experiment (by Game$^{d)5)}$ 26).

To sum up, (if the challenger does not abort the experiment) π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$.

Hence, the advantage of the adversary in Game$^{d)5)}$ 26 is $\mathsf{adv}_{26}^{d)5)} = 0$.

Collecting all the probabilities from Game$^{d)5)}$ 0 to Game$^{d)5)}$ 26, we have:

$$\begin{aligned}
\mathsf{adv}_0^{d)5)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^\lambda}\right) + (n_P \cdot n_S)^2 \cdot \left(\frac{1}{2^{\ell_k - 1}} + \frac{1}{2^{\ell_{dk}}} + \frac{1}{2^{\ell_{hk}-1}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} \\
& + (n_P \cdot n_S)^2 \cdot \Big(n_T \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(13)\Big) \\
& \quad + (4 \cdot n_T + 1) \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1) \\
& \quad + (n_T(n_T - 1) + 1) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2)\Big) \\
& \quad + (2 \cdot n_T(n_T - 2) + 2) \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) \\
& \quad + (n_T(n_T - 2) + 1) \cdot \Big(\mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)\Big)\Big)
\end{aligned}$$

FreshKey e)1). Now, we proceed with case e)1) in predicate FreshKey).

We consider that the tested instance π has an origin instance π' such that π is not an origin instance of π'. Therefore, π' must receive at least one message not computed by π. We show essentially that the EUF-CMA security property of the signature scheme forbids the adversary from making π accept. That is, since the adversary must alter one of the messages received by π', origin instance of π, then in return the signature computed by π' (based on data chosen by the adversary) cannot be valid for π.

Game$^{e)1)}$ 1. In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}_0^{e)1)} \leq \mathsf{adv}_1^{e)1)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}}$$

Game$^{e)1)}$ 2. In this game, the challenger aborts the experiment if either an ephemeral KEM public key *epk* or ciphertext *ct* is not unique among all instances and stages. This ensures that a matching instance has a unique partner.

There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts),

therefore we have that

$$\mathsf{adv}_1^{e)1)} \leq \mathsf{adv}_2^{e)1)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} \right)$$

$Game^{e)1)}$ *3.* In this game, the challenger aborts the experiment if a collision occurs in $\mathsf{Extract}$ outputs among all instances and stages. Therefore

$$\mathsf{adv}_2^{e)1)} \leq \mathsf{adv}_3^{e)1)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}}$$

$Game^{e)1)}$ *4.* In this game, the challenger aborts the experiment if a collision occurs in Expand outputs among all instances and stages. Therefore

$$\mathsf{adv}_3^{e)1)} \leq \mathsf{adv}_4^{e)1)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}}$$

$Game^{e)1)}$ *5.* In this game, the challenger aborts the experiment if an honest instance receives an AE ciphertext valid under a key distinct from the key used by the sender to compute the ciphertext. This reduces straightforwardly to the key-committing security of the AE scheme. Therefore

$$\mathsf{adv}_4^{e)1)} \leq \mathsf{adv}_5^{e)1)} + \mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}}$$

$Game^{e)1)}$ *6.* In this game, the challenger tries to guess the targeted instance π, its parent and the targeted stage t, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_5^{e)1)} \leq n_P \cdot n_S \cdot n_T \cdot \mathsf{adv}_6^{e)1)}$$

$Game^{e)1)}$ *7.* In this game, the challenger tries to guess which public key (i.e. which corresponding party) the signature received by π is verified with, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_6^{e)1)} \leq n_P \cdot \mathsf{adv}_7^{e)1)}$$

$Game^{e)1)}$ *8.* In this game, the challenger aborts the experiment if the adversary $\mathcal{A}$ produces a valid signature under an uncorrupted key for a message which was not output by an honest instance. This reduces straightforwardly to the security of the signature scheme SIG. We show how to build an adversary $\mathcal{B}_1$ against the EUF-CMA security of the signature scheme. This adversary is given a signing oracle. It behaves as follows. $\mathcal{B}_1$ sets $lpk_w = pk^*$ for some party w, with pk^* from the signature security experiment, and generates all other protocol parameters as specified. It uses its signing oracle in order to output honest signatures valid under lpk_w. If $\mathcal{A}$ submits a valid signature under lpk_w that was not output by an honest instance, then $\mathcal{B}_1$ uses these message and signature to break the EUF-CMA security of the signature scheme. Since this message was not signed before with the corresponding private key (because $\mathcal{B}_1$ did not query its signing oracle with this message to get the signature), then it is a valid forgery (i.e. $\mathcal{B}_1$

wins).

During a given stage, $\mathcal{B}_1$ makes at most 1 call to SIGN when it needs to output a signature for an honest instance (corresponding to party w) (i.e. $q_{\mathrm{SG}} \leq 1$).

$\mathcal{B}_1$ can answer any REVEALLCK query from $\mathcal{A}$ (since $\mathcal{B}_1$ manages the simulation) but for the private key corresponding to $lpk_w = pk^*$. However, in the context where $\mathcal{A}$ tries to compute a valid signature (on behalf of an honest instance owned by some party v) in order to make an honest instance π accept, the long term (signature) key of the putative partner party v cannot be revealed before the honest targeted instance π accepts, as otherwise $\mathsf{FreshAuth}$ becomes false. Therefore, if a query REVEALLCK(w) is issued before the targeted party accepts, then the simulation is stopped.

Taking into account that there are at most n_P parties, we have

$$\mathsf{adv}_7^{e)1)} \leq \mathsf{adv}_8^{e)1)} + \mathsf{adv}_{\mathsf{SIG}}^{\mathsf{euf\text{-}cma}}(1)$$

Let us consider first that the tested instance π is initiator. The adversary must compute either m_1 or m_3.

- If the adversary computes $m_1 = epk$ different from that of computed by π, then π and π' compute distinct values tr_0, tr_1, and eventually distinct values tr_2 (by Game$^{e)1)}$ 1). σ_R must be computed by π' (because π' is an origin instance of π). If the signature is not valid for π, then π does not accept. Let us assume that σ_R is valid for π. π' does not use the same value tr_2 to compute σ_R. Moreover, tr_2 (as computed by π) depends on ct which is unique (by Game$^{e)1)}$ 2). This means that no honest instance ever computes a signature to protect a message that incorporates the same value tr_2 as that of π. Therefore, σ_R is a valid forgery and the challenger aborts the experiment (by Game$^{e)1)}$ 8).
- Let us consider that the adversary computes m_3 (and does not alter m_1 sent by π).
  - If the two instances do not share pss at beginning of stage t, then they compute distinct values k_0 (by Game$^{e)1)}$ 3). Therefore, they compute distinct values dk_0 (by Game$^{e)1)}$ 4). Hence, they compute distinct values k_1 and hk_R. Therefore, π' computes c_R with a different key than that of used by π to verify and decrypt the AE ciphertext. If c_R is not valid for π, then π does not accept. If c_R is valid for π, then the challenger aborts the experiment by Game$^{e)1)}$ 5 (because c_R is then valid under two different keys).
    Moreover, if the KEM decapsulation returns an error, then π aborts the run and does not accept.
  - Let us consider that the two instances do share pss at beginning of stage t.
    - If the KEM decapsulation returns an error, then π stops the run and does not accept.
    - If the KEM decapsulation outputs a value distinct from that of computed by π' during the KEM encapsulation, then the two instances do not use the same AE key hk_R to encrypt and decrypt respectively

c_R (by Game$^{e)1)}$ 3 and Game$^{e)1)}$ 4). Moreover c_R is shared by π and π' because π' is an origin instance of π. If c_R is not valid for π, then π aborts the run and does not accept. If c_R is valid for π, then the challenger aborts the run (by Game$^{e)1)}$ 5).

Let us consider now that both pss (at beginning of stage t) and ss are shared by π and π'. Then, the two instances compute the same AE parameters (hk_I, n_I, tr_1) used to compute and decrypt m_3 (because m_1 and m_2 are also shared by the two instances). Let $\tilde{c_I} \neq c_I$ be the AE ciphertext computed by the adversary (and sent to π'). If $\tilde{c_I}$ is not valid for π', then π' does not accept and π does not either (since π' does not send m_4 and the latter must be computed by π'). If $\tilde{c_I}$ is valid for π', then let $c_I = \mathsf{AE.Enc}(hk_I, \ell_{c_I}, n_I, tr_1, x)$ be the AE ciphertext computed by π, with $\ell_{c_I} = |c_I|$ and $x = Z\|T\|\Sigma\|V\|S\|cert_I\|\sigma_I$.

If $|\tilde{c_I}| \neq |c_I|$, then π' computes a value tr_2 distinct from that of computed by π (by Game$^{e)1)}$ 1).

Let us assume that $|\tilde{c_I}| = |c_I|$. If π' computes $x = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I})$, then $c_I = \mathsf{AE.Enc}(hk_I, |c_I|, n_I, tr_1, x) = \mathsf{AE.Enc}(hk_I, |\tilde{c_I}|, n_I, tr_1, x) = \tilde{c_I}$ (because we assume AE is tidy). But $\tilde{c_I} \neq c_I$. Therefore, π' computes $x' = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I}) \neq x$. Consequently, π and π' compute distinct values tr_2 (by Game$^{e)1)}$ 1).

If σ_R is not valid for π, then π does not accept. Let us assume that σ_R is valid for π. π' does not use the same value tr_2 (as that of computed by π) to output σ_R. Moreover, tr_2 (as computed by π) depends on ct which is unique (by Game$^{e)1)}$ 2). This means that no honest instance ever computes a signature to protect a message that incorporates the same value tr_2 as that of π. Moreover, RevealLCK(π.pid) cannot be queried before π accepts in stage t. Therefore, we can leverage the EUF-CMA security of the signature scheme to claim that σ_R is a valid forgery. Consequently, the challenger aborts the experiment (by Game$^{e)1)}$ 8).

To sum up, when π is an initiator's instance, then π never accepts (when the challenger does not abort the experiment).

Now, let us consider that the tested instance π is responder. The adversary must compute either m_2 or m_4. Below, we consider a forged message m_2 and explain why a forged message m_4 is irrelevant in that context.

- Let us consider that the adversary computes m_2.
  - If π and π' do not share pss (at beginning of stage t) or ss, then they compute distinct values k_0 (by Game$^{e)1)}$ 3). Therefore, they compute distinct values dk_0 (by Game$^{e)1)}$ 4), k_1 and hk_I. Therefore, π' computes c_I with a key different than that of used by π to verify and decrypt the AE ciphertext. If c_I is not valid for π, then π does not accept. If c_I is valid for π, then the challenger aborts the experiment by Game$^{e)1)}$ 5 (because c_I is then valid under two different keys).

    Moreover, if the KEM decapsulation outputs an error, then π' does not send m_3 and π does not accept (since m_3 must come from π').

- Let us consider that pss (at beginning of stage t) and ss are shared by π and π'. The adversary must compute either the AE ciphertext or the KEM ciphertext or the MAC tag in m_2.
  * Let us assume first that the adversary computes the AE ciphertext. The two instances do share pss at beginning of stage t, and ss. Therefore, they output the same AE parameters (hk_R, n_R, tr_0) used to compute and decrypt m_2 (because m_1 is also shared by the two instances). Let $\tilde{c_R}$ be the AE ciphertext computed by the adversary (and sent to π'). If $\tilde{c_R}$ is not valid for π', then the latter does not accept and π does not either (since π' does not send m_3 and the latter must be computed by π'). If $\tilde{c_R}$ is valid for π', then let $c_R = \mathsf{AE.Enc}(hk_R, \ell_{c_R}, n_R, tr_0, x)$ be the AE ciphertext computed by π, with $\ell_{c_R} = |c_R|$ and $x = \Phi_R \| \Omega$.
    If $|\tilde{c_R}| \neq |c_R|$, then π' computes a value tr_1 distinct from that of computed by π (by Game$^{e)1)}$ 1).
    Let us assume that $|\tilde{c_R}| = |c_R|$. If π' computes $x = \mathsf{AE.Dec}(hk_R, n_R, tr_0, \tilde{c_R})$, then $c_R = \mathsf{AE.Enc}(hk_R, |c_R|, n_R, tr_0, x) = \mathsf{AE.Enc}(hk_R, |\tilde{c_R}|, n_R, tr_0, x) = \tilde{c_R}$ (because we assume AE is tidy). But $\tilde{c_R} \neq c_R$. Therefore π' computes $x' = \mathsf{AE.Dec}(hk_R, n_R, tr_0, \tilde{c_R}) \neq x$. Consequently, π and π' compute distinct values tr_1 (by Game$^{e)1)}$ 1).
    Therefore, π' computes σ_I based on a value tr_1 different from that of used by π to verify the signature. If σ_I is not valid for π, then π does not accept. Let us assume that σ_I is valid for π. The value tr_1 (as computed by π) depends on epk which is unique (by Game$^{e)1)}$ 2). This means that no honest instance ever computes a signature to protect a message that incorporates the same value tr_1 as that of π. Moreover, RevealLCK(π.pid) is not queried before π accepts in stage t. Therefore, we can we can leverage the EUF-CMA security of the signature scheme to claim that σ_I is a valid forgery. Consequently, the challenger aborts the experiment (by Game$^{e)1)}$ 8).
  * Let us assume that the adversary computes the KEM ciphertext. If π' receives $\tilde{ct} \neq ct$, computed by the adversary, then π and π' compute distinct values tr_1. Consequently, again, either π does not accept or the challenger aborts the experiment (by Game$^{e)1)}$ 8).
  * Let us assume that the adversary computes the MAC tag. Since π and π' share pss at beginning of stage t, as well as m_1, they compute the same MAC key hk_R^{τ}. If the adversary computes $\tilde{\tau_R}$ (and does not alter ct), then $\tilde{\tau_R} \neq \tau_R$ (where τ_R is computed by π). Therefore, necessarily $\tilde{\tau_R}$ is invalid for π' (because we assume the MAC function $\mathsf{MAC.Tag}$ is deterministic). Hence π' does not accept and π does not either (since it does not receive m_3 which must be computed by π').

– Let us consider that the adversary does not alter m_2. Then, it must forge a valid message m_4 in order for π not to be an origin instance of π'. However, m_2 computed by π is received unmodified by π' (by assumption) and π becomes an origin instance of π' when π computes m_4, which contradicts the assumption of this subcase. Moreover, π' is an origin instance of π.

Therefore, π necessarily accepts, no matter what the adversary does with m_4. Consequently, the adversary has the necessary means (with the allowed $\textsc{RevealPSS}(\pi, t-1)$, $\textsc{RevealECK}(\pi, t)$, $\textsc{RevealEQK}(\pi, t)$ queries) to compute the session keys computed by π and to know b_{key}. That is, the adversary always wins. This explains why we exclude this scenario (it is a trivial attack).

To sum up, when π is a responder's instance, then π never accepts (when the challenger does not abort the experiment).

Remark 7. In the reasoning above, we rely upon the EUF-CMA security of the signature scheme and also on the KEY-CMT property of the AE function. The data that is not shared between by π and its origin instance π' (in particular tr_0, tr_1 or tr_2) is signed by π' but is also protected with the AE scheme and involved in the computation of the AE parameters. Therefore, if a transcript (e.g. tr_0) is not shared between the two instances, then the AE parameters computed from the transcript (e.g. hk_R, n_R) will be different between the two instances (by Game$^{e)1)}$ 4). This implies that the corresponding AE ciphertext (e.g. c_R) is either invalid to π or the challenger aborts the experiment because the key commitment property is then broken. This explains why the AE parameters (key and nonce) are computed with two distinct calls to Expand.

Let us assume, on the contrary, that the two AE parameters (key and nonce) are output with a single call to Expand. If a transcript is not shared by the two instances, then this does not necessarily imply that the AE key is not shared either, but rather that the pair (AE key, AE nonce) is not shared. The KEY-CMT property (corresponding to the CMTD-1 notion of Bellare and Hoang [13]) is then not enough, and the CMTD-3 notion is required (where the adversary must produce two distinct triplets (key, nonce, header) corresponding to the same ciphertext and two (possibly distinct) valid plaintext messages). We choose to use two distinct calls to Expand and to rely on the weaker KEY-CMT property (this is less demanding to the AE function).

Finally, π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$ which does not help the adversary guess the challenge bit b_{key}. Therefore, the adversary can do no better than guess that bit, and $\mathsf{adv}_8^{e)1)} = 0$.

Collecting all the probabilities from Game$^{e)1)}$ 0 to Game$^{e)1)}$ 8 we have:

$$\mathsf{adv}_0^{e)1)} \leq \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{key\text{-}cmt}} + n_P^2 \cdot n_S \cdot n_T \cdot \mathsf{adv}_{\mathsf{SIG}}^{\mathsf{euf\text{-}cma}}(1) + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} \right)$$

FreshKey e)2). Now, we proceed with case e)2) in predicate $\mathsf{FreshKey}$.

We show essentially that the INT-CTXT security property of the AE scheme forbids the adversary from making π accept. That is, the adversary must alter one of the messages received by π', origin instance of π. Therefore, there are two possibilities. Either π' rejects the message (because it is invalid) or the

challenger aborts the run (because the INT-CTXT property of the AE scheme is then broken).

$Game^{e)2)}$ *1.* In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}_0^{e)2)} \leq \mathsf{adv}_1^{e)2)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}}$$

$Game^{e)2)}$ *2.* In this game, the challenger aborts the experiment if either an ephemeral KEM public key *epk* or ciphertext *ct* is not unique among all instances and stages. This ensures that the targeted instance shares its cid with at most one instance.

There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_1^{e)2)} \leq \mathsf{adv}_2^{e)2)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$

$Game^{e)2)}$ *3.* In this game, the challenger aborts the experiment if a KEM decapsulation failure occurs. That is, the challenger aborts the experiment if, on the one hand, the encapsulation done by an honest instance with input an ephemeral public key *epk* outputs a ciphertext *ct* and a shared secret *ss*, and, on the other hand, upon reception of *ct* by the honest instance that has sent *epk* the decapsulation with the corresponding private key *esk* outputs a value $ss' \neq ss$. We have that

$$\mathsf{adv}_2^{e)2)} \leq \mathsf{adv}_3^{e)2)} + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}$$

$Game^{e)2)}$ *4.* In this game, the challenger aborts the experiment if a collision occurs in $\mathsf{Extract}$ outputs among among all instances and stages. Therefore

$$\mathsf{adv}_3^{e)2)} \leq \mathsf{adv}_4^{e)2)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}}$$

$Game^{e)2)}$ *5.* In this game, the challenger aborts the experiment if a collision occurs in Expand outputs among among all instances and stages. Therefore

$$\mathsf{adv}_4^{e)2)} \leq \mathsf{adv}_5^{e)2)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}}$$

$Game^{e)2)}$ *6.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, and the instance π' that shares the same cid as π and its parent, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_5^{e)2)} \leq (n_P \cdot n_S)^2 \cdot n_T \cdot \mathsf{adv}_6^{e)2)}$$

Game$^{e)2)}$ *7.* In this game, we replace in stage t the shared secret ss computed by π with a value $\tilde{ss}$ of same size uniformly drawn at random. We do the same for π' in stage t. Recall that in stage t, π and π' share the same cid. Hence, they share the same KEM public key epk and ciphertext ct. Moreover, by Game$^{e)2)}$ 3, decapsulation failures are excluded. Consequently, π and π' share the same value ss in stage t. In addition, no query $\textsc{RevealECK}(\pi, t)$ nor $\textsc{RevealECK}(\pi', t)$ is issued. Therefore, detecting the change implies an adversary $\mathcal{B}_1$ able to break the IND-CPA security of the KEM scheme. Here we rely upon the IND-CPA notion because the two instances π and π' do share the KEM public key and ciphertext. Therefore, we have

$$\mathsf{adv}_6^{e)2)} \leq \mathsf{adv}_7^{e)2)} + \mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}}$$

Game$^{e)2)}$ *8.* In this game, we replace (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot) = \mathsf{Extract}(\cdot, \tilde{ss})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$. We do a similar change at π' (which shares the same cid as well as the same shared secret ss as π).

Since $\tilde{ss}$ is sampled uniformly at random by Game$^{e)2)}$ 7 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_2$ queries its $\textsc{Func}$ oracle to get k_1. However, to that point we have no guarantee that the instances π and π' share the same value pss at the beginning of stage t. If not, the respective inputs dk_0 to $\textsc{Func}$ are different for π and π' (because, by Games$^{e)2)}$ 4 and 5, no collision occurs in $\mathsf{Extract}$ or Expand during the computation of k_0 and dk_0). Therefore, we have to count for at most 2 queries to $\textsc{Func}$ (i.e. $q_{\textsc{Fc}} \leq 2$).

Therefore

$$\mathsf{adv}_7^{e)2)} \leq \mathsf{adv}_8^{e)2)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2)$$

Game$^{e)2)}$ *9.* As explained in Game$^{e)2)}$ 8, to that point, we cannot ensure that the two instances π and π' compute the same value dk_0, input to compute k_1. It may happen that using distinct values dk_0 as input to $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$, the outputs be equal. We want to preclude such an event. The probability that such a collision occurs at π is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_1. Therefore, in this game, the challenger aborts the experiment if such a collision occurs.

Hence,

$$\mathsf{adv}_8^{e)2)} \leq \mathsf{adv}_9^{e)2)} + \frac{1}{2^{\ell_k}}$$

Game$^{e)2)}$ *10.* In this game, the challenger aborts the experiment if one of the two keys k_0 computed by π and π' respectively is equal to the key k_1 computed by π. Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Expand is keyed with distinct keys. Then, we will be able to safely replace Expand with a random function when keyed with a specific (unique) key, and still rely upon the CR

property of Expand when keyed with another specific (unique) key.

By Game$^{e)2)}$ 8, the ℓ_k-bit long key k_1 computed respectively by π is output by a random function. Therefore, we have that

$$\mathsf{adv}_9^{e)2)} \leq \mathsf{adv}_{10}^{e)2)} + \frac{2}{2^{\ell_k}}$$

Game$^{e)2)}$ 11. In this game, we replace (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this, because k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{e)2)}$ 8). We apply a similar change at π' if the latter shares the same value pss as π.

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand.

During the stage, $\mathcal{B}_3$ queries its oracle FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1. Since π and π' share cid, one query to FUNC is enough to get hk_R, n_R and dk_1 for π and π' (if the latter uses the same value pss as π). However, we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I, hk'_R and n'_R. Therefore, it may happen that $\mathcal{B}_3$ queries its oracle twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Therefore

$$\mathsf{adv}_{10}^{e)2)} \leq \mathsf{adv}_{11}^{e)2)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(11)$$

Game$^{e)2)}$ 12. By Game$^{e)2)}$ 9, if π and π' use distinct inputs dk_0, then they compute distinct values k_1 (i.e. π' does not use the same function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ as π). We want to exclude the event that, while π and π' use distinct functions, they compute the same AE key hk_R needed to compute and decrypt (depending on whether π is an initiator's or responder's instance) the AE ciphertext c_R sent or received by π. By Game$^{e)2)}$ 11, the probability that π computes this same AE key as π' in such a case is $1/2^{\ell_{hk}}$, where ℓ_{hk} is the bitlength of hk_R.

Therefore, in this game the challenger aborts the experiment if such a collision occurs.

Hence

$$\mathsf{adv}_{11}^{e)2)} \leq \mathsf{adv}_{12}^{e)2)} + \frac{1}{2^{\ell_{hk}}}$$

Game$^{e)2)}$ 13. In this game, the challenger aborts the experiment if the instance π, when behaving as initiator's instance, receives a valid first AE ciphertext (c_R) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it.

By Game$^{e)2)}$ 11, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_R) used to decrypt the ciphertext (c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_4$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_4$ makes at most 1 call to DEC in order to decrypt c_R

(i.e. $q_{\mathrm{D}} \leq 1$). The ciphertext comes from π' since the latter is an origin instance of π. If π' uses the same value pss as π (in addition to the same cid), then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_4$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{12}^{e)2)} \leq \mathsf{adv}_{13}^{e)2)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Game$^{e)2)}$ *14.* If π and π' do not share pss at beginning of stage t, then they compute distinct values k_1 (by Game$^{e)2)}$ 9). Still, k_1, computed by π' is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{e)2)}$ 8). Therefore, in this game, we replace (at the origin instance π' when π behaves as a responder's instance) the function $\mathsf{Expand}(k_1,\cdot)$ with a random function $\mathsf{RF}'_{\mathsf{Expand}(k_1,\cdot)}$.

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of Expand.

During the stage, $\mathcal{B}_5$ queries its oracle Func to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1 (i.e. $q_{\mathrm{Fc}} \leq 7$).

Therefore

$$\mathsf{adv}_{13}^{e)2)} \leq \mathsf{adv}_{14}^{e)2)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)$$

Game$^{e)2)}$ *15.* In this game, the challenger aborts the experiment if the instance π', when π behaves as a responder's instance, receives a valid first AE ciphertext (c_R) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π' to decrypt it.

By Game$^{e)2)}$ 14, $\mathsf{Expand}(k_1,\cdot)$ is replaced with a random function $\mathsf{RF}'_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_R) used to decrypt the ciphertext (c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_6$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_6$ makes at most 1 call to Dec in order to decrypt c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss as π', then both instances use the same random function to compute the AE key. If this instance uses the same inputs to the random function as π', then the two instances compute the same AE key. In such a case, $\mathcal{B}_6$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$). Note that this message may come from the adversary (and not from an honest instance). However, even if π' does not receive the honest message, then $\mathcal{B}_6$ may have to compute such a message for an honest "sending" instance (which would then be "replaced" by the adversary with a message of its choice).

Consequently, we have

$$\mathsf{adv}_{14}^{e)2)} \leq \mathsf{adv}_{15}^{e)2)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Game$^{e)2)}$ *16.* In this game, the challenger aborts the experiment if the instance π, when it behaves as initiator, receives a valid last AE ciphertext (c'_R) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{e)2)}$ 11, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk'_R) used to decrypt the ciphertext (c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_7$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_7$ makes at most 1 call to its oracle DEC in order to decrypt c'_R (i.e. $q_{\mathrm{D}} \leq 1$). Since π has an origin instance, the received AE ciphertext must come from that (honest) instance. Therefore, $\mathcal{B}_7$ may have to compute such a message for that honest instance if the latter uses the same AE key as π. In such a case, the oracle ENC is queried at most 1 time (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{15}^{e)2)} \leq \mathsf{adv}_{16}^{e)2)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Game$^{e)2)}$ *17.* In this game, the challenger aborts the experiment if the instance π', when π behaves as a responder's instance, receives a valid final AE ciphertext (c'_R) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π' to decrypt it.

By Game$^{e)2)}$ 14, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}'_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk'_R) used to decrypt the ciphertext (c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_8$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_8$ makes at most 1 call to DEC in order to decrypt c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value *pss* as π', then both instances use the same random function to compute the AE key. If this instance uses the same inputs to the random function as π', then the two instances compute the same AE key. In such a case, $\mathcal{B}_8$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$). Note that this message may come from the adversary (and not from an honest instance). However, even if π' does not receive the honest message, then $\mathcal{B}_8$ may have to compute such a message for an honest "sending" instance (which would then be "replaced" by the adversary with a message of its choice).

Consequently, we have

$$\mathsf{adv}_{16}^{e)2)} \leq \mathsf{adv}_{17}^{e)2)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Let us consider first that the tested instance π is initiator. The adversary cannot compute m_1 because we assume that π and its origin instance π' share cid. Let us consider that the adversary computes m_3.

– If π and π' do not share the same value pss at beginning of stage t, then they compute distinct AE keys (hk_R) to encrypt and decrypt the first AE ciphertext (c_R) received by π, unless a collision occurs between the keys computed by π and π' respectively. But such an event is excluded by Game$^{e)2)}$ 12. π verifies the received AE ciphertext c_R against the value tr_0 it has computed. tr_0 is unique (by Game$^{e)2)}$ 1) because it is computed from epk (also unique by Game$^{e)2)}$ 2). Therefore, no honest instance (besides π') uses tr_0 to compute an AE ciphertext. But π' uses an AE key to compute c_R different from that of used by π to verify c_R. Therefore, either the first AE ciphertext received by π is invalid and rejected by π (and the run is aborted), or the ciphertext is valid but the challenger aborts the experiment by Game$^{e)2)}$ 13 because c_R is then a valid AE forgery.
  Moreover, if the KEM decapsulation does not return an error (by Game$^{e)2)}$ 3). Hence, π does not abort the run upon the KEM decapsulation.
– Let us consider now that pss is shared by π and π' at beginning of stage t. The two instances share also ss (by Game$^{e)2)}$ 3). Therefore, the two instances use the same AE parameters (hk_R, n_R, tr_0) to compute and decrypt c_R. Consequently, they share the same AE parameters (hk_I, n_I, tr_1) used to compute and decrypt m_3 (because m_1 and m_2 are also shared by the two instances). Let $\tilde{c_I} \neq c_I$ be the AE ciphertext computed by the adversary (and sent to π'). If $\tilde{c_I}$ is not valid for π', then π' does not accept and π does not either (since π' does not send m_4 and the latter must be computed by π'). If $\tilde{c_I}$ is valid for π', then let $c_I = \mathsf{AE.Enc}(hk_I, \ell_{c_I}, n_I, tr_1, x)$ be the AE ciphertext computed by π, with $\ell_{c_I} = |c_I|$ and $x = Z\|T\|\Sigma\|V\|S\|cert_I\|\sigma_I$.
  If $|\tilde{c_I}| \neq |c_I|$, then π' computes a value tr_2 distinct from that of computed by π (by Game$^{e)2)}$ 1).
  Let us assume that $|\tilde{c_I}| = |c_I|$. If π' computes $x = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I})$, then $c_I = \mathsf{AE.Enc}(hk_I, |c_I|, n_I, tr_1, x) = \mathsf{AE.Enc}(hk_I, |\tilde{c_I}|, n_I, tr_1, x) = \tilde{c_I}$ (because we assume AE is tidy). But $\tilde{c_I} \neq c_I$. Therefore, π' computes $x' = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I}) \neq x$. Consequently, π and π' compute distinct values tr_2 (by Game$^{e)2)}$ 1).
  By Game$^{e)2)}$ 2, epk is unique. Therefore, tr_2 (computed by π) is also unique (by Game$^{e)2)}$ 1). Moreover, tr_2 is not shared by π'. Consequently, there exists no honest instance that ever uses tr_2 as AE header to compute an AE ciphertext. If c'_R (computed by π' and decrypted by π with tr_2 as AE header) is not valid for π, then π does not accept. If c'_R is valid for π, then we can leverage the INT-CTXT security of the AE function to claim that c'_R is a valid AE forgery. Then, the challenger aborts the experiment (by Game$^{e)2)}$ 16).

Now, let us consider that the tested instance π is responder. The adversary must compute either m_2 or m_4.

– If π and π' do not share the same value pss at beginning of stage t, then they compute distinct AE keys (hk_R) to encrypt and decrypt the first AE ciphertext received by π', unless a collision occurs between the keys computed by π and π' respectively. But such an event is excluded by Game$^{e)2)}$ 12. π'

verifies the received AE ciphertext against the value tr_0 it has computed. tr_0 is unique (by $\text{Game}^{e)2)}$ 1) because it is computed from epk (also unique by $\text{Game}^{e)2)}$ 2). Therefore, no honest instance (besides π) uses tr_0 as AE header to compute an AE ciphertext. But π uses anyway an AE key to compute c_R different from that of used by π to verify the received AE ciphertext (be the latter sent by π or by the adversary). Therefore, either the first AE ciphertext received by π' is invalid and rejected by π' (and the run is aborted), or the ciphertext is valid but the challenger aborts the experiment by $\text{Game}^{e)2)}$ 15 because the first AE ciphertext received by π' is then a valid AE forgery. If π' aborts the run, then π does not accept (because m_3 received by π must be computed by π').
Moreover, if the KEM decapsulation does not return an error (by $\text{Game}^{e)2)}$ 3). Hence, π' does not abort the run upon the KEM decapsulation.

– Let us assume now that pss is shared by π and π' at beginning of stage t. The two instances share also ss (by $\text{Game}^{e)2)}$ 3).
  - Let us consider that the adversary computes m_2. That is, it computes either the AE ciphertext or the MAC tag in m_2 (because it cannot alter the KEM ciphertext since π and π' share cid in stage t).
    * Let us assume first that the adversary computes the AE ciphertext. The two instances do share pss at beginning of stage t, and ss. Therefore, they output the same AE parameters (hk_R, n_R, tr_0) used to compute and decrypt the AE ciphertext (because m_1 is also shared by the two instances). Let $\tilde{c_R}$ be the AE ciphertext computed by the adversary (and sent to π'). We have $\tilde{c_R} \neq c_R$ where c_R is computed by π. If $\tilde{c_R}$ is not valid for π', then the latter does not accept and π does not either (since π' does not send m_3 and the latter must be computed by π'). Let us assume that $\tilde{c_R}$ is valid for π'. As said, π' uses tr_0 as AE header to decrypt $\tilde{c_R}$. tr_0 is computed from epk which is unique (by $\text{Game}^{e)2)}$ 2). Therefore, tr_0 is also unique (by $\text{Game}^{e)2)}$ 1) and shared only by π and π' (because they share also cid). However, π' uses tr_0 to compute an AE ciphertext (c_R) distinct from that of received by π ($\tilde{c_R}$). Therefore, no honest instance ever computes $\tilde{c_R}$ using tr_0 as AE header. Consequently, we can leverage the INT-CTXT security of the AE function to claim that $\tilde{c_R}$ is a valid forgery. Then, the challenger aborts the experiment (by $\text{Game}^{e)2)}$ 15).
    * Let us assume that the adversary computes the MAC tag. Since π and π' share pss at beginning of stage t, as well as m_1, they compute the same MAC key hk_R^τ. Moreover, ct is shared by π and π'. Therefore, π and π' use the same inputs to compute and verify the MAC tag. If the adversary computes $\tilde{\tau_R}$, then $\tilde{\tau_R} \neq \tau_R$ (where τ_R is computed by π). Therefore, necessarily $\tilde{\tau_R}$ is invalid for π' (because we assume the MAC function $\mathsf{MAC.Tag}$ is deterministic). Hence π' does not accept and π does not either (since it does not receive m_3 which must be computed by π').

- Let us consider that the adversary computes m_4. Let $\tilde{c'_R} \neq c'_R$ be the value computed by the adversary, where c'_R is the value computed by π.
  - If $\tilde{c'_R}$ is not valid for π', then $\tilde{c'_R}$ does not belong to $\pi'.\mathsf{m_r}[t]$. Therefore, π is an origin instance of π' (because m_2 is received by π' unmodified by the adversary). This contradicts the assumption of subcase e)2).
  - Let us assume that $\tilde{c'_R}$ is valid for π'. π' uses the same parameters hk'_R, n'_R as π because both instances share the same value pss at beginning of stage t as well as the messages m_1, m_2, m_3, and the KEM shared secret ss (by Game$^{e)2)}$ 3). Hence, π and π' share also the same value tr_2. Moreover, tr_2 is computed from epk which is unique (by Game$^{e)2)}$ 2). Therefore, tr_2 is also unique (by Game$^{e)2)}$ 1). This means that no honest instance (besides π) ever computes an AE ciphertext with tr_2 as AE header. Consequently, we can leverage the INT-CTXT security of the AE function to claim that $\tilde{c'_R}$ is a valid forgery. Then, the challenger aborts the experiment (by Game$^{e)2)}$ 17).

  To sum up, this scenario (the adversary computes m_4) cannot be considered because it contradicts the assumption of subcase e)2).

Finally, π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$ which does not help the adversary guess the challenge bit b_{key}. Therefore, the adversary can do no better than guess that bit, and $\mathsf{adv}_{17}^{e)2)} = 0$.

Collecting all the probabilities from Game$^{e)2)}$ 0 to Game$^{e)2)}$ 17 we have:

$$\begin{aligned}\mathsf{adv}_0^{e)2)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}} + (n_P \cdot n_S)^2 \cdot n_T \cdot \left(\frac{1}{2^{\ell_{hk}}} + \frac{3}{2^{\ell_k}}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\ & + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} \\ & + (n_P \cdot n_S)^2 \cdot n_T \cdot \Big(\mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7) \\ & \quad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(11) + 4 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)\Big)\end{aligned}$$

FreshKey e)3). Now, we proceed with case e)3) in predicate FreshKey.

We show essentially that the KEY-CMT security property of the AE scheme forbids the adversary from making π accept. That is, the adversary must alter one of the messages received by π', origin instance of π. Therefore, there are two possibilities. Either π rejects the message (because it is invalid) or the challenger aborts the run (because the KEY-CMT property of the AE scheme is then broken).

Game$^{e)3)}$ 1. In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}_0^{e)3)} \leq \mathsf{adv}_1^{e)3)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}}$$

$Game^{e)3)}$ *2.* In this game, the challenger aborts the experiment if a collision occurs in outputs of the chosen 2-universal hash function H_{ec}. These outputs are λ-bit long. Therefore, by definition of H_{ec} (cf. Definition 24), we have

$$\mathsf{adv}_1^{e)3)} \leq \mathsf{adv}_2^{e)3)} + \frac{n_P \cdot n_S \cdot n_T}{2^\lambda}$$

where the multiplicative factor $n_P \cdot n_S \cdot n_T$ comes from the fact that a new 2-universal hash function H_{ec} is chosen during each stage.

$Game^{e)3)}$ *3.* In this game, the challenger aborts the experiment if a collision occurs in $\mathsf{Extract}$ outputs among all instances and stages. Therefore

$$\mathsf{adv}_2^{e)3)} \leq \mathsf{adv}_3^{e)3)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}}$$

$Game^{e)3)}$ *4.* In this game, the challenger aborts the experiment if a collision occurs in Expand outputs among all instances and stages. Therefore

$$\mathsf{adv}_3^{e)3)} \leq \mathsf{adv}_4^{e)3)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}}$$

$Game^{e)3)}$ *5.* In this game, the challenger aborts the experiment if an honest instance receives an AE ciphertext valid under a key distinct from the key used by the sender to compute the ciphertext. This reduces straightforwardly to the key-committing security of the AE scheme. Therefore

$$\mathsf{adv}_4^{e)3)} \leq \mathsf{adv}_5^{e)3)} + \mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}}$$

$Game^{e)3)}$ *6.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, and the instance π' that is an origin instance of π in stage t and its parent, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_5^{e)3)} \leq (n_P \cdot n_S)^2 \cdot n_T \cdot \mathsf{adv}_6^{e)3)}$$

$Game^{e)3)}$ *7.* In this game, we replace the ephemeral quantum secret computed in stage t by the tested instance π with a value $\tilde{K}$ of same size uniformly drawn at random. We apply this change only if π and π' share the messages m_1, m_2 and m_3, unmodified by the adversary. We apply the same change with $\tilde{K}$ at π' if the latter computes the same value W as π.

The messages m_1, m_2, m_3 transport (among other parameters) the classical data related to the QKD phase. If these messages are computed by π and π' and remain unaltered by the adversary, then their integrity is ensured. Recall also that, by assumption, $\pi.\mathsf{eqk}[t] \neq \perp$ and $\pi'.\mathsf{eqk}[t] \neq \perp$. Therefore, this implies that π does compute a QKD secret value, and we can leverage the proof of Tomamichel and Leverrier [109].

By Tomamichel and Leverrier [109], the change is indistinguishable to the adversary. In addition, neither $\textsc{RevealEQK}(\pi, t)$ nor $\textsc{RevealEQK}(\pi', t)$ is queried.

Tomamichel and Leverrier [109] prove the security of the BB84 protocol based on the following experiment. On the one hand (real context), the real BB84 protocol is executed. On the other hand (ideal context), a BB84 run is executed and the output key is replaced with a value of same length uniformly drawn at random. The maximum probability of distinguishing the context is $p_\Delta = (1 + \epsilon_\Delta)/2$ (see Section 4.7). By Game$^{e)3)}$ 2, collisions in H_{ec} are excluded. Therefore, we can remove the corresponding probability (correctness component) $\epsilon_{ec} = 2^{-\lambda}$ from ϵ_Δ and consider instead the secrecy component only $\epsilon_s = \epsilon_\Delta - \epsilon_{ec}$, as well as the probability $p_s = (1 + \epsilon_s)/2$.

Let F be the event that the adversary detects the change made in this game. We have

$$\begin{aligned}
\mathsf{adv}_6^{e)3)} &\leq \mathsf{adv}_7^{e)3)} + \Pr[\mathcal{A} \text{ wins} \mid \text{real QKD context}] \\
&\qquad - \Pr[\mathcal{A} \text{ wins} \mid \text{ideal QKD context}] \\
&\leq \mathsf{adv}_7^{e)3)} + \Pr[F] \\
&\leq \mathsf{adv}_7^{e)3)} + 2 \cdot p_s - 1 \\
&= \mathsf{adv}_7^{e)3)} + \epsilon_s
\end{aligned}$$

To that point, in stage t, π (and possibly π') uses $\tilde{K}$ to key $\mathsf{Extract}^{\mathsf{swap}}$ in order to compute k_2.

Game$^{e)3)}$ 8. In this game, we replace in stage t at the targeted instance π the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{K}, \cdot) = \mathsf{Extract}(\cdot, \tilde{K})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$. We apply this change only if π and π' share the messages m_1, m_2 and m_3, unmodified by the adversary. We apply the same change at π' if it uses the same value W as π in stage t.

Since $\tilde{K}$ is uniformly drawn at random by Game$^{e)3)}$ 7 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_1$ queries Func to compute k_2. We cannot ensure that π and π' use the same value dk_1. Therefore, there might be (at most) 2 calls to Func (i.e. $q_{\mathrm{Fc}} \leq 2$).

Therefore

$$\mathsf{adv}_7^{e)3)} \leq \mathsf{adv}_8^{e)3)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2)$$

Game$^{e)3)}$ 9. In this game, the challenger aborts the experiment if a collision occurs between k_2, computed by π, and one of the fours keys k_0, k_1 computed by π and π' respectively. Indeed, we will need to rely upon the CR property of function Expand but also on the PRF security of Expand (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Expand is keyed with distinct keys. The, we will be able to safely replace Expand with a random function when keyed with a specific key, and still rely upon the CR property of Expand when keyed with another specific key.

We apply this change only if π and π' share the messages m_1, m_2 and m_3,

unmodified by the adversary.

By Game$^{e)3)}$ 8, the ℓ_k-bit key k_2 computed by π is output by a random function. Therefore

$$\mathsf{adv}_8^{e)3)} \leq \mathsf{adv}_9^{e)3)} + \frac{4}{2^{\ell_k}}$$

Remark 8. We recall the assumption on the size of the KEM shared secret ss, the secret internal state pss and the output of the QKD phase W: the sizes are distinct. Hence, no collision can occur between ss, pss and W, and we can replace some $\mathsf{Extract}(\cdot, x)$ functions with random functions and rely upon the CR property of other $\mathsf{Extract}(\cdot, x')$ functions, with $x, x' \in \{ss, pss, W\}$, $x \neq x'$.

Game$^{e)3)}$ 10. In this game, we replace in stage t the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$. We apply this change only if π and π' share the messages m_1, m_2 and m_3, unmodified by the adversary. We apply the same change at π' if it shares the same value k_2 as π in stage t.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$ (by Game$^{e)3)}$ 8) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_2$ queries FUNC to get tk_I, tk_R and pss. We cannot ensure that π and π' use the same value tr_3. Therefore, there might be (at most) 2×3 calls to FUNC (i.e. $q_{\mathrm{FC}} \leq 6$).

Therefore

$$\mathsf{adv}_9^{e)3)} \leq \mathsf{adv}_{10}^{e)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(6)$$

Let us consider first that π is initiator and π' responder. Since π is not an origin instance of π' (by assumption), the adversary must compute m_1 or m_3.

- Let us assume that the adversary computes m_1. It sends to π' a value epk distinct from that of computed by π.
  - Upon reception of the KEM ciphertext computed by π', if the decapsulation outputs an error, then π aborts the run.
  - Otherwise, π and π' compute distinct values hk_R (by Game$^{e)3)}$ 1 and Game$^{e)3)}$ 4). If c_R computed by π' is invalid, then π aborts the run. If c_R is valid, then the challenger aborts the experiment (by Game$^{e)3)}$ 5).
- Let us assume that the adversary computes m_3 (and does not alter m_1).
  - Let us assume that π and π' do not share ss or pss at beginning of stage t. Then, π computes hk_R distinct from that of computed by π' (by Game$^{e)3)}$ 3 and Game$^{e)3)}$ 4). If c_R computed by π' is invalid, then π aborts the run. If c_R is valid, then the challenger aborts the experiment (by Game$^{e)3)}$ 5).
  - Now, we assume that ss and pss (at beginning of stage t) are shared, by π and π'. In that case, π and π' compute the same AE parameters hk_I, n_I, tr_1 because they share pss (by assumption), m_1 (by assumption) and m_2 (because π' is an origin instance of π). Let $\tilde{c_I} \neq c_I$ be the AE ciphertext computed by the adversary (and sent to π'). If $\tilde{c_I}$ is not valid for π', then π' does not accept and π does not either (since π' does not send m_4 and the latter must be computed by π'). If $\tilde{c_I}$ is valid for π',

then let $c_I = \mathsf{AE.Enc}(hk_I, \ell_{c_I}, n_I, tr_1, x)$ be the AE ciphertext computed by π, with $\ell_{c_I} = |c_I|$ and $x = Z\|T\|\Sigma\|V\|S\|cert_I\|\sigma_I$.
If $|\tilde{c_I}| \neq |c_I|$, then π' computes a value tr_2 distinct from that of computed by π (by Game$^{e)3)}$ 1).
Let us assume that $|\tilde{c_I}| = |c_I|$. If π' computes $x = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I})$, then $c_I = \mathsf{AE.Enc}(hk_I, |c_I|, n_I, tr_1, x) = \mathsf{AE.Enc}(hk_I, |\tilde{c_I}|, n_I, tr_1, x) = \tilde{c_I}$ (because we assume AE is tidy). But $\tilde{c_I} \neq c_I$. Therefore, π' computes $x' = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I}) \neq x$. Consequently, π and π' compute distinct values tr_2 (by Game$^{e)3)}$ 1).
Since π and π' use distinct values tr_2, they compute distinct keys hk'_R (by Game$^{e)3)}$ 4). If π receives $m_4 = c'_R$, then it is computed by π' (because π' is an origin instance of π). If c'_R is invalid, then π aborts the run. If c'_R is valid, then the challenger aborts the experiment (by Game$^{e)3)}$ 5).

To sum up, π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$.

Let us consider that π is responder and π' is initiator. The adversary must compute either m_2 or m_4.

- Let us assume that π and π' do not share ss or pss at beginning of stage t. Then, π computes hk_I distinct from that of computed by π' (by Game$^{e)3)}$ 3 and Game$^{e)3)}$ 4). If c_I computed by π' is invalid, then π aborts the run. If c_I is valid, then the challenger aborts the experiment (by Game$^{e)3)}$ 5).
- Now, we assume that ss and pss (at beginning of stage t) are shared by π and π'.
  - Let us assume that the adversary computes m_2. The adversary must compute either the KEM ciphertext, the AE ciphertext or the MAC tag in m_2.
    - Let us assume that the adversary computes the KEM ciphertext. If the decapsulation outputs an error, then π' aborts the run and π never accepts (since the messages received by π must come from its origin instance π'). Otherwise, π and π' compute distinct values hk_I (by Game$^{e)3)}$ 1 and Game$^{e)3)}$ 4). If c_I (computed by π') is invalid, then π aborts the run. If c_I is valid, then the challenger aborts the experiment (by Game$^{e)3)}$ 5).
    - Let us assume that the adversary computes the MAC tag (and does not alter ct). The two instances compute the same key hk^{τ}_R because they share pss (by assumption), ct (by assumption) and epk (because π' is an origin instance of π). Necessarily, the MAC tag $\tilde{\tau_R} \neq \tau_R$ computed by the adversary (where τ_R is computed by π) is not valid (because we assume the MAC function $\mathsf{MAC.Tag}$ is deterministic). Therefore, π' aborts the run and π never accepts.
    - Let us assume that the adversary computes the AE ciphertext (and does not alter the MAC tag, nor the KEM ciphertext). π and π' share hk_R, n_R, tr_0 because they share ss, pss and ct (by assumption), and epk (because π' is an origin instance of π). Let $\tilde{c_R}$ be the AE ciphertext computed by the adversary (and sent to π'). If $\tilde{c_R}$ is not valid for π', then the latter does not accept and π does not either

(since π' does not send m_3 and the latter must be computed by π'). If $\tilde{c_R}$ is valid for π', then let $c_R = \mathsf{AE.Enc}(hk_R, \ell_{c_R}, n_R, tr_0, x)$ be the AE ciphertext computed by π, with $\ell_{c_R} = |c_R|$ and $x = \Phi_R \| \Omega$.
If $|\tilde{c_R}| \neq |c_R|$, then π' computes a value tr_1 distinct from that of computed by π (by Game$^{e)3)}$ 1).
Let us assume that $|\tilde{c_R}| = |c_R|$. If π' computes $x = \mathsf{AE.Dec}(hk_R, n_R, tr_0, \tilde{c_R})$, then $c_R = \mathsf{AE.Enc}(hk_R, |c_R|, n_R, tr_0, x) = \mathsf{AE.Enc}(hk_R, |\tilde{c_R}|, n_R, tr_0, x) = \tilde{c_R}$ (because we assume AE is tidy). But $\tilde{c_R} \neq c_R$. Therefore π' computes $x' = \mathsf{AE.Dec}(hk_R, n_R, tr_0, \tilde{c_R}) \neq x$. Consequently, π and π' compute distinct values tr_1 (by Game$^{e)3)}$ 1).
Since π and π' use distinct values tr_1, they compute distinct keys hk_I (by Game$^{e)3)}$ 4). If c_I (computed by π') is invalid, then π aborts the run. If c_I is valid, then the challenger aborts the experiment (by Game$^{e)3)}$ 5).

- Let us assume that the adversary computes m_4 (and does not alter m_2). The messages m_1 and m_3 computed by π' are received by π (because π' is an origin instance of π). The message m_2 computed by π is received by π' (by assumption). Therefore, the integrity of the messages m_1, m_2 and m_3 is ensured, and the three messages are shared by π and π'. Consequently, the change described in Game$^{e)3)}$ 10 is applied. That is, the session keys are output by a random function. Therefore, the adversary has no better probability of guessing b_{key} than $1/2$ and its advantage in Game$^{e)3)}$ 10 is $\mathsf{adv}_{10}^{e)3)} = 0$.

Overall, either the tested instance π does not accept or the session keys are random. This implies that $\mathsf{adv}_{10}^{e)3)} = 0$.

Collecting all the probabilities from Game$^{e)3)}$ 0 to Game$^{e)3)}$ 10, we have:

$$\mathsf{adv}_0^{e)3)} \leq \frac{n_P \cdot n_S \cdot n_T}{2^\lambda} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{key\text{-}cmt}} + (n_P \cdot n_S)^2 \cdot n_T \cdot \left(\epsilon_s + \frac{1}{2^{\ell_k - 2}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(6) \right)$$

FreshKey e)4). Now, we proceed with case e)4) in predicate FreshKey.

We show essentially that the INT-CTXT security property of the AE scheme forbids the adversary from making π accept. That is, since the adversary must alter one of the messages received by π', origin instance of π, then in return the last AE ciphertext computed by π' (based on data chosen by the adversary) cannot be valid for π.

Game$^{e)4)}$ 1. In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}_0^{e)4)} \leq \mathsf{adv}_1^{e)4)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}}$$

$Game^{e)4)}$ *2.* In this game, the challenger aborts the experiment if either an ephemeral KEM public key epk or ciphertext ct is not unique among all instances and stages.

There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_1^{e)4)} \leq \mathsf{adv}_2^{e)4)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$

$Game^{e)4)}$ *3.* In this game, the challenger aborts the experiment if a KEM decapsulation failure occurs. That is, the challenger aborts the experiment if, on the one hand, the encapsulation done by an honest instance with input an ephemeral public key epk outputs a ciphertext ct and a shared secret ss, and, on the other hand, upon reception of ct by the honest instance that has sent epk the decapsulation with the corresponding private key esk outputs a value $ss' \neq ss$. We have that

$$\mathsf{adv}_2^{e)4)} \leq \mathsf{adv}_3^{e)4)} + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}$$

$Game^{e)4)}$ *4.* In this game, the challenger aborts the experiment if a collision occurs in $\mathsf{Extract}$ outputs among among all instances and stages. Therefore

$$\mathsf{adv}_3^{e)4)} \leq \mathsf{adv}_4^{e)4)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}}$$

$Game^{e)4)}$ *5.* In this game, the challenger aborts the experiment if a collision occurs in Expand outputs among among all instances and stages. Therefore

$$\mathsf{adv}_4^{e)4)} \leq \mathsf{adv}_5^{e)4)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}}$$

$Game^{e)4)}$ *6.* In this game, the challenger aborts the experiment if the key k_0 used by any instance to key Expand in stage $t' \geq 1$ is equal to the key k_1 or k_2 used to key Expand in stage t', or is equal to the key k_0, k_1 or k_2 used to key Expand in stages u, $t' < u < t$, or is equal to the key k_0 or k_1 used to key Expand in stage t.

Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (when keyed with a pseudorandom key). That is, in the latter case, we will replace Expand with a random function. Therefore, we must be able to distinguish the keys that key Expand when we want to leverage its CR property and the keys that key Expand which will eventually be replaced with a random function.

Let us first consider one instance. If the keys k_1, k_2 in stage t', and k_0, k_1, k_2 in stages u, $t' < u < t$, and k_0 and k_1 in stage t are chosen uniformly at random, then the probability that one of these keys collide with the key k_0 used in stage t' is $1/2^{\ell_k}$. Therefore, the probability that none of these keys (in stages t', and u, $t' < u < t$, and t) collide with k_0 used in stage t' is

$$\left(1 - \frac{1}{2^{\ell_k}}\right)^2 \times \left(1 - \frac{1}{2^{\ell_k}}\right)^{3(n_T-2)} \times \left(1 - \frac{1}{2^{\ell_k}}\right)^2 = \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}$$

because each instance executes at most n_T stages. Let $A_{\tilde{\pi}}$ be such an event for a given instance $\tilde{\pi}$.

Let us now consider the event that $A_{\tilde{\pi}}$ holds for all instances $\tilde{\pi}$: $\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}$. Let us consider the opposite event that, for at least one instance $\tilde{\pi}$, the key k_0 in stage t' be equal to at least another key k_1 or k_2 in stage t', or k_0, k_1 or k_2 in a stage u, $t' < u < t$, or k_0 or k_1 in stage t. The corresponding event is $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}$. The challenger aborts the experiment if the latter event holds.

We have $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}} = \bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}$. Therefore, the corresponding probability is

$$\begin{aligned}
\Pr\left[\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}\right] &= \Pr\left[\bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}\right] \\
&\leq \sum_{\tilde{\pi}} \Pr\left[\overline{A_{\tilde{\pi}}}\right] \\
&= \sum_{\tilde{\pi}} (1 - \Pr[A_{\tilde{\pi}}]) \\
&= \sum_{\tilde{\pi}} \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \\
&= n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right)
\end{aligned}$$

because there are at most $n_P \cdot n_S$ instances.

If $\frac{1}{2^{\ell_k}}$ is close to 0, then $1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2} \simeq \frac{3 \cdot n_T - 2}{2^{\ell_k}}$. Hence

$$n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \simeq \frac{n_P \cdot n_S \cdot (3 \cdot n_T - 2)}{2^{\ell_k}}.$$

Therefore, we have that

$$\mathsf{adv}_5^{e)4)} \leq \mathsf{adv}_6^{e)4)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right)$$

Game$^{e)4)}$ 7. In this game, the challenger aborts the experiment if the parameter *pss* used by any instance in stage t' as input to the function Extract is equal to at least one of the subsequent parameters *pss* used in input to Extract in stages u, $t' < u \leq t$.

This rule follows the same rationale as that of introduced in Game$^{e)4)}$ 6. Indeed, in subsequent games we will rely upon the CR property of the function Extract but also on the PRF security of Extract (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Extract is keyed with

distinct keys. Then, we will be able to safely replace Extract with a random function when keyed with a specific (unique) key, and still rely upon the CR property of Extract when keyed with another specific (unique) key.

Let us consider first one instance. If the parameters pss used in stages u, $t' < u \leq t$, are chosen uniformly at random, then the probability that one of theses values collide with pss used in stage t' is $1/2^{\ell_{pss}}$. Therefore, the probability that none of these values be equal to pss used in stage t' is $\left(1 - 2^{-\ell_{pss}}\right)^{n_T - 1}$, because an instance executes at most n_T stages. Let $A_{\tilde{\pi}}$ be such an event for an instance $\tilde{\pi}$.

Let us now consider the event that $A_{\tilde{\pi}}$ holds for all instances $\tilde{\pi}$: $\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}$. Let us consider the opposite event that, for at least one instance $\tilde{\pi}$, the parameter pss used in stage t' be equal to at least a parameter pss used in a stage u, $t' < u \leq t$. The corresponding event is $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}$. The challenger aborts the experiment if the latter event holds.

We have $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}} = \bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}$. Therefore, the corresponding probability is

$$\begin{aligned}
\Pr\left[\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}\right] &= \Pr\left[\bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}\right] \\
&\leq \sum_{\tilde{\pi}} \Pr\left[\overline{A_{\tilde{\pi}}}\right] \\
&= \sum_{\tilde{\pi}} \left(1 - \Pr[A_{\tilde{\pi}}]\right) \\
&= \sum_{\tilde{\pi}} \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \\
&= n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right)
\end{aligned}$$

because there are at most $n_P \cdot n_S$ instances.

If $\frac{1}{2^{\ell_{pss}}}$ is close to 0, then $1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1} \simeq \frac{n_T - 1}{2^{\ell_{pss}}}$. Hence

$$n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \simeq \frac{n_P \cdot n_S \cdot (n_T - 1)}{2^{\ell_{pss}}}.$$

Therefore, we have that

$$\mathsf{adv}_6^{e)4)} \leq \mathsf{adv}_7^{e)4)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right)$$

Game$^{e)4)}$ *8.* In this game, the challenger aborts the experiment if a collision occurs in outputs of the chosen 2-universal hash function H_{ec}. These outputs are

λ-bit long. Therefore, by definition of H_{ec} (cf. Definition 24), we have

$$\mathsf{adv}_7^{e)4)} \leq \mathsf{adv}_8^{e)4)} + \frac{n_P \cdot n_S \cdot n_T}{2^\lambda}$$

Game$^{e)4)}$ *9.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, the stage t', and the instance π' that is origin instance of π (in stage t) and its parent, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_8^{e)4)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \mathsf{adv}_9^{e)4)}$$

Game$^{e)4)}$ *10.* In this game, we replace in stage t' the shared secret ss computed by π with a value $\tilde{ss}$ of same size uniformly drawn at random. We do the same for π' in stage t'. Recall that in stage t', π and π' match. Hence, they share the same KEM public key epk and ciphertext ct. Moreover, by Game$^{e)4)}$ 3, decapsulation failures are excluded. Consequently, π and π' share the same value ss in stage t'. In addition, no query RevealECK(π, t') nor RevealECK(π', t') is issued. Therefore, detecting the change implies an adversary $\mathcal{B}_1$ able to break the IND-CPA security of the KEM scheme. Here we rely upon the IND-CPA notion because the two instances π and π' match in stage t'. Thus, they share the KEM public key and ciphertext. Therefore, we have

$$\mathsf{adv}_9^{e)4)} \leq \mathsf{adv}_{10}^{e)4)} + \mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}}$$

Game$^{e)4)}$ *11.* In this game, we replace (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot) = \mathsf{Extract}(\cdot, \tilde{ss})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$. We do a similar change at π' (which shares the same cid as well as the same shared secret ss as π).

Since $\tilde{ss}$ is sampled uniformly at random by Game$^{e)4)}$ 10 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_2$ queries its Func oracle to get k_1. However, to that point we have no guarantee that the instances π and π' share the same value pss at the beginning of stage t'. If not, the respective inputs dk_0 to Func are different for π and π' (because, by Games$^{e)4)}$ 4 and 5, no collision occurs in $\mathsf{Extract}$ or Expand during the computation of k_0 and dk_0). Therefore, we have to count for at most 2 queries to Func (i.e. $q_{\text{Fc}} \leq 2$).

Therefore

$$\mathsf{adv}_{10}^{e)4)} \leq \mathsf{adv}_{11}^{e)4)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2)$$

Game$^{e)4)}$ *12.* As explained in Game$^{e)4)}$ 11, to that point, we cannot ensure that the two instances compute the same value dk_0, input to compute k_1. It may happen that using distinct values dk_0 as input to $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$, the outputs be equal. We want to preclude such an event. The probability that such a collision occurs at π is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_1. Therefore, in this game, the

challenger aborts the experiment if such a collision occurs.

Hence

$$\mathsf{adv}_{11}^{e)4)} \leq \mathsf{adv}_{12}^{e)4)} + \frac{1}{2^{\ell_k}}$$

Game$^{e)4)}$ *13.* In this game, the challenger aborts the experiment if the key k_0 computed by π' is equal to the key k_1 computed by π, when the two instances do not use the same value pss at beginning of stage t'. Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Expand is keyed with distinct keys. Then, we will be able to safely replace Expand with a random function when keyed with a specific (unique) key, and still rely upon the CR property of Expand when keyed with another specific (unique) key.

By Game$^{e)4)}$ 11, the ℓ_k-bit long key k_1 computed respectively by π is output by a random function. Therefore, we have that

$$\mathsf{adv}_{12}^{e)4)} \leq \mathsf{adv}_{13}^{e)4)} + \frac{1}{2^{\ell_k}}$$

Game$^{e)4)}$ *14.* In this game, we replace (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this, because k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{e)4)}$ 11). We apply a similar change at π' if the latter shares the same value pss as π.

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand.

During the stage, $\mathcal{B}_3$ queries its oracle Func to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1. Since π and π' share cid, one query to Func is enough to get hk_R, n_R and dk_1 for π and π' (if the latter uses the same value pss as π). Moreover, π and π' match in stage t'. Therefore (if the two instances share pss at beginning of the stage), they use the same inputs to compute the other parameters (each instance can copy the parameter already computed by its partner instead of computing it anew). Consequently, 7 calls to the oracle Func are enough to compute the four parameters (i.e. $q_{\mathrm{Fc}} \leq 7$).

Therefore

$$\mathsf{adv}_{13}^{e)4)} \leq \mathsf{adv}_{14}^{e)4)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)$$

Game$^{e)4)}$ *15.* By Game$^{e)4)}$ 12, if π and π' use distinct inputs dk_0, then they compute distinct values k_1 (i.e. π' does not use the same function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ as π). We want to exclude the event that, while π and π' use distinct functions, they compute the same AE key (hk_I or hk_R) needed to compute and decrypt the first AE ciphertext received by π (c_I or c_R). By Game$^{e)4)}$ 14, the probability that π computes this same AE key as π' in such a case is $1/2^{\ell_{hk}}$, where ℓ_{hk} is the bitlength of hk_I and hk_R.

Therefore, in this game the challenger aborts the experiment if any of these

two such collisions occurs.

Hence

$$\mathsf{adv}_{14}^{e)4)} \leq \mathsf{adv}_{15}^{e)4)} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{e)4)}$ *16.* In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it.

By Game$^{e)4)}$ 14, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_4$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_4$ makes at most 1 call to DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value *pss* (and cid) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_4$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{15}^{e)4)} \leq \mathsf{adv}_{16}^{e)4)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

To that point, in stage t', if π and π' use distinct values *pss*, then they compute distinct values dk_0 (because, by Games$^{e)4)}$ 4 and 5, no collision occurs in $\mathsf{Extract}$ and Expand outputs during the computation of k_0 and dk_0). In turn, this implies that the two instances output distinct values k_1 (by Game$^{e)4)}$ 12) and distinct AE keys to compute and decrypt the first AE ciphertext received by π (by Game$^{e)4)}$ 15). Therefore, if the first AE ciphertext received by π is invalid, then π aborts the run (hence, the AE ciphertext does not appear in the list $\mathsf{m_r}$ of the messages received by π). Let us assume that the ciphertext is valid. The AE ciphertext is verified by π against an AE header (tr_0 or tr_1, depending on whether π is initiator or responder) that is unique (by Game$^{e)4)}$ 1) since it is computed from *epk* and *ct* which are unique also (by Game$^{e)4)}$ 2). Therefore, no honest instance but π' can use such an header to compute the AE ciphertext. But π' uses an AE key distinct from that of used by π to decrypt the received AE ciphertext. Therefore, we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext received by π is a valid forgery. Therefore, the challenger aborts the experiment (by Game$^{e)4)}$ 16).

Since π and π' match in stage t' (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that both instances do share the same value *pss* at beginning of stage t'.

Game$^{e)4)}$ *17.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}(dk_1, \cdot)$ with a random function

$\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$. We can safely apply the change at both instances because (by the preceding games) we a have excluded the possibility for the two instances to use distinct values dk_1.

Since dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{e)4)}$ 14) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of $\mathsf{Extract}$.

During the stage, $\mathcal{B}_5$ makes at most 1 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 1$). Indeed, the two instances π and π' use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and by Game$^{e)4)}$ 8, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{e)4)}$ 14 and 16, and the same transcript tr_2 since, again, they are matching).

Therefore

$$\mathsf{adv}_{16}^{e)4)} \leq \mathsf{adv}_{17}^{e)4)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

Game$^{e)4)}$ 18. In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Expand}(k_2,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{e)4)}$ 17) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_6$ to break the PRF security of Expand.

We can safely apply this change because the two instances π and π' do compute the same value k_2. Indeed, the two instances use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{e)4)}$ 17). Moreover, the two instances use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{e)4)}$ 14 and 16, and the same transcripts since, again, they are matching).

This implies also that, during the stage, $\mathcal{B}_6$ makes at most 3 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$). Therefore

$$\mathsf{adv}_{17}^{e)4)} \leq \mathsf{adv}_{18}^{e)4)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, the two partnered instances π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{e)4)}$ 14). They share also tr_0 and c_R (because the instances match). Therefore, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R and n'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \mathtt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \mathtt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which

is excluded by Game$^{e)4)}$ 8).

This implies that π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and inputs to compute the new value pss, hence they share the same value which is, in addition, indistinguishable from a value uniformly drawn at random (by Game$^{e)4)}$ 18).

Moreover, recall that neither RevealPSS(π, t') nor RevealPSS(π', t') is queried. Therefore, at the end of stage t', pss is indistinguishable from a value chosen uniformly at random and known only to π and π'.

Next, we will rely upon the PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ (keyed with pss). The next six games are repeated for stages $t'+1, \ldots, t-1$ (in order to make this appear, we denote each of these games as "Game$^{e)4)}$ $n+$"). Since the following six games rely upon the PRF security of Expand, Extract and the swap-PRF security of the latter, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake. We recall also that the instances π and π' are matching in stages u (hence, the changes described below apply to π' as well as π) and neither RevealPSS(π, u) nor RevealPSS(π', u) is queried for stages $u \in \{t'+1, \ldots, t-1\}$.

Game$^{e)4)}$ 19+. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{e)4)}$ 20+. PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{e)4)}$ 21+. PRF security of Extract keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{e)4)}$ 22+. PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{e)4)}$ 23+. PRF security of Extract keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{e)4)}$ 24+. PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{e)4)}$ 8 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \texttt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{e)4)}$ 8.

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$, shared by π and π' (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$) and unknown to the adversary. Taking into account that there are at most n_T stages per instance,

we have:

$$\mathsf{adv}_{18}^{e)4)} \leq \mathsf{adv}_{24+}^{e)4)} + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \Big)$$

Now, we consider stage t.

Game$^{e)4)}$ *25.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(pss, \cdot) = \mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$. We do the same at π' since it shares pss with π.

At beginning of stage t, pss is a value indistinguishable from a value of same size uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect a change is upper-bounded by the advantage of an adversary $\mathcal{B}_7$ to break the swap-PRF security of $\mathsf{Extract}$. During stage t, $\mathcal{B}_7$ queries its FUNC oracle at most once in order to compute k_0 (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{24}^{e)4)} \leq \mathsf{adv}_{25}^{e)4)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1)$$

Game$^{e)4)}$ *26.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$. By Game$^{e)4)}$ 25, k_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_8$ to break the PRF security of Expand. We do a similar change at π' (since it shares pss with π).

During stage t, $\mathcal{B}_8$ queries its FUNC oracle at most twice in order to compute hk_R^τ and dk_0 (i.e. $q_{\mathrm{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{25}^{e)4)} \leq \mathsf{adv}_{26}^{e)4)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2)$$

Game$^{e)4)}$ *27.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$. By Game$^{e)4)}$ 26, dk_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_9$ to break the PRF security of $\mathsf{Extract}$. We do a similar change at π' (since it shares pss with π).

We cannot ensure that π and π' share the same KEM ephemeral parameters (cid and the shared secret ss). Therefore, $\mathcal{B}_9$ may have to query its FUNC oracle at most twice in order to compute k_1 (i.e. $q_{\mathrm{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{26}^{e)4)} \leq \mathsf{adv}_{27}^{e)4)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2)$$

To that point, in stage t, π and π' use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (since they share pss at beginning of stage t) unknown to the adversary.

Game$^{e)4)}$ *28.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We do the same at instance π' if the latter shares cid with π in stage t (then, k_1 is also shared by π and π' because *pss* is shared by the two instances at beginning of stage t).

By Game$^{e)4)}$ 27, k_1 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_{10}$ to break the PRF security of Expand. We do a similar change at π' (if it shares cid with π).

During the stage, $\mathcal{B}_{10}$ queries its oracle FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1. If π and π' share cid in stage t, then one query to FUNC is enough to get hk_R, n_R and dk_1 for π and π'. But, then we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I, hk'_R and n'_R. Therefore, it may happen that $\mathcal{B}_{10}$ queries its oracle twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Now, we have also to take into account the possibility that π and π' do not use the same inputs to the random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (cf. Game$^{e)4)}$ 27) and yet output the same value k_1. If π and π' do not share, in stage t, the same parameters *epk*, *ct*, then FUNC may have to be queried once per instance to compute hk_R and n_R. That is, $q_{\mathrm{FC}} \leq 1 + 2 \times 6$.

Therefore

$$\mathsf{adv}_{27}^{e)4)} \leq \mathsf{adv}_{28}^{e)4)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(13)$$

Game$^{e)4)}$ *29.* In this game, the challenger aborts the experiment if the instance π receives a valid last AE ciphertext (c_I or c'_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{e)4)}$ 28, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk'_R) used to decrypt the ciphertext (c_I or c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_{11}$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_{11}$ makes at most 1 call to its oracle DEC in order to decrypt c_I or c'_R (i.e. $q_{\mathrm{D}} \leq 1$). Since π has an origin instance, the received AE ciphertext must come from that (honest) instance. Therefore, $\mathcal{B}_{11}$ may have to compute such a message for that honest instance if the latter uses the same AE key as π. In such a case, the oracle ENC is queried at most 1 time (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{28}^{e)4)} \leq \mathsf{adv}_{29}^{e)4)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1, 1)$$

Game$^{e)4)}$ *30.* In this game, the challenger aborts the experiment if the instance π', when π behaves as a responder's instance and shares the same cid as π', receives a valid final AE ciphertext (c'_R) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π' to decrypt it.

By Game$^{e)4)}$ 28, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$

(since π and π' use the same value pss at beginning of stage t, as well as cid by assumption, they share k_1, hence, the random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$). Therefore, the AE key (hk'_R) used to decrypt the ciphertext (c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_{12}$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_{12}$ makes at most 1 call to DEC in order to decrypt c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same values pss and cid as π', then both instances use the same random function to compute the AE key. If this instance uses the same inputs to the random function as π', then the two instances compute the same AE key. In such a case, $\mathcal{B}_{12}$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$). Note that this message may come from the adversary (and not from an honest instance). However, even if π' does not receive the honest message, then $\mathcal{B}_{12}$ may have to compute such a message for an honest "sending" instance (which would then be "replaced" by the adversary with a message of its choice).

Consequently, we have

$$\mathsf{adv}_{29}^{e)4)} \leq \mathsf{adv}_{30}^{e)4)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Let us consider first that the tested instance π is initiator. The adversary must compute either m_1 or m_3.

- If the adversary computes $m_1 = epk$ different from that of computed by π, then π and π' compute distinct values tr_0, tr_1, and eventually distinct values tr_2 (by Game$^{e)4)}$ 1). The last AE ciphertext c'_R received by π must be computed by π' (because π' is an origin instance of π). If c'_R is not valid for π, then π does not accept. Let us assume that c'_R is valid for π. π verifies the received AE ciphertext with an AE header (tr_2) distinct from that of used by π' to compute the AE ciphertext. Moreover, tr_2 is computed from ct which is unique (by Game$^{e)4)}$ 2). Therefore, tr_2 is unique (by Game$^{e)4)}$ 1). Thus, no honest instance has ever used tr_2 as AE header to compute an AE ciphertext. Therefore, we can leverage the INT-CTXT security of the AE function to claim that c'_R is a valid forgery. Consequently, the challenger aborts the experiment (by Game$^{e)4)}$ 29).
- Let us consider that the adversary computes m_3 (and does not alter m_1 sent by π). By Game$^{e)4)}$ 16, π and its origin instance π' share the same value pss at beginning of stage t. Consequently, they share the same AE parameters (hk_I, n_I, tr_1) used to compute and decrypt m_3 (because m_1 and m_2 are also shared by the two instances). Let $\tilde{c_I} \neq c_I$ be the AE ciphertext computed by the adversary (and sent to π'). If $\tilde{c_I}$ is not valid for π', then π' does not accept and π does not either (since π' does not send m_4 and the latter must be computed by π'). If $\tilde{c_I}$ is valid for π', then let $c_I = \mathsf{AE.Enc}(hk_I, \ell_{c_I}, n_I, tr_1, x)$ be the AE ciphertext computed by π, with $\ell_{c_I} = |c_I|$ and $x = Z\|T\|\Sigma\|V\|S\|cert_I\|\sigma_I$.
  If $|\tilde{c_I}| \neq |c_I|$, then π' computes a value tr_2 distinct from that of computed by

π (by Game$^{e)4)}$ 1).
Let us assume that $|\tilde{c_I}|= |c_I|$. If π' computes $x = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I})$, then $c_I = \mathsf{AE.Enc}(hk_I, |c_I|, n_I, tr_1, x) = \mathsf{AE.Enc}(hk_I, |\tilde{c_I}|, n_I, tr_1, x) = \tilde{c_I}$ (because we assume AE is tidy). But $\tilde{c_I} \neq c_I$. Therefore, π' computes $x' = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I}) \neq x$. Consequently, π and π' compute distinct values tr_2 (by Game$^{e)4)}$ 1).
By Game$^{e)4)}$ 2, epk is unique. Therefore, tr_2 (computed by π) is also unique (by Game$^{e)4)}$ 1). Moreover, tr_2 is not shared by π'. Consequently, there exists no honest instance that ever uses tr_2 as AE header to compute an AE ciphertext. If c'_R (computed by π' and decrypted by π with tr_2 as AE header) is not valid for π, then π does not accept. If c'_R is valid for π, then we can leverage the INT-CTXT security of the AE function to claim that c'_R is a valid AE forgery. Then, the challenger aborts the experiment (by Game$^{e)4)}$ 29).

Now, let us consider that the tested instance π is responder. The adversary must compute either m_2 or m_4.

- Let us consider that the adversary computes m_2. That is, it computes either the AE ciphertext or the KEM ciphertext or the MAC tag in m_2.
  - Let us assume first that the adversary computes the AE ciphertext. π and its origin instance π' share the same AE parameters (hk_R, n_R, tr_0) used to compute and decrypt m_2 (because m_1 is also shared by the two instances). Let $\tilde{c_R}$ be the AE ciphertext computed by the adversary (and sent to π'). If $\tilde{c_R}$ is not valid for π', then the latter does not accept and π does not either (since π' does not send m_3 and the latter must be computed by π'). If $\tilde{c_R}$ is valid for π', then let $c_R = \mathsf{AE.Enc}(hk_R, \ell_{c_R}, n_R, tr_0, x)$ be the AE ciphertext computed by π, with $\ell_{c_R} = |c_R|$ and $x = \Phi_R \| \Omega$.
    If $|\tilde{c_R}| \neq |c_R|$, then π' computes a value tr_1 distinct from that of computed by π (by Game$^{e)4)}$ 1).
    Let us assume that $|\tilde{c_R}| = |c_R|$. If π' computes $x = \mathsf{AE.Dec}(hk_R, n_R, tr_0, \tilde{c_R})$, then $c_R = \mathsf{AE.Enc}(hk_R, |c_R|, n_R, tr_0, x) = \mathsf{AE.Enc}(hk_R, |\tilde{c_R}|, n_R, tr_0, x) = \tilde{c_R}$ (because we assume AE is tidy). But $\tilde{c_R} \neq c_R$. Therefore π' computes $x' = \mathsf{AE.Dec}(hk_R, n_R, tr_0, \tilde{c_R}) \neq x$. Consequently, π and π' compute distinct values tr_1 (by Game$^{e)4)}$ 1).
    Therefore, π' computes the last AE ciphertext c_I received by π based on a value tr_1 different from that of used by π to verify the ciphertext. If c_I is not valid for π, then π does not accept. Let us assume that c_I is valid for π. As said, π' and π do not compute the same value tr_1 (used as AE header). Moreover, tr_1 (computed by π) is unique (by Game$^{e)4)}$ 1) because it is computed from epk which is also unique (by Game$^{e)4)}$ 2). Therefore, no honest instance ever uses tr_1 as AE header to compute an AE ciphertext. Consequently, we can leverage the INT-CTXT security of the AE function to claim that c_I is a valid forgery. In such a case, the challenger aborts the experiment (by Game$^{e)4)}$ 29).

- Let us assume that the adversary computes the KEM ciphertext. Then (assuming there is no KEM decapsulation error), π and π' compute distinct tr_1 values (by Game$^{e)4)}$ 1). We can reiterate the reasoning above and conclude as follows: either c_I is not valid for π and π does not accept, or c_I is a valid forgery and the challenger aborts the experiment (by Game$^{e)4)}$ 29).

- Let us assume that the adversary computes the MAC tag. Since π and π' share pss at beginning of stage t, as well as m_1, they compute the same MAC key hk_R^{τ}. Moreover, ct is shared by π and π'. Therefore, π and π' use the same inputs to compute and verify the MAC tag. If the adversary computes $\tilde{\tau_R}$, then $\tilde{\tau_R} \neq \tau_R$ (where τ_R is computed by π). Therefore, necessarily $\tilde{\tau_R}$ is invalid for π' (because we assume the MAC function $\mathsf{MAC.Tag}$ is deterministic). Hence π' does not accept and π does not either (since it does not receive m_3 which must be computed by π').

– Let us consider that the adversary computes m_4. Let $\tilde{c'_R} \neq c'_R$ be the value computed by the adversary, where c'_R is the value computed by π.

- If $\tilde{c'_R}$ is not valid for π', then $\tilde{c'_R}$ does not belong to $\pi'.\mathsf{m_r}[t]$. Therefore, π is an origin instance of π' (because m_2 is received by π' unmodified by the adversary). This contradicts the assumption of subcase e)4).

- Let us assume that $\tilde{c'_R}$ is valid for π'. π' uses the same parameters hk'_R, n'_R as π because both instances share the same value pss at beginning of stage t as well as the messages m_1, m_2, m_3, and the KEM shared secret ss (by Game$^{e)4)}$ 3). Hence, π and π' share also the same value tr_2. Moreover, tr_2 is computed from epk which is unique (by Game$^{e)4)}$ 2). Therefore, tr_2 is also unique (by Game$^{e)4)}$ 1). This means that no honest instance (besides π) ever computes an AE ciphertext with tr_2 as AE header. Consequently, we can leverage the INT-CTXT security of the AE function to claim that $\tilde{c'_R}$ is a valid AE forgery. Then, the challenger aborts the experiment (by Game$^{e)4)}$ 30).

To sum up, this scenario (the adversary computes m_4) cannot be considered because it contradicts the assumption of subcase e)4).

Finally, π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$ which does not help the adversary guess the challenge bit b_{key}. Therefore, the adversary can do no better than guess that bit, and $\mathsf{adv}_{30}^{e)4)} = 0$.

Collecting all the probabilities from Game$^{e)4)}$ 0 to Game$^{e)4)}$ 30 we have:

$$\begin{aligned}
\mathsf{adv}_0^{e)4)} \leq\ & n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \\
& + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \frac{1}{2^{\ell_{hk}-1}} + \frac{1}{2^{\ell_k-1}}\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) \\
& \quad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13) + 5 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) + (2 \cdot n_T - 3) \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) \\
& \quad + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) \\
& \quad\quad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\Big)
\end{aligned}$$

FreshKey e)5). Now, we proceed with case e)5) in predicate FreshKey.

Our strategy is the same as that of case e)4) in predicate FreshKey. That is, we rely upon the INT-CTXT security property of the AE scheme to forbid the adversary from making π accept (and π and its origin instance π' cannot share the same plaintext because π is not an origin instance of π').

The games for stage t' rely essentially on the security provided by the quantum key exchange. The games for stages $t'+1, \ldots, t$ are the same as in the previous case e)4).

Game$^{e)5)}$ *1.* Collision resistance of hash function H.

Game$^{e)5)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

Game$^{e)5)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Here, we do not rely upon the (IND-CPA or IND-CCA) security of the KEM scheme. However, its correctness is required as two instances sharing the same cid must compute the same KEM shared secret *ss* (the latter is involved in the computation of the intermediary secret k_1 which the new *pss* value eventually depends on).

Game$^{e)5)}$ *4.* Collision resistance of Extract.

Game$^{e)5)}$ *5.* Collision resistance of Expand.

Game$^{e)5)}$ *6.* Collision probability for the keys k_0, k_1, k_2 when pseudo-random. That is, the challenger aborts the experiment if a parameter k_0 or k_1 used in stage t' is equal to one of the following parameters: k_2 in stage t', or k_0, k_1, k_2 in a stage u, $t' < u < t$, or k_0, k_1 in stage t. Following a reasoning similar to that of subcase FreshAuth b)2), we have:

$$\mathsf{adv}_5^{e)5)} \leq \mathsf{adv}_6^{e)5)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)}\right)$$

Game$^{e)5)}$ *7.* Collision probability for the values *pss* when pseudo-random.

Game$^{e)5)}$ *8.* Collision resistance of H_{ec}.

To that point, we have:

$$\begin{aligned}\mathsf{adv}_0^{e)5)} \leq\ & \mathsf{adv}_8^{e)5)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} \\ & + n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T-1}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right)\end{aligned}$$

Game$^{e)5)}$ *9.* In this game, the challenger aborts the experiment if an honest instance receives an AE ciphertext valid under a key distinct from the key used by the sender to compute the ciphertext. This reduces straightforwardly to the key-committing security of the AE scheme. Therefore

$$\mathsf{adv}_8^{e)5)} \leq \mathsf{adv}_9^{e)5)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{key\text{-}cmt}}$$

Game$^{e)5)}$ *10.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, the putative party and partner instance π', and the stage t' where no query RevealEQK(π, t') nor RevealEQK(π', t'), where $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi'.\mathsf{eqk}[t'] \neq \perp$, is issued, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_9^{e)5)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \mathsf{adv}_{10}^{e)5)}$$

To that point, in stage t', if π and π' use distinct values *pss*, then they compute distinct values dk_0 (because no collision occurs in Extract and Expand outputs during the computation of k_0 and dk_0, by Games$^{e)5)}$ 4 and 5. In turn, this implies that the two instances output distinct values k_1 and distinct AE keys to compute and decrypt the first AE ciphertext received by π. Therefore, either the first AE ciphertext received by π is invalid and rejected by π (hence, it does not appear in the list $\mathsf{m_r}$ of the messages received by π), or the ciphertext is valid but the challenger aborts the experiment by Game$^{e)5)}$ 9. Since π and π'

match in stage t' (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that both instances do share the same value pss at the beginning of stage t'.

Moreover, since π and π' match in stage t', this implies that epk and ct are shared by π' and π. By unicity of epk and ct, π' is the unique origin instance of π. This implies in addition, that the plaintext data corresponding to c_R and c_I are also shared by the two instances (i.e. the (classical) data related to the quantum key agreement are exchanged through an authenticated channel). Moreover, by assumption, we have that $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi'.\mathsf{eqk}[t'] \neq \perp$ (i.e. $flag_{\mathsf{pe}} = \mathtt{true}$ in stage t'). Therefore, the two instances do compute an actual ephemeral quantum secret. Consequently, we can apply Theorem 1 and safely replace $\pi.\mathsf{eqk}[t']$ and $\pi'.\mathsf{eqk}[t']$ with a value of same size uniformly drawn at random.

Game$^{e)5)}$ *11.* In this game, we replace the ephemeral quantum secret computed in stage t' by the targeted instance π and its partner instance π' with a value $\tilde{K}$ of same size uniformly drawn at random.

Tomamichel and Leverrier [109] prove the security of the BB84 protocol based on the following experiment. On the one hand (real context), the real BB84 protocol is executed. On the other hand (ideal context), a BB84 run is executed and the output key is replaced with a value of same length uniformly drawn at random. The maximum probability of distinguishing the context is $p_\Delta = (1 + \epsilon_\Delta)/2$ (see Section 4.7). By Game$^{e)5)}$ 8, collisions in H_{ec} are excluded. Therefore, we can remove the corresponding probability (correctness component) $\epsilon_{ec} = 2^{-\lambda}$ from ϵ_Δ and consider instead the secrecy component only $\epsilon_s = \epsilon_\Delta - \epsilon_{ec}$, as well as the probability $p_s = (1 + \epsilon_s)/2$.

Let F be the event that the adversary detects the change made in this game. We have

$$\begin{aligned}
\mathsf{adv}_{10}^{e)5)} &\leq \mathsf{adv}_{11}^{e)5)} + \Pr[\mathcal{A} \text{ wins} \mid \text{real QKD context}] \\
&\quad - \Pr[\mathcal{A} \text{ wins} \mid \text{ideal QKD context}] \\
&\leq \mathsf{adv}_{11}^{e)5)} + \Pr[F] \\
&\leq \mathsf{adv}_{11}^{e)5)} + 2 \cdot p_s - 1 \\
&= \mathsf{adv}_{11}^{e)5)} + \epsilon_s
\end{aligned}$$

To that point, in stage t', π and π' use $\tilde{K}$ to key $\mathsf{Extract}^{\mathsf{swap}}$ in order to compute k_2.

Game$^{e)5)}$ *12.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{K}, \cdot) = \mathsf{Extract}(\cdot, \tilde{K})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$.

Since $\tilde{K}$ is uniformly drawn at random by Game$^{e)5)}$ 11 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_0$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_0$ makes at most 1 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{11}^{e)5)} \leq \mathsf{adv}_{12}^{e)5)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1)$$

Game$^{e)5)}$ *13.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$ (by Game$^{e)5)}$ 12) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_1$ makes at most 3 calls to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$).

Therefore

$$\mathsf{adv}_{12}^{e)5)} \leq \mathsf{adv}_{13}^{e)5)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3)$$

To that point, at the end of stage t', pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ known only to π and π'. Moreover, recall that neither REVEALPSS(π, t') nor REVEALPSS(π', t') is queried. Therefore, pss is indistinguishable from a random value uniformly chosen at random.

Next, we will rely upon the PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ (keyed with pss). The next six games are repeated for stages $t'+1, \ldots, t-1$ (in order to make this appear, we denote each of these games as "Game$^{e)5)}$ $n+$"). Since the following six games rely upon the PRF security of Expand, $\mathsf{Extract}$ and the swap-PRF security of the latter, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake. We recall also that the instances π and π' are matching in stages u (hence, the changes described below apply to π' as well as π) and neither REVEALPSS(π, u) nor REVEALPSS(π', u) is queried for stages $u \in \{t'+1, \ldots, t-1\}$.

Game$^{e)5)}$ *14+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{e)5)}$ *15+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{e)5)}$ *16+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{e)5)}$ *17+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{e)5)}$ *18+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{e)5)}$ *19+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{e)5)}$ 8 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \mathtt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{e)5)}$ 8.

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$, shared by π and π' (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$) and unknown to the adversary. Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{13}^{e)5)} \leq\ & \mathsf{adv}_{19+}^{e)5)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\end{aligned}$$

Now, we consider stage t.

Game$^{e)5)}$ *20.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot) = \mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$. We do the same at π' since it shares pss with π.

At beginning of stage t, pss is a value indistinguishable from a value of same size uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect a change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the swap-PRF security of $\mathsf{Extract}$. During stage t, $\mathcal{B}_2$ queries its FUNC oracle at most once in order to compute k_0 (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{19}^{e)5)} \leq \mathsf{adv}_{20}^{e)5)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1)$$

Game$^{e)5)}$ *21.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_0,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$. By Game$^{e)5)}$ 20, k_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand. We do a similar change at π' (since it shares pss with π).

During stage t, $\mathcal{B}_3$ queries its FUNC oracle at most twice in order to compute hk_R^τ and dk_0 (i.e. $q_{\mathrm{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{20}^{e)5)} \leq \mathsf{adv}_{21}^{e)5)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2)$$

Game$^{e)5)}$ *22.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}(dk_0,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$. By Game$^{e)5)}$ 21, dk_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_4$ to

break the PRF security of $\mathsf{Extract}$. We do a similar change at π' (since it shares pss with π).

We cannot ensure that π and π' share the same KEM ephemeral parameters (cid and the shared secret ss). Therefore, $\mathcal{B}_4$ may have to query its FUNC oracle at most twice in order to compute k_1 (i.e. $q_{\mathrm{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{21}^{e)5)} \leq \mathsf{adv}_{22}^{e)5)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2)$$

To that point, in stage t, π and π' use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (since they share pss at beginning of stage t) unknown to the adversary.

Game$^{e)5)}$ 23. In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_1,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. By Game$^{e)5)}$ 22, k_1 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of Expand. We do a similar change at π' (if it shares cid with π).

During the stage, $\mathcal{B}_5$ queries its oracle FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1. If π and π' share cid in stage t, then one query to FUNC is enough to get hk_R, n_R and dk_1 for π and π'. But, then we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I, hk'_R and n'_R. Therefore, it may happen that $\mathcal{B}_5$ queries its oracle twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Now, we have also to take into account the possibility that π and π' do not use the same inputs to the random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (cf. Game$^{e)5)}$ 22) and yet output the same value k_1. If π and π' do not share, in stage t, the same parameters epk, ct, then FUNC may have to be queried once per instance to compute hk_R and n_R. That is, $q_{\mathrm{FC}} \leq 1 + 2 \times 6$.

Therefore

$$\mathsf{adv}_{22}^{e)5)} \leq \mathsf{adv}_{23}^{e)5)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13)$$

Game$^{e)5)}$ 24. In this game, the challenger aborts the experiment if the instance π receives a valid last AE ciphertext (c_I or c'_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{e)5)}$ 23, $\mathsf{Expand}(k_1,\cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk'_R) used to decrypt the ciphertext (c_I or c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_6$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_6$ makes at most 1 call to its oracle DEC in order to decrypt c_I or c'_R (i.e. $q_{\mathrm{D}} \leq 1$). Since π has an origin instance, the received AE ciphertext must come from that (honest) instance. Therefore, $\mathcal{B}_6$ may have to compute such a message for that honest instance if the latter uses the same AE key as π. In such a case, the oracle ENC is queried at most 1 time (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{23}^{e)5)} \leq \mathsf{adv}_{24}^{e)5)} + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1)$$

Game$^{e)5)}$ *25.* In this game, the challenger aborts the experiment if the instance π', when π behaves as a responder's instance and shares the same cid as π', receives a valid final AE ciphertext (c'_R) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π' to decrypt it.

By Game$^{e)5)}$ 23, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (since π and π' use the same value pss at beginning of stage t, as well as cid by assumption, they share k_1, hence, the random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$). Therefore, the AE key (hk'_R) used to decrypt the ciphertext (c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_{12}$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_{12}$ makes at most 1 call to Dec in order to decrypt c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same values pss and cid as π', then both instances use the same random function to compute the AE key. If this instance uses the same inputs to the random function as π', then the two instances compute the same AE key. In such a case, $\mathcal{B}_{12}$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$). Note that this message may come from the adversary (and not from an honest instance). However, even if π' does not receive the honest message, then $\mathcal{B}_{12}$ may have to compute such a message for an honest "sending" instance (which would then be "replaced" by the adversary with a message of its choice).

Consequently, we have

$$\mathsf{adv}_{24}^{e)5)} \leq \mathsf{adv}_{25}^{e)5)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Let us consider first that the tested instance π is initiator. The adversary must compute either m_1 or m_3.

- If the adversary computes $m_1 = epk$ different from that of computed by π, then π and π' compute distinct values tr_0, tr_1, and eventually distinct values tr_2 (by Game$^{e)5)}$ 1). The last AE ciphertext c'_R received by π must be computed by π' (because π' is an origin instance of π). If c'_R is not valid for π, then π does not accept. Let us assume that c'_R is valid for π. π verifies the received AE ciphertext with an AE header (tr_2) distinct from that of used by π' to compute the AE ciphertext. Moreover, tr_2 is computed from ct which is unique (by Game$^{e)5)}$ 2). Therefore, tr_2 is unique (by Game$^{e)5)}$ 1). Thus, no honest instance has ever used tr_2 as AE header to compute an AE ciphertext. Therefore, we can leverage the INT-CTXT security of the AE function to claim that c'_R is a valid forgery. Consequently, the challenger aborts the experiment (by Game$^{e)5)}$ 24).
- Let us consider that the adversary computes m_3 (and does not alter m_1 sent by π). By Game$^{e)5)}$ 9, π and its origin instance π' share the same value pss at beginning of stage t. Consequently, they share the same AE parameters (hk_I, n_I, tr_1) used to compute and decrypt m_3 (because m_1 and m_2 are also shared by the two instances). Let $\tilde{c_I} \neq c_I$ be the AE ciphertext computed by the adversary (and sent to π'). If $\tilde{c_I}$ is not valid for

π', then π' does not accept and π does not either (since π' does not send m_4 and the latter must be computed by π'). If $\tilde{c_I}$ is valid for π', then let $c_I = \mathsf{AE.Enc}(hk_I, \ell_{c_I}, n_I, tr_1, x)$ be the AE ciphertext computed by π, with $\ell_{c_I} = |c_I|$ and $x = Z\|T\|\Sigma\|V\|S\|cert_I\|\sigma_I$.
If $|\tilde{c_I}| \neq |c_I|$, then π' computes a value tr_2 distinct from that of computed by π (by Game$^{e)5)}$ 1).
Let us assume $|\tilde{c_I}| = |c_I|$. If π' computes $x = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I})$, then $c_I = \mathsf{AE.Enc}(hk_I, |c_I|, n_I, tr_1, x) = \mathsf{AE.Enc}(hk_I, |\tilde{c_I}|, n_I, tr_1, x) = \tilde{c_I}$ (because we assume AE is tidy). But $\tilde{c_I} \neq c_I$. Therefore, π' computes $x' = \mathsf{AE.Dec}(hk_I, n_I, tr_1, \tilde{c_I}) \neq x$. Consequently, π and π' compute distinct values tr_2 (by Game$^{e)5)}$ 1).
By Game$^{e)5)}$ 2, epk is unique. Therefore, tr_2 (computed by π) is also unique (by Game$^{e)5)}$ 1). Moreover, tr_2 is not shared by π'. Consequently, there exists no honest instance that ever uses tr_2 as AE header to compute an AE ciphertext. If c'_R (computed by π' and decrypted by π with tr_2 as AE header) is not valid for π, then π does not accept. If c'_R is valid for π, then we can leverage the INT-CTXT security of the AE function to claim that c'_R is a valid AE forgery. Then, the challenger aborts the experiment (by Game$^{e)5)}$ 24).

Now, let us consider that the tested instance π is responder. The adversary must compute either m_2 or m_4.

- Let us consider that the adversary computes m_2. That is, it computes either the AE ciphertext or the KEM ciphertext or the MAC tag in m_2.
  - Let us assume first that the adversary computes the AE ciphertext. π and its origin instance π' share the same AE parameters (hk_R, n_R, tr_0) used to compute and decrypt m_2 (because m_1 is also shared by the two instances). Let $\tilde{c_R}$ be the AE ciphertext computed by the adversary (and sent to π'). If $\tilde{c_R}$ is not valid for π', then the latter does not accept and π does not either (since π' does not send m_3 and the latter must be computed by π'). If $\tilde{c_R}$ is valid for π', then let $c_R = \mathsf{AE.Enc}(hk_R, \ell_{c_R}, n_R, tr_0, x)$ be the AE ciphertext computed by π, with $\ell_{c_R} = |c_R|$ and $x = \Phi_R\|\Omega$.
    If $|\tilde{c_R}| \neq |c_R|$, then π' computes a value tr_1 distinct from that of computed by π (by Game$^{e)5)}$ 1).
    Let us assume that $|\tilde{c_R}| = |c_R|$. If π' computes $x = \mathsf{AE.Dec}(hk_R, n_R, tr_0, \tilde{c_R})$, then $c_R = \mathsf{AE.Enc}(hk_R, |c_R|, n_R, tr_0, x) = \mathsf{AE.Enc}(hk_R, |\tilde{c_R}|, n_R, tr_0, x) = \tilde{c_R}$ (because we assume AE is tidy). But $\tilde{c_R} \neq c_R$. Therefore π' computes $x' = \mathsf{AE.Dec}(hk_R, n_R, tr_0, \tilde{c_R}) \neq x$. Consequently, π and π' compute distinct values tr_1 (by Game$^{e)5)}$ 1).
    Therefore, π' computes the last AE ciphertext c_I received by π based on a value tr_1 different from that of used by π to verify the ciphertext. If c_I is not valid for π, then π does not accept. Let us assume that c_I is valid for π. As said, π' and π do not compute the same value tr_1 (used as AE header). Moreover, tr_1 (computed by π) is unique (by Game$^{e)5)}$ 1) because it is computed from epk which is also unique (by Game$^{e)5)}$ 2).

Therefore, no honest instance ever uses tr_1 as AE header to compute an AE ciphertext. Consequently, we can leverage the INT-CTXT security of the AE function to claim that c_I is a valid forgery. In such a case, the challenger aborts the experiment (by Game$^{e)5)}$ 24).

- Let us assume that the adversary computes the KEM ciphertext. Then (assuming there is no KEM decapsulation error), π and π' compute distinct tr_1 values (by Game$^{e)5)}$ 1). We can reiterate the reasoning above and conclude as follows: either c_I is not valid for π and π does not accept, or c_I is a valid forgery and the challenger aborts the experiment (by Game$^{e)5)}$ 24).
- Let us assume that the adversary computes the MAC tag. Since π and π' share pss at beginning of stage t, as well as m_1, they compute the same MAC key hk_R^τ. Moreover, ct is shared by π and π'. Therefore, π and π' use the same inputs to compute and verify the MAC tag. If the adversary computes $\tilde{\tau_R}$, then $\tilde{\tau_R} \neq \tau_R$ (where τ_R is computed by π). Therefore, necessarily $\tilde{\tau_R}$ is invalid for π' (because we assume the MAC function $\mathsf{MAC.Tag}$ is deterministic). Hence π' does not accept and π does not either (since it does not receive m_3 which must be computed by π').

– Let us consider that the adversary computes m_4. Let $\tilde{c'_R} \neq c'_R$ be the value computed by the adversary, where c'_R is the value computed by π.

- If $\tilde{c'_R}$ is not valid for π', then $\tilde{c'_R}$ does not belong to $\pi'.\mathsf{m_r}[t]$. Therefore, π is an origin instance of π' (because m_2 is received by π' unmodified by the adversary). This contradicts the assumption of subcase e)5).
- Let us assume that $\tilde{c'_R}$ is valid for π'. π' uses the same parameters hk'_R, n'_R as π because both instances share the same value pss at beginning of stage t as well as the messages m_1, m_2, m_3, and the KEM shared secret ss (by Game$^{e)5)}$ 3). Hence, π and π' share also the same value tr_2. Moreover, tr_2 is computed from epk which is unique (by Game$^{e)5)}$ 2). Therefore, tr_2 is also unique (by Game$^{e)5)}$ 1). This means that no honest instance (besides π) ever computes an AE ciphertext with tr_2 as AE header. Consequently, we can leverage the INT-CTXT security of the AE function to claim that $\tilde{c'_R}$ is a valid AE forgery. Then, the challenger aborts the experiment (by Game$^{e)5)}$ 25).

To sum up, this scenario (the adversary computes m_4) cannot be considered because it contradicts the assumption of subcase e)5).

Finally, π never accepts and a query $\textsc{Test}(\pi, t)$ returns $\perp$ which does not help the adversary guess the challenge bit b_{key}. Therefore, the adversary can do no better than guess that bit, and $\mathsf{adv}_{25}^{e)5)} = 0$.

Collecting all the probabilities from Game$^{e)5)}$ 0 to Game$^{e)5)}$ 25 we have:

$$\begin{aligned}
\mathsf{adv}_0^{e)5)} \leq\ & n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T-1}\right) \\
& + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \epsilon_s\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} + \mathsf{adv}^{\mathsf{key\text{-}cmt}}_{\mathsf{AE}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13) + 3 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) \\
& \quad + n_T \cdot \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + (n_T - 2) \cdot \Big(2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big) \\
& \quad + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)\Big)\Big)
\end{aligned}$$

FreshKey e)6). Now, we proceed with case e)6) in predicate FreshKey.

Our strategy is the same as that of case e)4) in predicate FreshKey (we rely upon the INT-CTXT security property of the AE scheme to forbid the adversary from making π accept).

The games for stage 1 rely essentially on the fact that *pss* is initialised with a value uniformly chosen at random. The games for stages $2, \ldots, t$ are the same as in the previous cases e)4) and e)5) for stages $t' + 1, \ldots, t$.

Game$^{e)6)}$ *1.* Collision resistance of hash function H.

Game$^{e)6)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

Game$^{e)6)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Here, we do not rely upon the (IND-CPA or IND-CCA) security of the KEM scheme. However, its correctness is required as two instances sharing the same $\mathsf{cid} = epk \| ct$ must compute the same KEM shared secret *ss* (the latter is involved in the computation of the intermediary secret values and keys).

Game$^{e)6)}$ *4.* Collision resistance of H_{ec}.

To that point, we have

$$\begin{aligned}
\mathsf{adv}_0^{e)6)} \leq\ & \mathsf{adv}_4^{e)6)} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\
& + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right)
\end{aligned}$$

$Game^{e)6)}$ *5.* In this game, the challenger tries to guess the targeted instance π, its parent, the putative partner party and instance π', and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_4^{e)6)} \leq (n_P \cdot n_S)^2 \cdot \mathsf{adv}_5^{e)6)}$$

$Game^{e)6)}$ *6.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(pss, \cdot) = \mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$.

Since pss is initialised with a value uniformly drawn at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_0$ to break the swap-PRF security of $\mathsf{Extract}$. During a given stage, $\mathcal{B}_0$ queries its FUNC oracle in order to compute k_0 (i.e. $q_{\mathrm{FC}} \leq 1$).

We do a similar change at π' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_5^{e)6)} \leq \mathsf{adv}_6^{e)6)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1)$$

$Game^{e)6)}$ *7.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$. We can do this because k_0 computed by π is now uniformly drawn at random by $\mathrm{Game}^{e)6)}$ 6. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the PRF security of Expand. During a given stage, $\mathcal{B}_1$ makes at most two queries to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 2$).

We apply a similar change at π' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_6^{e)6)} \leq \mathsf{adv}_7^{e)6)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2)$$

$Game^{e)6)}$ *8.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$. We can do this because dk_0 computed by π is now uniformly drawn at random by $\mathrm{Game}^{e)6)}$ 7. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the PRF security of $\mathsf{Extract}$. During a given stage, $\mathcal{B}_2$ makes at most one query to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 1$).

We apply a similar change at π' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_7^{e)6)} \leq \mathsf{adv}_8^{e)6)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

$Game^{e)6)}$ *9.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this because k_1 computed by π is now uniformly drawn at random by $\mathrm{Game}^{e)6)}$ 8. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand. During a given stage, $\mathcal{B}_3$ makes at most 7 queries to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 7$).

We apply a similar change at π' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_8^{e)6)} \leq \mathsf{adv}_9^{e)6)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)$$

Game$^{e)6)}$ *10.* If π and π' do not share the same value pss, they compute distinct keys k_0, dk_0, k_1, and hk_R and hk_I unless a collision occurs when π and π' choose each of the latter parameters. We want to exclude such events. Therefore, the challenger aborts the experiment if any such collision occurs.

When π and π' do not share pss, the probability that π computes the same value k_0 (or k_1) as π' is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_0 and k_1. The probability that π computes the same value dk_0 as π' is $1/2^{\ell_{dk}}$ where ℓ_{dk} is the bitlength of dk_0. The probability that π computes the same value hk_R (or hk_I, both values being uniformly drawn at random by Game$^{e)6)}$ 9) as π' is $1/2^{\ell_{hk}}$ where ℓ_{hk} is the bitlength of hk_R and hk_I.

Therefore

$$\mathsf{adv}_9^{e)6)} \leq \mathsf{adv}_{10}^{e)6)} + \frac{1}{2^{\ell_k - 1}} + \frac{1}{2^{\ell_{dk}}} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{e)6)}$ *11.* In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, nonce, header) to compute the ciphertext as π to decrypt it.

By Game$^{e)6)}$ 9, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_4$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_4$ makes at most 1 call to DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_4$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{10}^{e)6)} \leq \mathsf{adv}_{11}^{e)6)} + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1)$$

To that point, if π and π' do not share the same value pss at the beginning of the first stage, then they compute distinct AE keys corresponding to the first AE ciphertext received by π (which is sent by π' since both instances match in stage 1), unless a collision occurs in the computation of one of the intermediary keys or in the computation of the AE keys. Such events are excluded by Game$^{e)6)}$ 10. If the first AE ciphertext received by π is invalid, then π aborts the run (hence, it does not appear in the list $\mathsf{m_r}$ of the messages received by π) and does not accept. Let us assume that the AE ciphertext is valid for π. The ciphertext is decrypted by π with an AE header (tr_0 or tr_1 depending on whether π is initiator or

responder) computed from epk and ct (these values are shared by π and π' since they match in stage 1). epk and ct are unique (by Game$^{e)6)}$ 2). Hence, the AE header used by π is also unique (by Game$^{e)6)}$ 1). Therefore, no honest instance but π' computes an AE ciphertext with that same AE header. Moreover, π' and π use distinct AE keys to compute and decrypt the AE ciphertext respectively. Therefore, no honest instance has ever used the same AE parameters to compute the AE ciphertext as π to decrypt it. Consequently, we can leverage the INT-CTXT security of the AE function to claim that the AE ciphertext received and deemed valid by π is a valid forgery. Then, the challenger aborts the experiment (by Game$^{e)6)}$ 11). Therefore, if the challenger does not abort the experiment, then this implies that both instances do share the same value pss at the beginning of stage 1.

Game$^{e)6)}$ 12. In this game, we replace in stage 1 (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}(dk_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$.

Since dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{e)6)}$ 9) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of $\mathsf{Extract}$.

During the stage, $\mathcal{B}_5$ makes at most 1 call to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 1$). Indeed, the two instances π and π' use the same input W because they share the ciphertext c'_R (since they are matching in stage 1 by assumption) and by Game$^{e)6)}$ 4, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{e)6)}$ 9 and 11, and the same transcript tr_2 since, again, they are matching).

Therefore

$$\mathsf{adv}_{11}^{e)6)} \leq \mathsf{adv}_{12}^{e)6)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

Game$^{e)6)}$ 13. In this game, we replace in stage 1 (at the targeted instance π and its matching instance π') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{e)6)}$ 12) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of Expand.

We can safely apply this change because the two instances π and π' do compute the same value k_2. Indeed, the two instances use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{e)6)}$ 12). Moreover, the two instances use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and by Game$^{e)6)}$ 4, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{e)6)}$ 9 and 11, and the same transcripts since, again, they are matching).

This implies also that, during the stage, $\mathcal{B}_5$ makes at most 3 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$).

Therefore

$$\mathsf{adv}_{12}^{e)6)} \leq \mathsf{adv}_{13}^{e)6)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, the two partnered instances π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{e)6)}$ 9). They share also tr_0 and c_R (because the instances match). Therefore, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \mathtt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \mathtt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{e)6)}$ 4).

This implies that π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ (by Game$^{e)6)}$ 13) known only to them, and the same inputs to compute the new value pss. Hence they share this value. Moreover, neither RevealPSS$(\pi, 0)$ nor RevealPSS$(\pi', 0)$ is queried, and RevealPSS$(\pi, 1)$ and RevealPSS$(\pi', 1)$ are not queried either. Therefore, at the end of stage 1, pss is indistinguishable from a value chosen uniformly at random and known only to π and π'.

Now, we guess the stage t.

Game$^{e)6)}$ 14. In this game, the challenger tries to guess the stage t and aborts if it does not succeed. Therefore

$$\mathsf{adv}^{e)6)}_{13} \leq n_T \cdot \mathsf{adv}^{e)6)}_{14}$$

In the subsequent games, we use the swap-PRF security of $\mathsf{Extract}$ and the PRF security of $\mathsf{Extract}$ and Expand to eventually argue that the value pss at the beginning of stage t is random and known only to π and π'. Since the reasoning relies upon security properties already mentioned, once again we only indicate the properties at stake.

The next six games are repeated for stages u, $1 < u < t$ (in order to make this appear, we denote each of these games as "Game$^{e)6)}$ n+"). The corresponding changes are done at π and its matching instance π' in stages u.

In stages u, $1 < u < t$, π and π' share the inputs to the different random functions (replacing $\mathsf{Extract}$ and Expand) because they match, compute the same intermediary keys (hence use the same random functions), and there is no collision in H_{ec} outputs (by Game$^{e)6)}$ 4). Moreover, recall that RevealPSS(π, u) and RevealPSS(π', u) are not queried where π and π' match in stages u, $1 < u < t$.

Game$^{e)6)}$ 15+. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{e)6)}$ 16+. PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{e)6)}$ *17+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{e)6)}$ *18+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{e)6)}$ *19+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{e)6)}$ *20+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π' (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$).

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{14}^{e)6)} \leq\ & \mathsf{adv}_{20+}^{e)6)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\end{aligned}$$

Now, we consider stage t.

Game$^{e)6)}$ *21.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot) = \mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$. We do the same at π' since it shares pss with π.

At beginning of stage t, pss is a value indistinguishable from a value of same size uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect a change is upper-bounded by the advantage of an adversary $\mathcal{B}_6$ to break the swap-PRF security of $\mathsf{Extract}$. During stage t, $\mathcal{B}_6$ queries its FUNC oracle at most once in order to compute k_0 (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{20+}^{e)6)} \leq \mathsf{adv}_{21}^{e)6)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1)$$

Game$^{e)6)}$ *22.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_0,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$. By Game$^{e)6)}$ 21, k_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_6$ to break the PRF security of Expand. We do a similar change at π' (since it shares pss with π).

During stage t, $\mathcal{B}_6$ queries its FUNC oracle at most twice in order to compute hk_R^τ and dk_0 (i.e. $q_{\mathrm{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{21}^{e)6)} \leq \mathsf{adv}_{22}^{e)6)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2)$$

Game$^{e)6)}$ *23.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$. By Game$^{e)6)}$ 22, dk_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_7$ to break the PRF security of $\mathsf{Extract}$. We do a similar change at π' (since it shares pss with π).

We cannot ensure that π and π' share the same KEM ephemeral parameters (cid and the shared secret ss). Therefore, $\mathcal{B}_7$ may have to query its FUNC oracle at most twice in order to compute k_1 (i.e. $q_{\mathrm{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{22}^{e)6)} \leq \mathsf{adv}_{23}^{e)6)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2)$$

To that point, in stage t, π and π' use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (since they share pss at beginning of stage t) unknown to the adversary.

Game$^{e)6)}$ *24.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. By Game$^{e)6)}$ 23, k_1 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_8$ to break the PRF security of Expand. We do a similar change at π' (if it shares cid with π).

During the stage, $\mathcal{B}_8$ queries its oracle FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1. If π and π' share cid in stage t, then one query to FUNC is enough to get hk_R ,n_R and dk_1 for π and π'. But, then we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I and hk'_R and n'_R. Therefore, it may happen that $\mathcal{B}_8$ queries its oracle twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Now, we have also to take into account the possibility that π and π' do not use the same inputs to the random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (cf. Game$^{e)6)}$ 23) and yet output the same value k_1. If π and π' do not share, in stage t, the same parameters epk, ct, then FUNC may have to be queried once per instance to compute hk_R and n_R. That is, $q_{\mathrm{FC}} \leq 1 + 2 \times 6$.

Therefore

$$\mathsf{adv}_{23}^{e)6)} \leq \mathsf{adv}_{24}^{e)6)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13)$$

Game$^{e)6)}$ *25.* In this game, the challenger aborts the experiment if the instance π receives a valid last AE ciphertext (c_I or c'_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{e)6)}$ 24, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk'_R) used to decrypt the ciphertext (c_I or c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_9$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_9$ makes at most 1 call to its oracle DEC in order to decrypt c_I or c'_R (i.e. $q_{\mathrm{D}} \leq 1$). Since π has an origin instance, the received AE ciphertext must come from that (honest) instance. Therefore, $\mathcal{B}_9$ may have to

compute such a message for that honest instance if the latter uses the same AE key as π. In such a case, the oracle Enc is queried at most 1 time (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{24}^{e)6)} \leq \mathsf{adv}_{25}^{e)6)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Game$^{e)6)}$ *26.* In this game, the challenger aborts the experiment if the instance π', when π behaves as a responder's instance and shares the same cid as π', receives a valid final AE ciphertext (c'_R) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π' to decrypt it.

By Game$^{e)6)}$ 24, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (since π and π' use the same value pss at beginning of stage t, as well as cid by assumption, they share k_1, hence, the random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$). Therefore, the AE key (hk'_R) used to decrypt the ciphertext (c'_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_{10}$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_{10}$ makes at most 1 call to Dec in order to decrypt c'_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same values pss and cid as π', then both instances use the same random function to compute the AE key. If this instance uses the same inputs to the random function as π', then the two instances compute the same AE key. In such a case, $\mathcal{B}_{10}$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$). Note that this message may come from the adversary (and not from an honest instance). However, even if π' does not receive the honest message, then $\mathcal{B}_{10}$ may have to compute such a message for an honest "sending" instance (which would then be "replaced" by the adversary with a message of its choice).

Consequently, we have

$$\mathsf{adv}_{25}^{e)6)} \leq \mathsf{adv}_{26}^{e)6)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

The reasoning is similar as that of the previous case e)4) in predicate $\mathsf{FreshKey}$. Collecting all the probabilities from Game$^{e)6)}$ 0 to Game$^{e)6)}$ 26 we have:

$$\begin{aligned}
\mathsf{adv}_0^{e)6)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\
& + (n_P \cdot n_S)^2 \cdot \left(\frac{1}{2^{\ell_k - 1}} + \frac{1}{2^{\ell_{dk}}} + \frac{1}{2^{\ell_{hk}-1}}\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} \\
& + (n_P \cdot n_S)^2 \cdot \Big(n_T \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13)\Big) \\
& \quad + (3n_T + 2) \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) \\
& \quad + (n_T(n_T-1)+1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2)\Big) \\
& \quad + (2n_T(n_T-2)+2) \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) \\
& \quad + (n_T(n_T-2)+1) \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\Big)
\end{aligned}$$

A.4 Identity Privacy

Again, to prove this property, we consider the different conditions related to the $\mathsf{FreshIdentity}$ predicate, which correspond to as many trivial attacks.

Let π be the instance targeted by the adversary. That is, the real identity of π's parent is hidden to the adversary and chosen among two possible identities selected by the adversary. The adversary must guess the real one out of the two possible.

The most obvious option (according to us) for the adversary to win is to learn the identity carried in the certificate transmitted by π. Yet, other (encrypted or clear) data are also transmitted during a protocol run which may help the adversary. Consequently, we rely upon the length-hiding security of the AE encryption scheme to forbid the adversary from recovering any information that helps it win the experiment. In addition to the AE ciphertexts, an ephemeral KEM public key and ciphertext are exchanged. Therefore, in the proof, we assume that the ephemeral KEM parameters (public key, ciphertext and private key when the adversary is allowed to reveal the latter) do not enable the adversary to identify a party.

Consequently, our strategy is twofold. When the targeted instance interacts with an honest instance (case a)), the strategy is to essentially rely upon the ciphertext indistinguishability security of the AE scheme that encrypts in particular the certificate. In turn, we show that the AE encryption key is indistinguishable from a random value. This key depends on two main secret values: the ephemeral KEM shared secret and the *pss* value. Therefore, as long as one out of these two values remains hidden to the adversary, so does the certificate.

If the adversary interacts with the targeted instance, then it knows the ephemeral KEM shared secret and only *pss* remains secret (case b)). The targeted instance sends its encrypted certificate only upon reception of a first AE

ciphertext that must be forged by the adversary. Therefore, we rely essentially on the INT-CTXT security of the AE functions to ensure that the adversary is not able to forge a valid ciphertext (the AE keys are computed from *pss* among other parameters). As a special case, when π behaves as responder, then the adversary must authenticate first (i.e. it must send a valid signature). Here, we also rely upon the EUF-CMA security of the signature scheme (subcase b)1)).

Therefore, we decompose the proof according to the following cases:

- Case a): the targeted instance interacts with an honest instance (more precisely, the two instance share the ephemeral KEM public key and ciphertext). At least one of the conditions a)1) to a)4) is fulfilled.
  - Subcase a)1): the ephemeral KEM private key is not revealed in stage t.
  - Subcase a)2): *pss* is not revealed where *pss* is "safe" thanks to a previous (uncompromised) ephemeral KEM exchange.
  - Subcase a)3): *pss* is not revealed where *pss* is "safe" thanks to a previous (uncompromised) QKD phase.
  - Subcase a)4): *pss* is not revealed where *pss* is "safe" because it has been initialised with a secret value uniformly drawn at random.
- Case b): the targeted instance does *not* interact with an honest instance. At least one of the conditions b)1) to b)4) is fulfilled.
  - Subcase b)1): the targeted instance is responder and the private signature key of π's putative partner party is not revealed (before π accepts).
  - Subcase b)2): *pss* is not revealed where *pss* is "safe" thanks to a previous (uncompromised) ephemeral KEM exchange.
  - Subcase b)3): *pss* is not revealed where *pss* is "safe" thanks to a previous (uncompromised) QKD phase.
  - Subcase b)4): *pss* is not revealed where *pss* is "safe" because it has been initialised with a secret value uniformly drawn at random.

Let E_i be the event that the adversary wins the identity privacy experiment in Game i, and $\mathsf{adv}_i = \Pr[E_i] - 1/2$.

Game 0. This game corresponds to the identity privacy experiment described by Figure 5 in Section 3. Therefore

$$\Pr[E_0] = \mathsf{adv}_{\Pi}^{\mathsf{id\text{-}priv}}(\mathcal{A}) + \frac{1}{2} = \mathsf{adv}_0 + \frac{1}{2}$$

We split the proof according to the aforementioned subcases. Therefore, we have:

$$\begin{aligned}
\mathsf{adv}_0 &\leq \max\left(\mathsf{adv}_0^{a)}, \mathsf{adv}_0^{b)}\right) \\
&\leq \mathsf{adv}_0^{a)} + \mathsf{adv}_0^{b)} \\
&\leq \mathsf{adv}_0^{a)1)} + \mathsf{adv}_0^{a)2)} + \mathsf{adv}_0^{a)3)} + \mathsf{adv}_0^{a)4)} \\
&\quad + \mathsf{adv}_0^{b)1)} + \mathsf{adv}_0^{b)2)} + \mathsf{adv}_0^{b)3)} + \mathsf{adv}_0^{b)4)}
\end{aligned}$$

FreshIdentity a)1). We start with subcase a)1) in predicate $\mathsf{FreshIdentity}$.

$Game^{a)1)}$ *1.* In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}_0^{a)1)} \leq \mathsf{adv}_1^{a)1)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}}$$

$Game^{a)1)}$ *2.* In this game, the challenger aborts the experiment if either an ephemeral KEM public key epk or ciphertext ct is not unique among all instances and stages. This ensures that the targeted instance shares its cid with at most one instance.

There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_1^{a)1)} \leq \mathsf{adv}_2^{a)1)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$

$Game^{a)1)}$ *3.* In this game, the challenger aborts the experiment if a KEM decapsulation failure occurs. That is, the challenger aborts the experiment if, on the one hand, the encapsulation done by an honest instance with input an ephemeral public key epk outputs a ciphertext ct and a shared secret ss, and, on the other hand, upon reception of ct by the honest instance that has sent epk the decapsulation with the corresponding private key esk outputs a value $ss' \neq ss$. We have that

$$\mathsf{adv}_2^{a)1)} \leq \mathsf{adv}_3^{a)1)} + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}$$

$Game^{a)1)}$ *4.* In this game, the challenger aborts the experiment if a collision occurs in $\mathsf{Extract}$ outputs among among all instances and stages. Therefore

$$\mathsf{adv}_3^{a)1)} \leq \mathsf{adv}_4^{a)1)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}}$$

$Game^{a)1)}$ *5.* In this game, the challenger aborts the experiment if a collision occurs in Expand outputs among among all instances and stages. Therefore

$$\mathsf{adv}_4^{a)1)} \leq \mathsf{adv}_5^{a)1)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}}$$

$Game^{a)1)}$ *6.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, and the instance π' that shares the same cid as π and its parent, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_5^{a)1)} \leq (n_P \cdot n_S)^2 \cdot n_T \cdot \mathsf{adv}_6^{a)1)}$$

Game$^{a)1)}$ *7.* In this game, we replace in stage t the shared secret ss computed by π with a value $\tilde{ss}$ of same size uniformly drawn at random. We do the same for π' in stage t. Recall that in stage t, π and π' share the same cid. Hence, they share the same KEM public key epk and ciphertext ct. Moreover, by Game$^{a)1)}$ 3, decapsulation failures are excluded. Consequently, π and π' share the same value ss in stage t. In addition, no query $\textsc{RevealECK}(\pi, t)$ nor $\textsc{RevealECK}(\pi', t)$ is issued. Therefore, detecting the change implies an adversary $\mathcal{B}_1$ able to break the IND-CPA security of the KEM scheme. Here we rely upon the IND-CPA notion because the two instances π and π' do share the KEM public key and ciphertext. Therefore, we have

$$\mathsf{adv}_6^{a)1)} \leq \mathsf{adv}_7^{a)1)} + \mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}}$$

Game$^{a)1)}$ *8.* In this game, we replace (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot) = \mathsf{Extract}(\cdot, \tilde{ss})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$. We do a similar change at π' (which shares the same cid as well as the same shared secret ss as π).

Since $\tilde{ss}$ is sampled uniformly at random by Game$^{a)1)}$ 7 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_2$ queries its $\textsc{Func}$ oracle to get k_1. However, to that point we have no guarantee that the instances π and π' share the same value pss at the beginning of stage t. If not, the respective inputs dk_0 to $\textsc{Func}$ are different for π and π' (because, by Games$^{a)1)}$ 4 and 5, no collision occurs in $\mathsf{Extract}$ or Expand during the computation of k_0 and dk_0). Therefore, we have to count for at most 2 queries to $\textsc{Func}$ (i.e. $q_{\textsc{Fc}} \leq 2$).

Therefore

$$\mathsf{adv}_7^{a)1)} \leq \mathsf{adv}_8^{a)1)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2)$$

Game$^{a)1)}$ *9.* As explained in Game$^{a)1)}$ 8, to that point, we cannot ensure that the two instances π and π' compute the same value dk_0, input to compute k_1. It may happen that using distinct values dk_0 as input to $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$, the outputs be equal. We want to preclude such an event. The probability that such a collision occurs at π is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_1. Therefore, in this game, the challenger aborts the experiment if such a collision occurs.

Hence,

$$\mathsf{adv}_8^{a)1)} \leq \mathsf{adv}_9^{a)1)} + \frac{1}{2^{\ell_k}}$$

Game$^{a)1)}$ *10.* In this game, the challenger aborts the experiment if one of the two keys k_0 computed by π and π' respectively is equal to the key k_1 computed by π. Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Expand is keyed with distinct keys. Then, we will be able to safely replace Expand with a random function when keyed with a specific (unique) key, and still rely upon the CR

property of Expand when keyed with another specific (unique) key.

By Game$^{a)1)}$ 8, the ℓ_k-bit long key k_1 computed respectively by π is output by a random function. Therefore, we have that

$$\mathsf{adv}_9^{a)1)} \leq \mathsf{adv}_{10}^{a)1)} + \frac{2}{2^{\ell_k}}$$

Game$^{a)1)}$ 11. In this game, we replace (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this, because k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{a)1)}$ 8). We apply a similar change at π' if the latter shares the same value pss as π.

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand.

During the stage, $\mathcal{B}_3$ queries its oracle FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1. Since π and π' share cid, one query to FUNC is enough to get hk_R, n_R and dk_1 for π and π' (if the latter uses the same value pss as π). However, we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I, hk'_R and n'_R. Therefore, it may happen that $\mathcal{B}_3$ queries its oracle twice to get each of these parameters (i.e. $q_{\text{FC}} \leq 3 + 2 \times 4$).

Therefore

$$\mathsf{adv}_{10}^{a)1)} \leq \mathsf{adv}_{11}^{a)1)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(11)$$

Game$^{a)1)}$ 12. By Game$^{a)1)}$ 9, if π and π' use distinct inputs dk_0, then they compute distinct values k_1 (i.e. π' does not use the same function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ as π). We want to exclude the event that, while π and π' use distinct functions, they compute the same AE key (hk_I or hk_R) needed to compute and decrypt the first AE ciphertext received by π (c_I or c_R). By Game$^{a)1)}$ 11, the probability that π computes this same AE key as π' in such a case is $1/2^{\ell_{hk}}$, where ℓ_{hk} is the bitlength of hk_I and hk_R.

Therefore, in this game the challenger aborts the experiment if any of these two such collisions occurs.

Hence

$$\mathsf{adv}_{11}^{a)1)} \leq \mathsf{adv}_{12}^{a)1)} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{a)1)}$ 13. In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{a)1)}$ 11, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_4$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_4$ makes at most 1 call to DEC in order to decrypt c_I

or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss (and cid) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_4$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{12}^{a)1)} \leq \mathsf{adv}_{13}^{a)1)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Let us assume that the first AE ciphertext received by π is computed by π', and π and π' do not share the same value pss at beginning of stage t. Then, they compute distinct parameters k_0 (by Game$^{a)1)}$ 4), dk_0 (by Game$^{a)1)}$ 5), k_1 (by Game$^{a)1)}$ 9) and hk_I or hk_R (by Game$^{a)1)}$ 12) depending on whether π is initiator or responder. The transcripts tr_0 and tr_1 are used as AE header in the computation of the AE ciphertext (c_I or c_R). These values are computed from cid which is shared by π and π' in stage t. $\mathsf{cid} = epk \| ct$ is unique (by Game$^{a)1)}$ 2), hence so are tr_0 and tr_1 (by Game$^{a)1)}$ 1). Therefore, no other honest instance but π' computes an AE ciphertext with an AE header equal to tr_0 or tr_1. In addition, π' uses a distinct AE key to compute the AE ciphertext. Consequently, if the AE ciphertext is not valid, then π aborts the run. Otherwise, we can leverage the INT-CTXT security of the AE function to claim that the (valid) AE ciphertext received by π is a forgery. Then, the challenger aborts the experiment (by Game$^{a)1)}$ 13).

Likewise, let us assume that the first AE ciphertext received by π is not computed by an honest instance. Recall that RevealECK(π, t) and RevealECK(π', t) are not queried, as otherwise π becomes unfresh. Therefore, the random function used by π to compute the AE parameters is known only to π'. Consequently, if the first AE ciphertext received by π is valid, then the challenger aborts the experiment (by Game$^{a)1)}$ 13).

Therefore, in the remaining games, we consider that π and π' do share the same value pss (at beginning of stage t), and the (valid) first AE ciphertext received by π is computed by π'.

Since $\pi.pss[t-1] = \pi'.pss[t-1]$, π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ to compute the AE parameters (by Game$^{a)1)}$ 11). In addition, $\pi.\mathsf{cid}[t] = \pi'.\mathsf{cid}[t]$. Therefore, π and π' compute the same value tr_0 and the same AE parameters hk_R, n_R.

If π is a responder's instance, then the (valid) first AE ciphertext received by π is c_I and is computed by π' (by Game$^{a)1)}$ 13). This implies also that tr_1 (the corresponding AE header) as well as the AE parameters hk_I, n_I are shared by π' and π, as otherwise, c_I is a valid forgery (and the challenger aborts the experiment by Game$^{a)1)}$ 13).

Game$^{a)1)}$ 14. In this game, the challenger applies the following changes. When the targeted instance π computes an AE ciphertext from plaintext data p and outputs the corresponding ciphertext c, it stores the ciphertext c and the corresponding plaintext p in a table T indexed by the stage's cid, its current pss,

and an index i corresponding to the message numbering in the run (i.e. $i = 2$ for m_2, $i = 3$ for m_3, $i = 4$ for m_4): $\mathsf{T}[\pi.\mathsf{cid}[t], \pi.\mathsf{pss}[t-1], i] = (c, p)$. When π' receives an AE ciphertext c' (i.e. $m_3 = c_I$ if π' is a responder's instance, and c_R and $m_4 = c'_R$ otherwise), it does the following: it looks for an existing index $(\pi'.\mathsf{cid}[t], \pi'.\mathsf{pss}[t-1], i')$ in T, where i' corresponds to the index of the received message. If the latter exists and the received ciphertext is equal to the first entry c, then it uses the second entry p as decrypted data. Otherwise, π' behaves as in the previous game.

To that point, if π is initiator, then the first AE ciphertext it receives is c_R. Therefore, π and π' share the plaintext carried in c_R (because the two instances compute the same AE parameters hk_R, n_R, as explained above). Hence, they compute the same AE parameters hk_I, n_I. This implies that π' uses these same AE parameters to decrypt the AE ciphertext (c_I) it receives.

If π is responder, then the first AE ciphertext it receives is c_I. This implies that tr_1 and the AE parameters hk_I, n_I are shared by π and π' (as explained above). Consequently, π and π' share also the plaintext carried in c_I. Therefore, they compute the same value tr_2 as well as the same AE parameters hk'_R, n'_R. This implies that π' uses these same AE parameters to decrypt the last AE ciphertext (c'_R) it receives. Recall also that, as explained above, π' uses the same AE parameters hk_R, n_R as π to decrypt the first AE ciphertext (c_R) it receives.

Overall, π and π' share the same AE parameters to encrypt and decrypt respectively the AE ciphertexts. This implies that the change done in this game is sound and indistinguishable by the adversary.

Therefore

$$\mathsf{adv}_{13}^{a)1)} = \mathsf{adv}_{14}^{a)1)}$$

Game$^{a)1)}$ *15.* In this game, the challenger replaces the AE ciphertext c_I in message m_3 (resp. both c_R in m_2 and $c'_R = m_4$) if π is an initiator's (resp. responder's) instance with a value of same size uniformly drawn at random. π and π' keep using table T to respectively store the (uniformly random) ciphertext c and plaintext p, and to recover the corresponding plaintext.

Since the encryption key is indistinguishable from a value of same size uniformly chosen at random (by Game$^{a)1)}$ 11), this reduces to the ability for an adversary $\mathcal{B}_5$ to break the LH security of the AE scheme.

During stage t, $\mathcal{B}_5$ makes at most one query to its Enc and Dec oracles (i.e. $q_{\mathrm{E}} \leq 1$ and $q_{\mathrm{D}} \leq 1$) to encrypt the data protected in c_R, c_I, and c'_R respectively, and decrypt it (if π' shares pss with π).

Therefore, we have

$$\mathsf{adv}_{14}^{a)1)} \leq \mathsf{adv}_{15}^{a)1)} + 3 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{lh}}(1, 1)$$

To that point, any AE ciphertext sent by the targeted instance π (in particular the message carrying the encrypted certificate of π's parent party) is replaced by a value of same size uniformly chosen at random (by Game$^{a)1)}$ 15). Only an honest instance that shares the same cid in stage t and pss (at beginning of

the stage) as π is able to recover the corresponding plaintext (by Games$^{a)1)}$ 13 and 14). In particular, if π and π' do not share the same value pss at beginning of stage t, then π does not send the encrypted certificate (by Games$^{a)1)}$ 13). If π and π' share pss, then π' recovers the plaintext from $\mathsf{T}[\pi'.\mathsf{cid}[t], pss, i']$ where i' is index of the message received by π' and containing the ciphertext (by Game$^{a)1)}$ 14). But the cid is unique (by Game$^{a)1)}$ 2). Therefore, only π' gets π's plaintext (since it shares the same cid as π).

Note that, even if the adversary queries RevealLCK(u) or RevealLCK(v) such that $vid = \textsc{DrawVid}(u, v)$, it cannot use the revealed private key to verify the signature sent by the targeted instance (by Game$a$)1) 15).

Finally, we recall the assumptions we make: the ephemeral KEM parameters do not help the adversary identify a party.

Consequently, the adversary can do no better than guess the challenge bit b_{id}. Hence, $\mathsf{adv}_{15}^{a)1)} = 0$.

Collecting all the probabilities from Game$^{a)1)}$ 0 to Game$^{a)1)}$ 15 we have:

$$\begin{aligned}\mathsf{adv}_0^{a)1)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}} + (n_P \cdot n_S)^2 \cdot n_T \cdot \left(\frac{1}{2^{\ell_{hk}-1}} + \frac{3}{\ell_k}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) \\ & + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} \\ & + (n_P \cdot n_S)^2 \cdot n_T \cdot \Big(\mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(11) \\ & \quad + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1) + 3 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{lh}}(1,1)\Big)\end{aligned}$$

FreshIdentity a)2). Next, we consider subcase a)2) in predicate FreshIdentity. We show that the two considered instances π and π' must share the same pss at beginning of some stage t'.

Game$^{a)2)}$ 1. In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}_0^{a)2)} \leq \mathsf{adv}_1^{a)2)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}}$$

Game$^{a)2)}$ 2. In this game, the challenger aborts the experiment if either an ephemeral KEM public key epk or ciphertext ct is not unique among all instances and stages. This ensures that the targeted instance shares its cid with at most one instance.

There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_1^{a)2)} \leq \mathsf{adv}_2^{a)2)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$

$Game^{a)2)}$ *3.* In this game, the challenger aborts the experiment if a KEM decapsulation failure occurs. That is, the challenger aborts the experiment if, on the one hand, the encapsulation done by an honest instance with input an ephemeral public key epk outputs a ciphertext ct and a shared secret ss, and, on the other hand, upon reception of ct by the honest instance that has sent epk the decapsulation with the corresponding private key esk outputs a value $ss' \neq ss$. We have that

$$\mathsf{adv}_2^{a)2)} \leq \mathsf{adv}_3^{a)2)} + n_P \cdot n_S \cdot n_T \cdot \delta_{\mathsf{KEM}}$$

$Game^{a)2)}$ *4.* In this game, the challenger aborts the experiment if a collision occurs in Extract outputs among among all instances and stages. Therefore

$$\mathsf{adv}_3^{a)2)} \leq \mathsf{adv}_4^{a)2)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}}$$

$Game^{a)2)}$ *5.* In this game, the challenger aborts the experiment if a collision occurs in Expand outputs among among all instances and stages. Therefore

$$\mathsf{adv}_4^{a)2)} \leq \mathsf{adv}_5^{a)2)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}}$$

$Game^{a)2)}$ *6.* In this game, the challenger aborts the experiment if the key k_0 used by any instance to key Expand in stage $t' \geq 1$ is equal to the key k_1 or k_2 used to key Expand in stage t', or is equal to the key k_0, k_1 or k_2 used to key Expand in stages u, $t' < u < t$, or is equal to the key k_0 or k_1 used to key Expand in stage t.

Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (when keyed with a pseudo-random key). That is, in the latter case, we will replace Expand with a random function. Therefore, we must be able to distinguish the keys that key Expand when we want to leverage its CR property and the keys that key Expand which will eventually be replaced with a random function.

Let us first consider one instance. If the keys k_1, k_2 in stage t', and k_0, k_1, k_2 in stages u, $t' < u < t$, and k_0 and k_1 in stage t are chosen uniformly at random, then the probability that one of these keys collide with the key k_0 used in stage t' is $1/2^{\ell_k}$. Therefore, the probability that none of these keys (in stages t', and u, $t' < u < t$, and t) collide with k_0 used in stage t' is

$$\left(1 - \frac{1}{2^{\ell_k}}\right)^2 \times \left(1 - \frac{1}{2^{\ell_k}}\right)^{3(n_T-2)} \times \left(1 - \frac{1}{2^{\ell_k}}\right)^2 = \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}$$

because each instance executes at most n_T stages. Let $A_{\tilde{\pi}}$ be such an event for a given instance $\tilde{\pi}$.

Let us now consider the event that $A_{\tilde{\pi}}$ holds for all instances $\tilde{\pi}$: $\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}$. Let us consider the opposite event that, for at least one instance $\tilde{\pi}$, the key k_0 in stage t' be equal to at least another key k_1 or k_2 in stage t', or k_0, k_1 or k_2 in a stage u, $t' < u < t$, or k_0 or k_1 in stage t. The corresponding event is $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}$.

The challenger aborts the experiment if the latter event holds.

We have $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}} = \bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}$. Therefore, the corresponding probability is

$$\begin{aligned}
\Pr\left[\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}\right] &= \Pr\left[\bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}\right] \\
&\leq \sum_{\tilde{\pi}} \Pr\left[\overline{A_{\tilde{\pi}}}\right] \\
&= \sum_{\tilde{\pi}} (1 - \Pr[A_{\tilde{\pi}}]) \\
&= \sum_{\tilde{\pi}} \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \\
&= n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right)
\end{aligned}$$

because there are at most $n_P \cdot n_S$ instances.

If $\frac{1}{2^{\ell_k}}$ is close to 0, then $1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2} \simeq \frac{3 \cdot n_T - 2}{2^{\ell_k}}$. Hence

$$n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right) \simeq \frac{n_P \cdot n_S \cdot (3 \cdot n_T - 2)}{2^{\ell_k}}.$$

Therefore, we have

$$\mathsf{adv}_5^{a)2)} \leq \mathsf{adv}_6^{a)2)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2}\right)$$

Game$^{a)2)}$ 7. In this game, the challenger aborts the experiment if the parameter *pss* used by any instance in stage t' as input to the function Extract is equal to at least one of the subsequent parameters *pss* used in input to Extract in stages u, $t' < u \leq t$.

This rule follows the same rationale as that of introduced in Game$^{a)2)}$ 6. Indeed, in subsequent games we will rely upon the CR property of the function Extract but also on the PRF security of Extract (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Extract is keyed with distinct keys. Then, we will be able to safely replace Extract with a random function when keyed with a specific (unique) key, and still rely upon the CR property of Extract when keyed with another specific (unique) key.

Let us consider first one instance. If the parameters *pss* used in stages u, $t' < u \leq t$, are chosen uniformly at random, then the probability that one of theses values collide with *pss* used in stage t' is $1/2^{\ell_{pss}}$. Therefore, the probability that none of these values be equal to *pss* used in stage t' is $\left(1 - 2^{-\ell_{pss}}\right)^{n_T - 1}$, because an instance executes at most n_T stages. Let $A_{\tilde{\pi}}$ be such an event for an

instance $\tilde{\pi}$.

Let us now consider the event that $A_{\tilde{\pi}}$ holds for all instances $\tilde{\pi}$: $\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}$. Let us consider the opposite event that, for at least one instance $\tilde{\pi}$, the parameter pss used in stage t' be equal to at least a parameter pss used in a stage u, $t' < u \leq t$. The corresponding event is $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}$. The challenger aborts the experiment if the latter event holds.

We have $\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}} = \bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}$. Therefore, the corresponding probability is

$$\begin{aligned}
\Pr\left[\overline{\bigcap_{\tilde{\pi}} A_{\tilde{\pi}}}\right] &= \Pr\left[\bigcup_{\tilde{\pi}} \overline{A_{\tilde{\pi}}}\right] \\
&\leq \sum_{\tilde{\pi}} \Pr\left[\overline{A_{\tilde{\pi}}}\right] \\
&= \sum_{\tilde{\pi}} (1 - \Pr[A_{\tilde{\pi}}]) \\
&= \sum_{\tilde{\pi}} \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \\
&= n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right)
\end{aligned}$$

because there are at most $n_P \cdot n_S$ instances.

If $\frac{1}{2^{\ell_{pss}}}$ is close to 0, then $1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1} \simeq \frac{n_T - 1}{2^{\ell_{pss}}}$. Hence

$$n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \simeq \frac{n_P \cdot n_S \cdot (n_T - 1)}{2^{\ell_{pss}}}.$$

Therefore, we have that

$$\mathsf{adv}_6^{a)2)} \leq \mathsf{adv}_7^{a)2)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right)$$

Game$^{a)2)}$ *8.* In this game, the challenger aborts the experiment if a collision occurs in outputs of the chosen 2-universal hash function H_{ec}. These outputs are λ-bit long. Therefore, by definition of H_{ec} (cf. Definition 24), we have

$$\mathsf{adv}_7^{a)2)} \leq \mathsf{adv}_8^{a)2)} + \frac{n_P \cdot n_S \cdot n_T}{2^{\lambda}}$$

Game$^{a)2)}$ *9.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, the stage t', and the instance π' that shares the same cid as π and its parent, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_8^{a)2)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \mathsf{adv}_9^{a)2)}$$

Game$^{a)2)}$ *10.* In this game, we replace in stage t' the shared secret ss computed by π with a value $\tilde{ss}$ of same size uniformly drawn at random. We do the same for π' in stage t'. Recall that in stage t', π and π' match. Hence, they share the same KEM public key epk and ciphertext ct. Moreover, by Game$^{a)2)}$ 3, decapsulation failures are excluded. Consequently, π and π' share the same value ss in stage t'. In addition, no query $\text{REVEALECK}(\pi, t')$ nor $\text{REVEALECK}(\pi', t')$ is issued. Therefore, detecting the change implies an adversary $\mathcal{B}_1$ able to break the IND-CPA security of the KEM scheme. Here we rely upon the IND-CPA notion because the two instances π and π' do share the KEM public key and ciphertext. Therefore, we have

$$\mathsf{adv}_9^{a)2)} \leq \mathsf{adv}_{10}^{a)2)} + \mathsf{adv}_{\mathsf{KEM}}^{\mathsf{ind\text{-}cpa}}$$

Game$^{a)2)}$ *11.* In this game, we replace (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss}, \cdot) = \mathsf{Extract}(\cdot, \tilde{ss})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$. We do a similar change at π' (which shares the same cid as well as the same shared secret ss as π).

Since $\tilde{ss}$ is sampled uniformly at random by Game$^{a)2)}$ 10 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_2$ queries its FUNC oracle to get k_1. However, to that point we have no guarantee that the instances π and π' share the same value pss at the beginning of stage t'. If not, the respective inputs dk_0 to FUNC are different for π and π' (because, by Games$^{a)2)}$ 4 and 5, no collision occurs in $\mathsf{Extract}$ or Expand during the computation of k_0 and dk_0). Therefore, we have to count for at most 2 queries to FUNC (i.e. $q_{\text{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{10}^{a)2)} \leq \mathsf{adv}_{11}^{a)2)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2)$$

Game$^{a)2)}$ *12.* As explained in Game$^{a)2)}$ 11, to that point, we cannot ensure that the two instances π and π' compute the same value dk_0, input to compute k_1. It may happen that using distinct values dk_0 as input to $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$, the outputs be equal. We want to preclude such an event. The probability that such a collision occurs at π is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_1. Therefore, in this game, the challenger aborts the experiment if such a collision occurs.

Hence,

$$\mathsf{adv}_{11}^{a)2)} \leq \mathsf{adv}_{12}^{a)2)} + \frac{1}{2^{\ell_k}}$$

Game$^{a)2)}$ *13.* In this game, the challenger aborts the experiment if the key k_0 computed by π' is equal to the key k_1 computed by π, when the two instances do not use the same value pss at beginning of stage t'. Indeed, in subsequent games we will rely upon the CR property of the function Expand but also on the PRF security of Expand (keyed with a pseudo-random key). Therefore, we must be able to distinguish when Expand is keyed with distinct keys. Then, we will be able to safely replace Expand with a random function when keyed with a specific

(unique) key, and still rely upon the CR property of Expand when keyed with another specific (unique) key.

By Game$^{a)2)}$ 11, the ℓ_k-bit long key k_1 computed respectively by π is output by a random function. Therefore, we have that

$$\mathsf{adv}_{12}^{a)2)} \leq \mathsf{adv}_{13}^{a)2)} + \frac{1}{2^{\ell_k}}$$

Game$^{a)2)}$ 14. In this game, we replace (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this, because k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$ (by Game$^{a)2)}$ 11). We apply a similar change at π' if the latter shares the same value pss as π.

Since k_1 is sampled uniformly at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand.

During the stage, $\mathcal{B}_3$ queries its oracle FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1. Since π and π' share cid, one query to FUNC is enough to get hk_R, n_R and dk_1 for π and π' (if the latter uses the same value pss as π). Moreover, π and π' match in stage t'. Therefore (if the two instances share pss at beginning of the stage), they use the same inputs to compute the other parameters (each instance can copy the parameter already computed by its partner instead of computing it anew). Consequently, 7 calls to the oracle FUNC are enough to compute the four parameters (i.e. $q_{\mathrm{FC}} \leq 7$).

Therefore

$$\mathsf{adv}_{13}^{a)2)} \leq \mathsf{adv}_{14}^{a)2)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)$$

Game$^{a)2)}$ 15. By Game$^{a)2)}$ 12, if π and π' use distinct inputs dk_0, then they compute distinct values k_1 (i.e. π' does not use the same function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ as π). We want to exclude the event that, while π and π' use distinct functions, they compute the same AE key (hk_I or hk_R) needed to compute and decrypt the first AE ciphertext received by π (c_I or c_R). By Game$^{a)2)}$ 14, the probability that π computes this same AE key as π' in such a case is $1/2^{\ell_{hk}}$, where ℓ_{hk} is the bitlength of hk_I and hk_R.

Therefore, in this game the challenger aborts the experiment if any of these two such collisions occurs.

Hence

$$\mathsf{adv}_{14}^{a)2)} \leq \mathsf{adv}_{15}^{a)2)} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{a)2)}$ 16. In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{a)2)}$ 14, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_4$ to break the INT-CTXT security of

the AE function.

During a given stage, $\mathcal{B}_4$ makes at most 1 call to DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss (and cid) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_4$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{15}^{a)2)} \leq \mathsf{adv}_{16}^{a)2)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

Let us assume that π and π' do not share the same value pss at beginning of stage t. Then, they compute distinct parameters k_0 (by Game$^{a)2)}$ 4), dk_0 (by Game$^{a)2)}$ 5), k_1 (by Game$^{a)2)}$ 12) and hk_I or hk_R (by Game$^{a)2)}$ 15) depending on whether π is initiator or responder. The transcripts tr_0 and tr_1 are used as AE header in the computation of the AE ciphertext (c_I or c_R). These values are computed from cid which is shared by π and π' in stage t. $\mathsf{cid} = epk \| ct$ is unique (by Game$^{a)2)}$ 2), hence so are tr_0 and tr_1 (by Game$^{a)2)}$ 1). Therefore, no other honest instance but π' computes an AE ciphertext with an AE header equal to tr_0 or tr_1. In addition, π' uses a distinct AE key to compute the AE ciphertext. Consequently, if the AE ciphertext is not valid, then π aborts the run. Otherwise, we can leverage the INT-CTXT security of the AE function to claim that the (valid) AE ciphertext received by π is a forgery. Then, the challenger aborts the experiment (by Game$^{a)2)}$ 16).

Since π and π' match in stage t' (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that both instances do share the same value pss at beginning of stage t'.

Therefore, in the remaining games, we consider that π and π' do share the same value pss at beginning of stage t'.

Game$^{a)2)}$ 17. In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}(dk_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$. We can safely apply the change at both instances because (by the preceding games) we a have excluded the possibility for the two instances to use distinct values dk_1.

Since dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{a)2)}$ 14) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of $\mathsf{Extract}$.

During the stage, $\mathcal{B}_5$ makes at most 1 call to FUNC (i.e. $q_{\mathrm{Fc}} \leq 1$). Indeed, the two instances π and π' use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and by Game$^{a)2)}$ 8, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{a)2)}$ 14 and 16, and the same transcript tr_2 since, again, they are matching).

Therefore

$$\mathsf{adv}_{16}^{a)2)} \leq \mathsf{adv}_{17}^{a)2)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1)$$

Game$^{a)2)}$ *18.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{a)2)}$ 17) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_6$ to break the PRF security of Expand.

We can safely apply this change because the two instances π and π' do compute the same value k_2. Indeed, the two instances use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{a)2)}$ 17). Moreover, the two instances use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{a)2)}$ 14 and 16, and the same transcripts since, again, they are matching).

This implies also that, during the stage, $\mathcal{B}_6$ makes at most 3 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$). Therefore

$$\mathsf{adv}_{17}^{a)2)} \leq \mathsf{adv}_{18}^{a)2)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3)$$

To that point, the two partnered instances π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{a)2)}$ 14). They share also tr_0 and c_R (because the instances match). Therefore, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \texttt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \texttt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{a)2)}$ 8).

This implies that π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and inputs to compute the new value pss, hence they share the same value which is, in addition, indistinguishable from a value uniformly drawn at random (by Game$^{a)2)}$ 18).

Moreover, recall that neither REVEALPSS(π, t') nor REVEALPSS(π', t') is queried. Therefore, at the end of stage t', pss is indistinguishable from a value chosen uniformly at random and known only to π and π'.

Next, we will rely upon the PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ (keyed with pss). The next six games are repeated for stages $t'+1, \ldots, t-1$ (in order to make this appear, we denote each of these games as "Game$^{a)2)}$ $n+$"). Since the following six games rely upon the PRF security of Expand, $\mathsf{Extract}$ and the swap-PRF security of the latter, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake. We recall also that the instances π and π' are matching in stages u (hence, the changes described below apply to

π' as well as π) and neither $\textsc{RevealPSS}(\pi, u)$ nor $\textsc{RevealPSS}(\pi', u)$ is queried for stages $u \in \{t'+1, \ldots, t-1\}$.

Game$^{a)2)}$ *19+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)2)}$ *20+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{a)2)}$ *21+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)2)}$ *22+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{a)2)}$ *23+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)2)}$ *24+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{a)2)}$ 8 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \texttt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{a)2)}$ 8.

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$, shared by π and π' (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$) and unknown to the adversary. Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{18}^{a)2)} \leq\ & \mathsf{adv}_{24+}^{a)2)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \Big)\end{aligned}$$

Now, we consider stage t.

Game$^{a)2)}$ *25.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(pss, \cdot) = \mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$. We do the same at π' since it shares pss with π.

At beginning of stage t, pss is a value indistinguishable from a value of same size uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect a change is upper-bounded by the advantage of an adversary $\mathcal{B}_7$ to break the swap-PRF security of $\mathsf{Extract}$. During stage t, $\mathcal{B}_7$ queries its $\textsc{Func}$ oracle at most once in order to compute k_0 (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{24}^{a)2)} \leq \mathsf{adv}_{25}^{a)2)} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1)$$

Game$^{a)2)}$ *26.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$. By Game$^{a)2)}$ 25, k_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_8$ to break the PRF security of Expand. We do a similar change at π' (since it shares pss with π).

During stage t, $\mathcal{B}_8$ queries its FUNC oracle at most twice in order to compute hk_R^τ and dk_0 (i.e. $q_{\mathrm{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{25}^{a)2)} \leq \mathsf{adv}_{26}^{a)2)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2)$$

Game$^{a)2)}$ *27.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$. By Game$^{a)2)}$ 26, dk_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_9$ to break the PRF security of $\mathsf{Extract}$. We do a similar change at π' (since it shares pss with π).

In stage t, π and π' share cid. Moreover, by Game$^{a)2)}$ 3, decapsulation failures are excluded. Therefore, the two instances share the same KEM value ss. Therefore, $\mathcal{B}_9$ queries its FUNC oracle at most once in order to compute k_1 (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{26}^{a)2)} \leq \mathsf{adv}_{27}^{a)2)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1)$$

To that point, in stage t, π and π' use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (since they share pss at beginning of stage t) unknown to the adversary. They also use the same input ss (since the share cid and by Game$^{a)2)}$ 3). Therefore, π and π' compute the same value k_1.

Game$^{a)2)}$ *28.* In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. By Game$^{a)2)}$ 27, k_1 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_{10}$ to break the PRF security of Expand. We do a similar change at π' (since it shares pss and also ss with π).

During the stage, $\mathcal{B}_{10}$ queries its oracle FUNC to get hk_R, n_R, hk_I, n_I, hk'_R, n'_R and dk_1. Since π and π' share cid in stage t (and pss at beginning of the stage), one query to FUNC is enough to get hk_R, n_R and dk_1 for π and π'. However, we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I, hk'_R, and n'_R. Therefore, it may happen that $\mathcal{B}_{10}$ queries its oracle twice to get each of these parameters (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

Therefore

$$\mathsf{adv}_{27}^{a)2)} \leq \mathsf{adv}_{28}^{a)2)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(11)$$

Game$^{a)2)}$ *29.* In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a

responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{a)2)}$ 28, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_{11}$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_{11}$ makes at most 1 call to Dec in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value cid in stage t and pss (at beginning of the stage) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_{11}$ may have to query (at most 1 time) its Enc oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{28}^{a)2)} \leq \mathsf{adv}_{29}^{a)2)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

To that point, π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ to compute the AE parameters (by Game$^{a)2)}$ 28). In addition, $\pi.\mathsf{cid}[t] = \pi'.\mathsf{cid}[t]$ (by assumption). Therefore, π and π' compute the same value tr_0 and the same AE parameters hk_R, n_R.

If π is a responder's instance, then the (valid) first AE ciphertext received by π is c_I and is computed by π' (by Game$^{a)2)}$ 29). This implies also that tr_1 (the corresponding AE header) as well as the AE parameters hk_I, n_I are shared by π' and π, as otherwise, c_I is a valid forgery (and the challenger aborts the experiment by Game$^{a)2)}$ 29).

Game$^{a)2)}$ 30. In this game, the challenger applies the following changes. When the targeted instance π computes an AE ciphertext from plaintext data p and outputs the corresponding ciphertext c, it stores the ciphertext c and the corresponding plaintext p in a table T indexed by the stage's cid, its current pss, and an index i corresponding to the message numbering in the run (i.e. $i = 2$ for m_2, $i = 3$ for m_3, $i = 4$ for m_4): $\mathsf{T}[\pi.\mathsf{cid}[t], \pi.\mathsf{pss}[t-1], i] = (c, p)$. When π' receives an AE ciphertext c' (i.e. $m_3 = c_I$ if π' is a responder's instance, and c_R and $m_4 = c'_R$ otherwise), it does the following: it looks for an existing index $(\pi'.\mathsf{cid}[t], \pi'.\mathsf{pss}[t-1], i')$ in T, where i' corresponds to the index of the received message. If the latter exists and the received ciphertext is equal to the first entry c, then it uses the second entry p as decrypted data. Otherwise, π' behaves as in the previous game.

To that point, if π is initiator, then the first AE ciphertext it receives is c_R. Therefore, π and π' share the plaintext carried in c_R (because the two instances compute the same AE parameters hk_R, n_R, as explained above). Hence, they compute the same AE parameters hk_I, n_I. This implies that π' uses these same AE parameters to decrypt the AE ciphertext (c_I) it receives.

If π is responder, then the first AE ciphertext it receives is c_I. This implies that tr_1 and the AE parameters hk_I, n_I are shared by π and π' (as explained

above). Consequently, π and π' share also the plaintext carried in c_I. Therefore, they compute the same value tr_2 as well as the same AE parameters hk'_R, n'_R. This implies that π' uses these same AE parameters to decrypt the last AE ciphertext (c'_R) it receives. Recall also that, as explained above, π' uses the same AE parameters hk_R, n_R as π to decrypt the first AE ciphertext (c_R) it receives.

Overall, π and π' share the same AE parameters to encrypt and decrypt respectively the AE ciphertexts. This implies that the change done in this game is sound and indistinguishable by the adversary.

Therefore

$$\mathsf{adv}_{29}^{a)2)} = \mathsf{adv}_{30}^{a)2)}$$

Game$^{a)2)}$ *31.* In this game, the challenger replaces the AE ciphertext c_I in message m_3 (resp. both c_R in m_2 and $c'_R = m_4$) if π is an initiator's (resp. responder's) instance with a value of same size uniformly drawn at random. π and π' keep using table T to respectively store the (uniformly random) ciphertext c and plaintext p, and to recover the corresponding plaintext.

Since the encryption key is indistinguishable from a value of same size uniformly chosen at random (by Game$^{a)2)}$ 28), this reduces to the ability for an adversary $\mathcal{B}_5$ to break the LH security of the AE scheme.

During stage t, $\mathcal{B}_5$ makes at most one query to its Enc and Dec oracles (i.e. $q_{\mathrm{E}} \leq 1$ and $q_{\mathrm{D}} \leq 1$) to encrypt the data protected in c_R, c_I, and c'_R respectively, and decrypt it (if π' shares pss with π).

Therefore, we have

$$\mathsf{adv}_{30}^{a)2)} \leq \mathsf{adv}_{31}^{a)2)} + 3 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{lh}}(1,1)$$

To that point, any AE ciphertext sent by the targeted instance π (in particular the message carrying the encrypted certificate of π's parent party) is replaced by a value of same size uniformly chosen at random (by Game$^{a)2)}$ 31). Only an honest instance that shares the same cid in stage t and pss (at beginning of the stage) as π is able to recover the corresponding plaintext (by Games$^{a)2)}$ 29 and 30). In particular, if π and π' do not share the same value pss at beginning of stage t, then π does not send the encrypted certificate (by Games$^{a)2)}$ 29). If π and π' share pss, then π' recovers the plaintext from $\mathsf{T}[\pi'.\mathsf{cid}[t], pss, i']$ where i' is index of the message received by π' and containing the ciphertext (by Game$^{a)2)}$ 30). But the cid is unique (by Game$^{a)2)}$ 2). Therefore, only π' gets π's plaintext (since it shares the same cid as π).

Note that, even if the adversary queries RevealLCK(u) or RevealLCK(v) such that $vid = \textsc{DrawVid}(u, v)$, it cannot use the revealed private key to verify the signature sent by the targeted instance because the signature is encrypted.

Finally, we recall the assumptions we make: the ephemeral KEM parameters do not help the adversary identify a party.

Consequently, the adversary can do no better than guess the challenge bit b_{id}. Hence, $\mathsf{adv}_{31}^{a)2)} = 0$.

Collecting all the probabilities from Game$^{a)2)}$ 0 to Game$^{a)2)}$ 31 we have:

$$\begin{aligned}
\mathsf{adv}_0^{a)2)} \leq\ & n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{3 \cdot n_T - 2} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T - 1}\right) \\
& + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \frac{1}{2^{\ell_{hk}-1}} + \frac{1}{2^{\ell_k-1}}\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Extract}} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{Expand}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + 4 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) + 3 \cdot \mathsf{adv}^{\mathsf{lh}}_{\mathsf{AE}}(1,1) \\
& \quad + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(11) + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) \\
& \quad\quad + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) \\
& \quad\quad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\Big)
\end{aligned}$$

FreshIdentity a)3). Next, we consider subcase a)3) in predicate FreshIdentity. The reasoning for stage t' of this subcase is the same as that of stage t' of subcase b)2) in predicate FreshAuth. The reasoning for the subsequent stages $(t'+1, \ldots, t)$ is the same as that of stages $t'+1, \ldots, t$ in the previous subcase (a)2) in predicate FreshIdentity).

Game$^{a)3)}$ *1.* Collision resistance of hash function H.

Game$^{a)3)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

Game$^{a)3)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Here, we do not rely upon the (IND-CPA or IND-CCA) security of the KEM scheme. However, its correctness is required as two instances sharing the same cid must compute the same KEM shared secret *ss* (the latter is involved in the computation of the intermediary secret k_1 which the new *pss* value eventually depends on).

Game$^{a)3)}$ *4.* Collision resistance of Extract.

Game$^{a)3)}$ *5.* Collision resistance of Expand.

Game$^{a)3)}$ *6.* Collision probability for the keys k_0, k_1, k_2 when pseudo-random. The challenger aborts the experiment if a parameter k_0 or k_1 used in stage t' is equal to one of the following parameters: k_2 in stage t', or k_0, k_1, k_2 in a stage

u, $t' < u < t$, or k_0, k_1 in stage t. Following a reasoning similar to that of the subcase FreshAuth b)1), we have:

$$\mathsf{adv}_5^{a)3)} \leq \mathsf{adv}_6^{a)3)} + n_P \cdot n_S \cdot \left(1 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)}\right)$$

$Game^{a)3)}$ *7.* Collision probability for the values pss when pseudo-random.

$Game^{a)3)}$ *8.* Collision resistance of H_{ec}.

To that point, we have:

$$\begin{aligned}\mathsf{adv}_0^{a)3)} \leq\ & \mathsf{adv}_8^{a)3)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} \\ & + n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T-1}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right)\end{aligned}$$

$Game^{a)3)}$ *9.* In this game, the challenger aborts the experiment if an honest instance receives an AE ciphertext valid under a key distinct from the key used by the sender to compute the ciphertext. This reduces straightforwardly to the key-committing security of the AE scheme. Therefore

$$\mathsf{adv}_8^{a)3)} \leq \mathsf{adv}_9^{a)3)} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{key\text{-}cmt}}$$

$Game^{a)3)}$ *10.* In this game, the challenger tries to guess the targeted instance π, its parent, the targeted stage t, the putative party and partner instance π', and the stage t' where no query $\textsc{RevealEQK}(\pi, t')$ nor $\textsc{RevealEQK}(\pi', t')$, where $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi'.\mathsf{eqk}[t'] \neq \perp$, is issued, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_9^{a)3)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \mathsf{adv}_{10}^{a)3)}$$

To that point, in stage t', if π and π' use distinct values pss, then they compute distinct values dk_0 (because no collision occurs in Extract and Expand outputs during the computation of k_0 and dk_0, by $\text{Games}^{a)3)}$ 4 and 5. In turn, this implies that the two instances output distinct values k_1 and distinct AE keys to compute and decrypt the first AE ciphertext received by π. Therefore, either the first AE ciphertext received by π is invalid and rejected by π (hence, it does not appear in the list $\mathsf{m_r}$ of the messages received by π), or the ciphertext is valid but the challenger aborts the experiment by $\text{Game}^{a)3)}$ 9.

Since π and π' match in stage t' (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that both instances do share the same value pss at the beginning of

stage t'.

Moreover, since π and π' match in stage t', this implies that epk and ct are shared by π' and π. By unicity of epk and ct, π' is the unique origin instance of π. This implies in addition, that the plaintext data corresponding to c_R and c_I are also shared by the two instances (i.e. the (classical) data related to the quantum key agreement are exchanged through an authenticated channel). Moreover, by assumption, we have that $\pi.\mathsf{eqk}[t'] \neq \perp$ and $\pi'.\mathsf{eqk}[t'] \neq \perp$ (i.e. $flag_{\mathsf{pe}} = \mathtt{true}$ in stage t'). Therefore, the two instances do compute an actual ephemeral quantum secret. Consequently, we can apply Theorem 1 and safely replace $\pi.\mathsf{eqk}[t']$ and $\pi'.\mathsf{eqk}[t']$ with a value of same size uniformly drawn at random.

Game$^{a)3)}$ *11.* In this game, we replace the ephemeral quantum secret computed in stage t' by the targeted instance π and its partner instance π' with a value $\tilde{K}$ of same size uniformly drawn at random.

Tomamichel and Leverrier [109] prove the security of the BB84 protocol based on the following experiment. On the one hand (real context), the real BB84 protocol is executed. On the other hand (ideal context), a BB84 run is executed and the output key is replaced with a value of same length uniformly drawn at random. The maximum probability of distinguishing the context is $p_\Delta = (1 + \epsilon_\Delta)/2$ (see Section 4.7). By Game$^{a)3)}$ 8, collisions in H_{ec} are excluded. Therefore, we can remove the corresponding probability (correctness component) $\epsilon_{ec} = 2^{-\lambda}$ from ϵ_Δ and consider instead the secrecy component only $\epsilon_s = \epsilon_\Delta - \epsilon_{ec}$, as well as the probability $p_s = (1 + \epsilon_s)/2$.

Let F be the event that the adversary detects the change made in this game. We have

$$\begin{aligned}
\mathsf{adv}_{10}^{a)3)} &\leq \mathsf{adv}_{11}^{a)3)} + \Pr[\mathcal{A} \text{ wins} \mid \text{real QKD context}] \\
&\quad - \Pr[\mathcal{A} \text{ wins} \mid \text{ideal QKD context}] \\
&\leq \mathsf{adv}_{11}^{a)3)} + \Pr[F] \\
&\leq \mathsf{adv}_{11}^{a)3)} + 2 \cdot p_s - 1 \\
&= \mathsf{adv}_{11}^{a)3)} + \epsilon_s
\end{aligned}$$

To that point, in stage t', π and π' use $\tilde{K}$ to key $\mathsf{Extract}^{\mathsf{swap}}$ in order to compute k_2.

Game$^{a)3)}$ *12.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}^{\mathsf{swap}}(\tilde{K}, \cdot) = \mathsf{Extract}(\cdot, \tilde{K})$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$.

Since $\tilde{K}$ is uniformly drawn at random by Game$^{a)3)}$ 11 the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_0$ to break the swap-PRF security of $\mathsf{Extract}$.

During a given stage, $\mathcal{B}_0$ makes at most 1 call to FUNC (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{11}^{a)3)} \leq \mathsf{adv}_{12}^{a)3)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1)$$

Game$^{a)3)}$ *13.* In this game, we replace in stage t' (at the targeted instance π and its matching instance π') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$ (by Game$^{a)3)}$ 12) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the PRF security of Expand.

During a given stage, $\mathcal{B}_1$ makes at most 3 calls to FUNC (i.e. $q_{\mathrm{FC}} \leq 3$).

Therefore

$$\mathsf{adv}_{12}^{a)3)} \leq \mathsf{adv}_{13}^{a)3)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3)$$

To that point, at the end of stage t', pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ known only to π and π'. Moreover, recall that neither REVEALPSS(π, t') nor REVEALPSS(π', t') is queried. Therefore, pss is indistinguishable from a random value uniformly chosen at random.

Next, we will rely upon the PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ (keyed with pss). The next six games are repeated for stages $t'+1, \ldots, t-1$ (in order to make this appear, we denote each of these games as "Game$^{a)3)}$ $n+$"). Since the following six games rely upon the PRF security of Expand, $\mathsf{Extract}$ and the swap-PRF security of the latter, we do not rewrite the similar reasoning already given above. We merely indicate the security properties at stake. We recall also that the instances π and π' are matching in stages u (hence, the changes described below apply to π' as well as π) and neither REVEALPSS(π, u) nor REVEALPSS(π', u) is queried for stages $u \in \{t'+1, \ldots, t-1\}$.

Game$^{a)3)}$ *14+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)3)}$ *15+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{a)3)}$ *16+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)3)}$ *17+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{a)3)}$ *18+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)3)}$ *19+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{a)3)}$ 8 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \mathtt{false}$ then they both use $W = 0$, otherwise one

instance uses K_I and the other K_R which must be equal by Game$^{a)3)}$ 8.

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$, shared by π and π' (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$) and unknown to the adversary. Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{13}^{a)3)} \leq\ & \mathsf{adv}_{19+}^{a)3)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + 2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) \\ & \qquad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)\Big)\end{aligned}$$

Now, we consider stage t.

Game$^{a)3)}$ 20. In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(pss, \cdot) = \mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$. We do the same at π' since it shares pss with π.

At beginning of stage t, pss is a value indistinguishable from a value of same size uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect a change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the swap-PRF security of $\mathsf{Extract}$. During stage t, $\mathcal{B}_2$ queries its FUNC oracle at most once in order to compute k_0 (i.e. $q_{\mathrm{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{19}^{a)3)} \leq \mathsf{adv}_{20}^{a)3)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1)$$

Game$^{a)3)}$ 21. In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$. By Game$^{a)3)}$ 20, k_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand. We do a similar change at π' (since it shares pss with π).

During stage t, $\mathcal{B}_3$ queries its FUNC oracle at most twice in order to compute hk_R^τ and dk_0 (i.e. $q_{\mathrm{FC}} \leq 2$).

Therefore

$$\mathsf{adv}_{20}^{a)3)} \leq \mathsf{adv}_{21}^{a)3)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2)$$

Game$^{a)3)}$ 22. In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$. By Game$^{a)3)}$ 21, dk_0 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_4$ to break the PRF security of $\mathsf{Extract}$. We do a similar change at π' (since it shares pss with π).

In stage t, π and π' share cid. Moreover, by Game$^{a)3)}$ 3, decapsulation failures are excluded. Therefore, the two instances share the same KEM value ss. Therefore, $\mathcal{B}_4$ queries its FUNC oracle at most once in order to compute k_1 (i.e.

$q_{\text{FC}} \leq 1$).

Therefore

$$\mathsf{adv}_{21}^{a)3)} \leq \mathsf{adv}_{22}^{a)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

To that point, in stage t, π and π' use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ (since they share pss at beginning of stage t) unknown to the adversary. They also use the same input ss (since the share cid and by Game$^{a)3)}$ 3). Therefore, π and π' compute the same value k_1.

Game$^{a)3)}$ 23. In this game, we replace in stage t (at the targeted instance π) the function $\mathsf{Expand}(k_1,\cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. By Game$^{a)3)}$ 22, k_1 is now uniformly chosen at random. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of Expand. We do a similar change at π' (since it shares pss and also ss with π).

During the stage, $\mathcal{B}_5$ queries its oracle FUNC to get hk_R ,n_R, hk_I, n_I, hk'_R, n'_R and dk_1. Since π and π' share cid in stage t (and pss at beginning of the stage), one query to FUNC is enough to get hk_R, n_R and dk_1 for π and π'. However, we cannot ensure that π' uses the same inputs as π to compute hk_I, n_I, hk'_R, n'_R. Therefore, it may happen that $\mathcal{B}_5$ queries its oracle twice to get each of these parameters (i.e. $q_{\text{FC}} \leq 3 + 2 \times 4$).

Therefore

$$\mathsf{adv}_{22}^{a)3)} \leq \mathsf{adv}_{23}^{a)3)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(11)$$

Game$^{a)3)}$ 24. In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{a)3)}$ 23, $\mathsf{Expand}(k_1,\cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_6$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_6$ makes at most 1 call to DEC in order to decrypt c_I or c_R (i.e. $q_{\text{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value cid in stage t and pss (at beginning of the stage) as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_6$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\text{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{23}^{a)3)} \leq \mathsf{adv}_{24}^{a)3)} + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1)$$

To that point, π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ to compute the AE parameters (by Game$^{a)3)}$ 23). In addition, $\pi.\mathsf{cid}[t] = \pi'.\mathsf{cid}[t]$ (by assumption). Therefore, π and π' compute the same value tr_0 and the same AE

parameters hk_R, n_R.

If π is a responder's instance, then the (valid) first AE ciphertext received by π is c_I and is computed by π' (by Game$^{a)3)}$ 24). This implies also that tr_1 (the corresponding AE header) as well as the AE parameters hk_I, n_I are shared by π' and π, as otherwise, c_I is a valid forgery (and the challenger aborts the experiment by Game$^{a)3)}$ 24).

Game$^{a)3)}$ 25. In this game, the challenger applies the following changes. When the targeted instance π computes an AE ciphertext from plaintext data p and outputs the corresponding ciphertext c, it stores the ciphertext c and the corresponding plaintext p in a table T indexed by the stage's cid, its current pss, and an index i corresponding to the message numbering in the run (i.e. $i = 2$ for m_2, $i = 3$ for m_3, $i = 4$ for m_4): $\mathsf{T}[\pi.\mathsf{cid}[t], \pi.\mathsf{pss}[t-1], i] = (c, p)$. When π' receives an AE ciphertext c' (i.e. $m_3 = c_I$ if π' is a responder's instance, and c_R and $m_4 = c'_R$ otherwise), it does the following: it looks for an existing index $(\pi'.\mathsf{cid}[t], \pi'.\mathsf{pss}[t-1], i')$ in T, where i' corresponds to the index of the received message. If the latter exists and the received ciphertext is equal to the first entry c, then it uses the second entry p as decrypted data. Otherwise, π' behaves as in the previous game.

To that point, if π is initiator, then the first AE ciphertext it receives is c_R. Therefore, π and π' share the plaintext carried in c_R (because the two instances compute the same AE parameters hk_R, n_R, as explained above). Hence, they compute the same AE parameters hk_I, n_I. This implies that π' uses these same AE parameters to decrypt the AE ciphertext (c_I) it receives.

If π is responder, then the first AE ciphertext it receives is c_I. This implies that tr_1 and the AE parameters hk_I, n_I are shared by π and π' (as explained above). Consequently, π and π' share also the plaintext carried in c_I. Therefore, they compute the same value tr_2 as well as the same AE parameters hk'_R, n'_R. This implies that π' uses these same AE parameters to decrypt the last AE ciphertext (c'_R) it receives. Recall also that, as explained above, π' uses the same AE parameters hk_R, n_R as π to decrypt the first AE ciphertext (c_R) it receives.

Overall, π and π' share the same AE parameters to encrypt and decrypt respectively the AE ciphertexts. This implies that the change done in this game is sound and indistinguishable by the adversary.

Therefore

$$\mathsf{adv}_{24}^{a)3)} = \mathsf{adv}_{25}^{a)3)}$$

Game$^{a)3)}$ 26. In this game, the challenger replaces the AE ciphertext c_I in message m_3 (resp. both c_R in m_2 and $c'_R = m_4$) if π is an initiator's (resp. responder's) instance with a value of same size uniformly drawn at random. π and π' keep using table T to respectively store the (uniformly random) ciphertext c and plaintext p, and to recover the corresponding plaintext.

Since the encryption key is indistinguishable from a value of same size uniformly chosen at random (by Game$^{a)3)}$ 23), this reduces to the ability for an adversary $\mathcal{B}_5$ to break the LH security of the AE scheme.

During stage t, $\mathcal{B}_5$ makes at most one query to its Enc and Dec oracles (i.e.

$q_{\mathrm{E}} \leq 1$ and $q_{\mathrm{D}} \leq 1$) to encrypt the data protected in c_R, c_I, and c'_R respectively, and decrypt it (if π' shares pss with π).

Therefore, we have

$$\mathsf{adv}_{25}^{a)3)} \leq \mathsf{adv}_{26}^{a)3)} + 3 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{lh}}(1,1)$$

The conclusion is the same as in the previous subcase a)2). Consequently, the adversary can do no better than guess the challenge bit b_{id}. Hence, $\mathsf{adv}_{26}^{a)3)} = 0$.

Collecting all the probabilities from Game$^{a)3)}$ 0 to Game$^{a)3)}$ 26 we have:

$$\begin{aligned}
\mathsf{adv}_{0}^{a)3)} \leq\ & n_P \cdot n_S \cdot \left(2 - \left(1 - \frac{1}{2^{\ell_k}}\right)^{6(n_T-1)} - \left(1 - \frac{1}{2^{\ell_{pss}}}\right)^{n_T-1}\right) \\
& + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \epsilon_s\right) \\
& + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{cr}} + \mathsf{adv}_{\mathsf{AE}}^{\mathsf{key\text{-}cmt}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(11) + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1) + 3 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{lh}}(1,1) \\
& \quad + n_T \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + (2n_T - 3) \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) \\
& \quad + (n_T - 1) \cdot \Big(\mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3)\Big) \\
& \quad + (n_T - 2) \cdot \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)\Big)
\end{aligned}$$

FreshIdentity a)4). Now, we consider subcase a)4) in predicate FreshIdentity. The reasoning for stages $1, \ldots, t-1$ of this subcase is the same as that of stages $1, \ldots, t-1$ of subcase b)3) in predicate FreshAuth. The reasoning for stage t is the same as that of stage t in FreshIdentity a)2).

Game$^{a)4)}$ *1.* Collision resistance of hash function H.

Game$^{a)4)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

Game$^{a)4)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Here, we do not rely upon the (IND-CPA or IND-CCA) security of the KEM scheme. However, its correctness is required as two instances sharing the same $\mathsf{cid} = epk \| ct$ must compute the same KEM shared secret ss (the latter is involved in the computation of the intermediary secret values and keys).

Game$^{a)4)}$ *4.* Collision resistance of H_{ec}.

To that point, we have

$$\mathsf{adv}_0^{a)4)} \leq \mathsf{adv}_4^{a)4)} + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right)$$

Game$^{a)4)}$ *5.* In this game, the challenger tries to guess the targeted instance π, its parent, the putative partner party and instance π', and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_4^{a)4)} \leq (n_P \cdot n_S)^2 \cdot \mathsf{adv}_5^{a)4)}$$

Game$^{a)4)}$ *6.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Extract}^{\mathsf{swap}}(pss, \cdot) = \mathsf{Extract}(\cdot, pss)$ with a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$.

Since pss is initialised with a value uniformly drawn at random, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_0$ to break the swap-PRF security of $\mathsf{Extract}$. During a given stage, $\mathcal{B}_0$ queries its FUNC oracle in order to compute k_0 (i.e. $q_{\mathrm{FC}} \leq 1$).

We do a similar change at π' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_5^{a)4)} \leq \mathsf{adv}_6^{a)4)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1)$$

Game$^{a)4)}$ *7.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Expand}(k_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$. We can do this because k_0 computed by π is now uniformly drawn at random by Game$^{a)4)}$ 6. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_1$ to break the PRF security of Expand. During a given stage, $\mathcal{B}_1$ makes at most two queries to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 2$).

We apply a similar change at π' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_6^{a)4)} \leq \mathsf{adv}_7^{a)4)} + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2)$$

Game$^{a)4)}$ *8.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Extract}(dk_0, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$. We can do this because dk_0 computed by π is now uniformly drawn at random by Game$^{a)4)}$ 7. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_2$ to break the PRF security of $\mathsf{Extract}$. During a given stage, $\mathcal{B}_2$ makes at most one query to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 1$).

We apply a similar change at π' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_7^{a)4)} \leq \mathsf{adv}_8^{a)4)} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1)$$

Game$^{a)4)}$ *9.* In this game, we replace in stage 1 (at the targeted instance π) the function $\mathsf{Expand}(k_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. We can do this because k_1 computed by π is now uniformly drawn at random by Game$^{a)4)}$ 8. Therefore, the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_3$ to break the PRF security of Expand. During a given stage, $\mathcal{B}_3$ makes at most 7 queries to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 7$).

We apply a similar change at π' if the latter uses the same value pss as π (at beginning of stage 1).

Therefore

$$\mathsf{adv}_8^{a)4)} \leq \mathsf{adv}_9^{a)4)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)$$

Game$^{a)4)}$ *10.* If π and π' do not share the same value pss, they compute distinct keys k_0, dk_0, k_1, and hk_R and hk_I unless a collision occurs when π and π' choose each of the latter parameters. We want to exclude such events. Therefore, the challenger aborts the experiment if any such collision occurs.

When π and π' do not share pss, the probability that π computes the same value k_0 (or k_1) as π' is $1/2^{\ell_k}$ where ℓ_k is the bitlength of k_0 and k_1. The probability that π computes the same value dk_0 as π' is $1/2^{\ell_{dk}}$ where ℓ_{dk} is the bitlength of dk_0. The probability that π computes the same value hk_R (or hk_I, both values being uniformly drawn at random by Game$^{a)4)}$ 9) as π' is $1/2^{\ell_{hk}}$ where ℓ_{hk} is the bitlength of hk_R and hk_I.

Therefore

$$\mathsf{adv}_9^{a)4)} \leq \mathsf{adv}_{10}^{a)4)} + \frac{1}{2^{\ell_k - 1}} + \frac{1}{2^{\ell_{dk}}} + \frac{1}{2^{\ell_{hk}-1}}$$

Game$^{a)4)}$ *11.* In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{a)4)}$ 9, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_4$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_4$ makes at most 1 call to DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$). If the ciphertext comes from an honest instance which uses the same value pss as π, then both instances use the same random function to compute the AE key, hence use the same AE key. In such a case, $\mathcal{B}_4$ may have to query (at most 1 time) its ENC oracle to compute the ciphertext (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{10}^{a)4)} \leq \mathsf{adv}_{11}^{a)4)} + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1, 1)$$

Let us assume that π and π' do not share the same value pss at beginning of stage t. Then, they compute distinct parameters k_0, dk_0, k_1, and hk_I or hk_R (by Game$^{a)4)}$ 10) depending on whether π is initiator or responder. The transcripts

tr_0 and tr_1 are used as AE header in the computation of the AE ciphertext (c_I or c_R). These values are computed from cid which is shared by π and π' in stage t. $\mathsf{cid} = epk \| ct$ is unique (by Game$^{a)4)}$ 2), hence so are tr_0 and tr_1 (by Game$^{a)4)}$ 1). Therefore, no other honest instance but π' computes an AE ciphertext with an AE header equal to tr_0 or tr_1. In addition, π' uses a distinct AE key to compute the AE ciphertext. Consequently, if the AE ciphertext is not valid, then π aborts the run. Otherwise, we can leverage the INT-CTXT security of the AE function to claim that the (valid) AE ciphertext received by π is a forgery. Then, the challenger aborts the experiment (by Game$^{a)4)}$ 11).

Since π and π' match in stage t' (i.e. they agree on the messages sent and received on either side), if the challenger does not abort the experiment, then this implies that both instances do share the same value pss at beginning of stage t'.

Therefore, in the remaining games, we consider that π and π' do share the same value pss at beginning of stage t'.

Game$^{a)4)}$ 12. In this game, we replace in stage 1 (at the targeted instance π and its matching instance π') the function $\mathsf{Extract}(dk_1, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$.

Since dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{a)4)}$ 9) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_5$ to break the PRF security of $\mathsf{Extract}$.

During the stage, $\mathcal{B}_5$ makes at most 1 call to its FUNC oracle (i.e. $q_{\mathrm{FC}} \leq 1$). Indeed, the two instances π and π' use the same input W because they share the ciphertext c'_R (since they are matching in stage 1 by assumption) and by Game$^{a)4)}$ 4, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{a)4)}$ 9 and 11, and the same transcript tr_2 since, again, they are matching).

Therefore

$$\mathsf{adv}_{11}^{a)4)} \leq \mathsf{adv}_{12}^{a)4)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)$$

Game$^{a)4)}$ 13. In this game, we replace in stage 1 (at the targeted instance π and its matching instance π') the function $\mathsf{Expand}(k_2, \cdot)$ with a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$.

Since k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{a)4)}$ 12) the probability for $\mathcal{A}$ to detect the change is upper-bounded by the advantage of an adversary $\mathcal{B}_6$ to break the PRF security of Expand.

We can safely apply this change because the two instances π and π' do compute the same value k_2. Indeed, the two instances use the same random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (by Game$^{a)4)}$ 12). Moreover, the two instances use the same input W because they share the ciphertext c'_R (since they are matching in stage t' by assumption) and by Game$^{a)4)}$ 4, and use the same AE parameters (hk'_R, n'_R) to encrypt and decrypt c'_R (because they use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ by Games$^{a)4)}$ 9 and 11, and the same transcripts since, again, they are matching).

This implies also that, during the stage, $\mathcal{B}_6$ makes at most 3 call to FUNC

(i.e. $q_{\mathrm{FC}} \leq 3$).

Therefore

$$\mathsf{adv}_{12}^{a)4)} \leq \mathsf{adv}_{13}^{a)4)} + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)$$

To that point, the two partnered instances π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ (by Game$^{a)4)}$ 9). They share also tr_0 and c_R (because the instances match). Therefore, they compute the same parameters hk_R, n_R. Consequently, they share also the inputs to compute tr_1, hence tr_1. This implies that the two instances compute the same parameters hk_I, n_I. Since they share also c_I (because π and π' match), they use the same inputs to compute tr_2, hence they share the latter. Consequently, they also share hk'_R. Therefore, they share the plaintext corresponding to c'_R, in particular $flag_{\mathsf{pe}}$. The latter implies also that the two instances share the inputs in the computation of tr_3, hence tr_3.

If $flag_{\mathsf{pe}} = \texttt{false}$, then both instances use $W = 0$ in the computation of k_2. If $flag_{\mathsf{pe}} = \texttt{true}$, then the initiator's instance and the responder's instance use $W = K_I$ and $W = K_R$ respectively in computation of k_2. That is, the two instances use the same value W unless a collision occurs in H_{ec} outputs (which is excluded by Game$^{a)4)}$ 4).

This implies that π and π' use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ (by Game$^{a)4)}$ 13) known only to them, and the same inputs to compute the new value pss. Hence they share this value. Moreover, neither RevealPSS$(\pi, 0)$ nor RevealPSS$(\pi', 0)$ is queried, and RevealPSS$(\pi, 1)$ and RevealPSS$(\pi', 1)$ are not queried either. Therefore, at the end of stage 1, pss is indistinguishable from a value chosen uniformly at random and known only to π and π'.

Now, we guess the stage t.

Game$^{a)4)}$ 14. In this game, the challenger tries to guess the stage t and aborts if it does not succeed. Therefore

$$\mathsf{adv}_{13}^{a)4)} \leq n_T \cdot \mathsf{adv}_{14}^{a)4)}$$

In the subsequent games, we use the swap-PRF security of Extract and the PRF security of Extract and Expand to eventually argue that the value pss at the beginning of stage t is random and known only to π and π'. Since the reasoning relies upon security properties already mentioned, once again we only indicate the properties at stake.

The next six games are repeated for stages u, $1 < u < t$ (in order to make this appear, we denote each of these games as "Game$^{a)4)}$ n+"). The corresponding changes are done at π and its matching instance π' in stages u.

In stages u, $1 < u < t$, π and π' share the inputs to the different random functions (replacing Extract and Expand) because they match, compute the same intermediary keys (hence use the same random functions), and there is no collision in H_{ec} outputs (by Game$^{a)4)}$ 4). Moreover, recall that RevealPSS(π, u) and RevealPSS(π', u) are not queried where π and π' match in stages u, $1 < u < t$.

Game$^{a)4)}$ 15+. PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)4)}$ *16+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{a)4)}$ *17+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)4)}$ *18+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{a)4)}$ *19+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)4)}$ *20+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

To that point, at the end of stage $t-1$, pss is output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and shared by π and π' (i.e. $\pi.\mathsf{pss}[t-1] = \pi'.\mathsf{pss}[t-1]$).

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{14}^{a)4)} \leq\ & \mathsf{adv}_{20+}^{a)4)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + 2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) \\ & \qquad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7) \Big)\end{aligned}$$

Now, we consider stage t.

The successive games are the same as those of $\mathsf{FreshIdentity}$ a)2) for stage t. Therefore, we merely indicate the security properties at stake.

Game$^{a)4)}$ *21.* Swap-PRF security of $\mathsf{Extract}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)4)}$ *22.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{a)4)}$ *23.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{a)4)}$ *24.* PRF security of Expand keyed with k_1. 3 calls to the random function are sufficient to compute hk_R, n_R and dk_1 because π and π' share also the same inputs used in the computation (since π and π' share cid and pss). But, we cannot ensure that the two instances use the same inputs to compute hk_I, n_I, hk'_R, n'_R. Hence, 2×4 calls to the random function may be necessary (i.e. $q_{\mathrm{FC}} \leq 3 + 2 \times 4$).

hk_R, n_R, hk_I, n_I, hk'_R, n'_R, dk_1 are now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

Game$^{a)4)}$ *25.* INT-CTXT security of the AE function used to compute the first ciphertext received by π (i.e. c_R or c_I).

Game$^{a)4)}$ *26.* LH security of the AE functions used by π to compute the ciphertexts (i.e. $m_3 = c_I$ if π is an initiator's instance, and c_R in m_2 and $m_4 = c'_R$ otherwise).

To that point we have

$$\begin{aligned}\mathsf{adv}^{a)4)}_{20+} \leq \mathsf{adv}^{a)4)}_{26} &+ \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(11) \\ &+2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) + 3 \cdot \mathsf{adv}^{\mathsf{lh}}_{\mathsf{AE}}(1,1)\end{aligned}$$

The conclusion is the same as in FreshIdentity a)2). Consequently, the adversary can do no better than guess the challenge bit b_{id}. Hence, $\mathsf{adv}^{a)4)}_{26} = 0$.

Collecting all the probabilities from Game$^{a)4)}$ 0 to Game$^{a)4)}$ 26 we have:

$$\begin{aligned}\mathsf{adv}^{a)4)}_{0} \leq\ & (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\ &+(n_P \cdot n_S)^2 \cdot \left(\frac{1}{2^{\ell_k-1}} + \frac{1}{2^{\ell_{dk}}} + \frac{1}{2^{\ell_{hk}-1}}\right) \\ &+\mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} \\ &+(n_P \cdot n_S)^2 \cdot \Big(n_T \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(11) + 3 \cdot \mathsf{adv}^{\mathsf{lh}}_{\mathsf{AE}}(1,1)\Big) \\ &\quad +2(n_T+1) \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) \\ &\quad +(n_T(n_T-1)+1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2)\Big) \\ &\quad +(n_T(2 \cdot n_T - 3)+2) \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) \\ &\quad +(n_T(n_T-2)+1) \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7)\Big)\Big)\end{aligned}$$

FreshIdentity b)1). Now, we consider case b) in predicate FreshIdentity.

In these cases, the strategy is the following. The targeted instance must receive a valid first AE ciphertext before sending its encrypted certificate. We rely upon the unforgeability of the AE scheme to show that the targeted instance rejects the first AE ciphertext it received and never sends its encrypted certificate (subcases b)2) to b)4)). When the targeted instance is a responder's instance, we rely also upon the EUF-CMA security of the signature scheme (subcase b)1)) because, then, the adversary must authenticate first (with a signature).

We start with subcase b)1).

Game$^{b)1)}$ *1.* In this game, the challenger aborts the experiment if two honest instances compute the same digest for different inputs of hash function H. If this happens, it implies an adversary $\mathcal{B}_0$ that can break the collision resistance of H. Therefore

$$\mathsf{adv}^{b)1)}_{0} \leq \mathsf{adv}^{b)1)}_{1} + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}}$$

$Game^{b)1)}$ *2.* In this game, the challenger aborts the experiment if either an ephemeral KEM public key epk or ciphertext ct is not unique among all instances and stages. There are at most $(n_P \cdot n_S \cdot n_T)^2/2$ pairs of public keys (resp. ciphertexts), therefore we have that

$$\mathsf{adv}_1^{b)1)} \leq \mathsf{adv}_2^{b)1)} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right)$$

$Game^{b)1)}$ *3.* In this game, the challenger tries to guess the targeted instance π, its parent and the targeted stage t, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_2^{b)1)} \leq n_P \cdot n_S \cdot n_T \cdot \mathsf{adv}_3^{b)1)}$$

$Game^{b)1)}$ *4.* In this game, the challenger tries to guess which public key (i.e. which corresponding party) the signature received by π is verified with, and it aborts if the guess is incorrect. Therefore we have:

$$\mathsf{adv}_3^{b)1)} \leq n_P \cdot \mathsf{adv}_4^{b)1)}$$

$Game^{b)1)}$ *5.* In this game, the challenger aborts the experiment if the adversary $\mathcal{A}$ produces a valid signature under an uncorrupted key for a message which was not output by an honest instance. This reduces straightforwardly to the security of the signature scheme SIG. We show how to build an adversary $\mathcal{B}_1$ against the EUF-CMA security of the signature scheme. This adversary is given a signing oracle. It behaves as follows. $\mathcal{B}_1$ sets $lpk_w = pk^*$ for some party w, with pk^* from the signature security experiment, and generates all other protocol parameters as specified. It uses its signing oracle in order to output honest signatures valid under lpk_w. If $\mathcal{A}$ submits a valid signature under lpk_w that was not output by an honest instance, then $\mathcal{B}_1$ uses these message and signature to break the EUF-CMA security of the signature scheme. Since this message was not signed before with the corresponding private key (because $\mathcal{B}_1$ did not query its signing oracle with this message to get the signature), then it is a valid forgery (i.e. $\mathcal{B}_1$ wins).

During a given stage, $\mathcal{B}_1$ makes at most 1 call to SIGN when it needs to output a signature for an honest instance (corresponding to party w) (i.e. $q_{\mathrm{SG}} \leq 1$).

$\mathcal{B}_1$ can answer any REVEALLCK query from $\mathcal{A}$ (since $\mathcal{B}_1$ manages the simulation) but for the private key corresponding to $lpk_w = pk^*$. However, in the context where $\mathcal{A}$ tries to compute a valid signature (on behalf of an honest instance owned by some party v) in order to make an honest instance π accept, the long term (signature) key of the putative partner party v cannot be revealed before the honest targeted instance π accepts, as otherwise $\mathsf{FreshAuth}$ becomes false. Therefore, if a query REVEALLCK(w) is issued before the targeted party accepts, then the simulation is stopped.

Taking into account that there are at most n_P parties, we have

$$\mathsf{adv}_4^{b)1)} \leq \mathsf{adv}_5^{b)1)} + \mathsf{adv}_{\mathsf{SIG}}^{\mathsf{euf\text{-}cma}}(1)$$

To that point, the responder's instance π does not send its encrypted certificate unless it receives a valid signature σ_I. The signature that π must receive protects in particular a hash of the cid which is unique (by $\text{Game}^{b)1)}$ 2) and H outputs do not collide (by $\text{Game}^{b)1)}$ 1). Therefore, no honest instance computes such a signature. If the adversary queries $\text{RevealLCK}(\pi.\mathsf{pid}, t)$ before π accepts, then $\mathsf{FreshIdentity}$ is false. Moreover, if π receives a valid signature not computed by an honest instance, then the challenger aborts the experiment (by $\text{Game}^{b)1)}$ 5). Therefore, π never sends its encrypted certificate and the adversary can do no better than guess the challenge bit b_{id}. Hence, the advantage of the adversary in $\text{Game}^{b)1)}$ 5 is $\mathsf{adv}_5^{b)1)} = 0$.

Collecting all probabilities from $\text{Game}^{b)1)}$ 0 to $\text{Game}^{b)1)}$ 5, we have

$$\mathsf{adv}_0^{b)1)} \leq (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} \right) + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + n_P^2 \cdot n_S \cdot n_T \cdot \mathsf{adv}_{\mathsf{SIG}}^{\mathsf{euf\text{-}cma}} \quad (1)$$

FreshIdentity b)2). Now, we proceed with subcase b)2) in predicate $\mathsf{FreshIdentity}$.

The games related to stages u, $t' \leq u < t$, are similar to those of subcase c)3) in predicate $\mathsf{FreshKey}$. The main difference with the latter subcase lies in the treatment of stage t.

$Game^{b)2)}$ *1.* Collision resistance of hash function H.

$Game^{b)2)}$ *2.* Collision of the ephemeral KEM public key epk and ciphertext ct values.

$Game^{b)2)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

$Game^{b)2)}$ *4.* Collision resistance of H_{ec}.

$Game^{b)2)}$ *5.* Guess of tested instance π and its parent, matching instance π'' and its parent, and stages t' and t.

$Game^{b)2)}$ *6.* IND-CPA security of the ephemeral KEM scheme. Recall that π and π'' are matching in stage t', and $\text{RevealECK}(\pi, t')$ and $\text{RevealECK}(\pi'', t')$ are not queried.

$Game^{b)2)}$ *7.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with $\tilde{ss}$.

To that point we have no guarantee that the instances π and π'' share the same value pss at the beginning of stage t'. If not, the respective inputs dk_0 to Func might be different for π and π''. Therefore, we have to count for at most 2 queries to Func (i.e. $q_{\text{Fc}} \leq 2$).

k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{ss},\cdot)}$.

$Game^{b)2)}$ *8.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\text{Fc}} \leq 7$).

Game$^{b)2)}$ *9.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)2)}$ *10.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Collecting all probabilities from Game$^{b)2)}$ 0 to Game$^{b)2)}$ 10, we have

$$\begin{aligned}\mathsf{adv}_0^{b)2)} \leq\ & (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\ & + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}} + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{b)2)}_{10}\Big)\end{aligned}$$

To that point, at the end of stage t', if π and π'' share the same value pss at beginning of stage t', then they share the same value pss at beginning of stage $t'+1$. In addition, this value is indistinguishable from a value drawn at random.

The following games are repeated for stages $t'+1, \ldots, t-1$.

Game$^{b)2)}$ *11+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)2)}$ *12+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)2)}$ *13+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)2)}$ *14+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{b)2)}$ *15+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)2)}$ *16+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{b)2)}$ 4 to ensure that the matching instances π and π'' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \texttt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{b)2)}$ 4.

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{10}^{b)2)} \leq\ & \mathsf{adv}_{16+}^{b)2)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \Big)\end{aligned}$$

To that point, REVEALPSS(π, u) and REVEALPSS(π'', u), $t' \leq u < t$, are not queried. Consequently, by all the preceding games, at the end of stage $t-1$, the value pss computed by the tested instance π is indistinguishable from a value uniformly drawn at random. This value is shared by π'' if π and π'' share the same value pss at beginning of stage t'.

Now, we consider stage t. The following changes apply to the tested instance π.

Game$^{b)2)}$ *17.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)2)}$ *18.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)2)}$ *19.* PRF security of $\mathsf{Extract}$ keyed with dk_0.

By assumption, the instances π and π'' do not share cid in stage t. Thus, they may not share ss. However, it might happen that they key $\mathsf{Extract}$ with the same value dk_0 (if collisions occur in the computation of the latter parameter by π and π'' respectively). Therefore, we have to count for at most 2 queries to FUNC (i.e. $q_{\mathrm{FC}} \leq 2$).

k_1 is now output by $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$.

Game$^{b)2)}$ *20.* PRF security of Expand keyed with k_1.

Even though the instances π and π'' do not share ss in stage t (by assumption they do not use the same value cid) and pss at beginning of stage t, it might happen that they key Expand with the same value k_1 (if collisions occur in the computation of the latter parameter by π and π'' respectively). In addition, the two instances may not use the same inputs to the random function that replaces $\mathsf{Expand}(k_1,\cdot)$. Therefore, we have to count for at most 2 queries to FUNC when computing hk_R, n_R, hk_I, n_I, hk'_R, n'_R. However, π and π'' use the same (public) input to compute dk_1. Therefore $q_{\mathrm{FC}} \leq 2 \times 6 + 1$.

hk_I and hk_R are now output by $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

Game$^{b)2)}$ *21.* In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{b)2)}$ 20, $\mathsf{Expand}(k_1,\cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_0$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_0$ makes at most 1 call to its oracle DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$).

By Game$^{b)2)}$ 20, π'' is the only honest instance that might use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ as π to compute the AE key used by π to decrypt the first received AE ciphertext. This might happen if collisions occur in the computation of the intermediary parameters (e.g. k_0, dk_0) that lead to the AE key. Indeed, even if π'' shares pss at beginning of stage t' and the ephemeral KEM shared secret ss, it does not share cid in stage t. Hence it does not share tr_0 nor tr_1 with π. Although, it is unlikely that π'' eventually computes the same AE key as π, we do not prevent such an event. Therefore, it might happen that $\mathcal{B}_0$ queries its oracle ENC to output the first AE ciphertext received by π (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{20}^{b)2)} \leq \mathsf{adv}_{21}^{b)2)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\text{int-ctxt}}(1,1)$$

The first AE ciphertext decrypted by π is c_I or c_R (depending on whether π is responder or initiator). This AE ciphertext is decrypted with the AE key hk_I or hk_R. The AE ciphertext is decrypted by π with tr_1 or tr_0 as AE header. These parameters are computed from the ephemeral KEM values epk and ct which are unique (by Game$^{b)2)}$ 2). Hence, the AE header is also unique (by Game$^{b)2)}$ 1). By assumption, no honest instance shares $\mathsf{cid} = epk \| ct$ with π in stage t. Therefore, no honest instance uses tr_1 or tr_0 as AE header to compute the first AE ciphertext received by π. Consequently, if the AE ciphertext is not valid, then π aborts the run (and, in particular, it does not send its certificate). If the AE ciphertext is valid, then we can leverage the INT-CTXT security of the AE function to claim that it is a forgery. In that case, the challenger aborts the experiment (by Game$^{b)2)}$ 21).

Consequently, the targeted instance π never sends its encrypted certificate. Therefore, the adversary can do no better than guess the bit b_{id}, and $\mathsf{adv}_{21}^{b)2)} = 0$.

Collecting all the probabilities from Game$^{b)2)}$ 0 to Game$^{b)2)}$ 21 we have:

$$\begin{aligned}
\mathsf{adv}_{0}^{b)2)} \leq\ & (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} \right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}} \right) \\
& + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}_{\mathsf{KEM}}^{\text{ind-cpa}} + \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Extract}}^{\text{swap-prf}}(2) \\
& \quad + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\text{int-ctxt}}(1,1) + n_T \cdot \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(13) \\
& \quad + (n_T - 1) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\text{swap-prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) \Big) \\
& \quad + (2 \cdot n_T - 3) \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) \Big)
\end{aligned}$$

FreshIdentity b)3). Now, we proceed with subcase b)3) in predicate $\mathsf{FreshIdentity}$.

The games related to stages u, $t' \leq u < t$ are similar to those of subcase c)4) in predicate $\mathsf{FreshKey}$. The main difference with the latter subcase lies in the treatment of stage t.

$Game^{b)3)}$ *1.* Collision resistance of hash function H.

$Game^{b)3)}$ *2.* Collision of the ephemeral KEM public key epk and ciphertext ct values.

$Game^{b)3)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

$Game^{b)3)}$ *4.* Collision resistance of H_{ec}.

$Game^{b)3)}$ *5.* Guess of tested instance π and its parent, matching instance π'' and its parent, and stages t' and t.

$Game^{b)3)}$ *6.* Indistinguishability of the ephemeral quantum secret (it is is now replaced with a value $\tilde{K}$ uniformly chosen at random). In addition, recall that neither $\textsc{RevealEQK}(\pi, t')$ nor $\textsc{RevealEQK}(\pi'', t')$ is queried.

$Game^{b)3)}$ *7.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with $\tilde{K}$.

We cannot ensure that π and π'' use the same value pss at beginning of stage t'. Therefore, it may happen that they compute distinct values dk_1 (inputs to $\mathsf{Extract}^{\mathsf{swap}}$). Therefore, we have to count for at most to calls to $\textsc{Func}$ (i.e. $q_{\textsc{Fc}} \leq 2$).

k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(\tilde{K},\cdot)}$.

$Game^{b)3)}$ *8.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\textsc{Fc}} \leq 3$).

Collecting the probabilities from $\text{Game}^{b)3)}$ 0 to $\text{Game}^{b)3)}$ 8, we have:

$$\begin{aligned}\mathsf{adv}_0^{b)3)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\ & + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \epsilon_s\right) \\ & + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_8^{b)3)}\right)\end{aligned}$$

To that point, at the end of stage t', if π and π'' share the same value pss at beginning of stage t', then they share the same value pss at beginning of stage $t'+1$. In addition, this value is indistinguishable from a value drawn at random.

The following games are repeated for stages $t'+1, \ldots, t-1$.

$Game^{b)3)}$ *9+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

$Game^{b)3)}$ *10+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

$Game^{b)3)}$ *11+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

$Game^{b)3)}$ *12+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

$Game^{b)3)}$ *13+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

$Game^{b)3)}$ *14+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon $\mathrm{Game}^{b)3)}$ 4 to ensure that the matching instances π and π'' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \texttt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by $\mathrm{Game}^{b)3)}$ 4.

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{8}^{b)3)} \leq\ & \mathsf{adv}_{14+}^{b)3)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) \\ & \qquad + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) \Big)\end{aligned}$$

To that point, $\textsc{RevealPSS}(\pi, u)$ and $\textsc{RevealPSS}(\pi'', u)$, $t' \leq u < t$, are not queried. Consequently, by all the preceding games, at the end of stage $t-1$, the value pss computed by the tested instance π is indistinguishable from a value uniformly drawn at random. This value is shared by π'' if π and π'' share the same value pss at beginning of stage t'.

Now, we consider stage t. The following changes apply to the tested instance π.

$Game^{b)3)}$ *15.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

$Game^{b)3)}$ *16.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)3)}$ *17.* PRF security of $\mathsf{Extract}$ keyed with dk_0.

By assumption, the instances π and π'' do not share cid in stage t. Thus, they may not share ss. However, it might happen that they key $\mathsf{Extract}$ with the same value dk_0 (if collisions occur in the computation of the latter parameter by π and π'' respectively). Therefore, we have to count for at most 2 queries to FUNC (i.e. $q_{\mathrm{FC}} \leq 2$).

k_1 is now output by $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$.

Game$^{b)3)}$ *18.* PRF security of Expand keyed with k_1.

Even if the instances π and π'' do not share ss in stage t (by assumption they do not use the same value cid) and pss at beginning of stage t, it might happen that they key Expand with the same value k_1 (if collisions occur in the computation of the latter parameter by π and π'' respectively). In addition, the two instances may not use the same inputs to the random function that replaces $\mathsf{Expand}(k_1,\cdot)$. Therefore, we have to count for at most 2 queries to FUNC when computing hk_R, n_R, hk_I, n_I, hk'_R, n'_R. However, π and π'' use the same (public) input to compute dk_1. Therefore $q_{\mathrm{FC}} \leq 2 \times 6 + 1$.

hk_I and hk_R are now output by $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

Game$^{b)3)}$ *19.* In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{b)3)}$ 18, $\mathsf{Expand}(k_1,\cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_0$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_0$ makes at most 1 call to its oracle DEC in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$).

By Game$^{b)3)}$ 18, π'' is the only honest instance that might use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ as π to compute the AE key used by π to decrypt the first received AE ciphertext. This might happen if collisions occur in the computation of the intermediary parameters (e.g. k_0, dk_0) that lead to the AE key. Indeed, even if π'' shares pss at beginning of stage t' and the ephemeral KEM shared secret ss, it does not share cid in stage t. Hence it does not share tr_0 nor tr_1 with π. Although, it is unlikely that π'' eventually computes the same AE key as π, we do not prevent such an event. Therefore, it might happen that $\mathcal{B}_0$ queries its oracle ENC to output the first AE ciphertext received by π (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{18}^{b)3)} \leq \mathsf{adv}_{19}^{b)3)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

The reasoning and conclusion are the same as those of $\mathsf{FreshIdentity}$ b)2). The adversary can do no better than guess the bit b_{id}, and $\mathsf{adv}_{19}^{b)3)} = 0$.

Collecting all the probabilities from Game$^{b)3)}$ 0 to Game$^{b)3)}$ 19 we have:

$$\begin{aligned}
\mathsf{adv}_0^{b)3)} \leq\ & n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} + \epsilon_s\right) \\
& + \mathsf{adv}^{\mathsf{cr}}_{\mathsf{H}} \\
& + (n_P \cdot n_S \cdot n_T)^2 \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(13) + \mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(2) \\
& \quad + 2 \cdot \mathsf{adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}}(1,1) \\
& \quad + (n_T - 2) \cdot \Big(\mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(7) + 2 \cdot \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Extract}}(1)\Big) \\
& \quad + (n_T - 1) \cdot \Big(\mathsf{adv}^{\mathsf{swap\text{-}prf}}_{\mathsf{Extract}}(1) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(2) + \mathsf{adv}^{\mathsf{prf}}_{\mathsf{Expand}}(3)\Big)\Big)
\end{aligned}$$

FreshIdentity b)4). Now, we proceed with subcase b)4) in predicate FreshIdentity.

The games related to stages u, $t' \leq u < t$, are similar to those of subcase c)5) in predicate FreshKey. The main difference with the latter subcase lies in the treatment of stage t.

Game$^{b)4)}$ *1.* Collision resistance of hash function H.

Game$^{b)4)}$ *2.* Collision of the ephemeral KEM public key *epk* and ciphertext *ct* values.

Game$^{b)4)}$ *3.* Correctness of the KEM scheme (i.e. probability of a decapsulation failure).

Game$^{b)4)}$ *4.* Collision resistance of H_{ec}.

Game$^{b)4)}$ *5.* Guess of tested instance π and its parent, matching instance π' and its parent.

Now, we consider the very first stage (stage 1).

Game$^{b)4)}$ *6.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with *pss* (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)4)}$ *7.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)4)}$ *8.* PRF security of Extract keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)4)}$ *9.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{b)4)}$ *10.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)4)}$ *11.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Collecting all probabilities from Game$^{b)4)}$ 0 to Game$^{b)4)}$ 11, we have

$$\begin{aligned}\mathsf{adv}_0^{b)4)} \leq\ & (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}}\right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}}\right) \\ & + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} \\ & + (n_P \cdot n_S)^2 \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + 2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) \\ & \quad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7) + \mathsf{adv}_{11}^{b)4)}\Big)\end{aligned}$$

To that point, if π and π'' share the same initial value pss, then they share the same value pss at the end of stage 1. In addition, this value is indistinguishable from a value drawn at random (recall that $\textsc{RevealPSS}(\pi, 0)$ and $\textsc{RevealPSS}(\pi'', 0)$ are not queried).

Now, we guess stage t.

Game$^{b)4)}$ *12.* In this game, the challenger tries to guess the stage t and aborts if it does not succeed. Therefore

$$\mathsf{adv}_{11}^{b)4)} \leq n_T \cdot \mathsf{adv}_{12}^{b)4)}$$

The following games are repeated for stages $2, \ldots, t-1$.

Game$^{b)4)}$ *13+.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)4)}$ *14+.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)4)}$ *15+.* PRF security of $\mathsf{Extract}$ keyed with dk_0 (k_1 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)4)}$ *16+.* PRF security of Expand keyed with k_1 (dk_1 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 7$).

Game$^{b)4)}$ *17+.* PRF security of $\mathsf{Extract}$ keyed with dk_1 (k_2 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)4)}$ *18+.* PRF security of Expand keyed with k_2 (pss is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$ and $q_{\mathrm{FC}} \leq 3$).

Note that, in this game, we also rely upon Game$^{b)4)}$ 4 to ensure that the matching instances π and π' use the same value W as input to $\mathsf{RF}_{\mathsf{Extract}(dk_1,\cdot)}$ (hence compute the same value k_2, that is, use the same function $\mathsf{RF}_{\mathsf{Expand}(k_2,\cdot)}$). More specifically, if $flag_{\mathsf{pe}} = \mathtt{false}$ then they both use $W = 0$, otherwise one instance uses K_I and the other K_R which must be equal by Game$^{b)4)}$ 4.

Taking into account that there are at most n_T stages per instance, we have:

$$\begin{aligned}\mathsf{adv}_{12}^{b)4)} \leq\ & \mathsf{adv}_{18+}^{b)4)} \\ & + (n_T - 2) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + 2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) \\ & \qquad + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7)\Big)\end{aligned}$$

To that point, RevealPSS(π, u) and RevealPSS(π'', u), $0 \leq u < t$, are not queried. Consequently, by all the preceding games, at the end of stage $t-1$, the value pss computed by the tested instance π is indistinguishable from a value uniformly drawn at random. This value is shared by π'' if π and π'' share the same initial value pss.

Now, we consider stage t. The following changes apply to the tested instance π.

Game$^{b)4)}$ *19.* PRF security of $\mathsf{Extract}^{\mathsf{swap}}$ keyed with pss (k_0 is now output by a random function $\mathsf{RF}_{\mathsf{Extract}^{\mathsf{swap}}(pss,\cdot)}$ and $q_{\mathrm{FC}} \leq 1$).

Game$^{b)4)}$ *20.* PRF security of Expand keyed with k_0 (dk_0 is now output by a random function $\mathsf{RF}_{\mathsf{Expand}(k_0,\cdot)}$ and $q_{\mathrm{FC}} \leq 2$).

Game$^{b)4)}$ *21.* PRF security of $\mathsf{Extract}$ keyed with dk_0.

By assumption, the instances π and π'' do not share cid in stage t. Thus, they may not share ss. However, it might happen that they key $\mathsf{Extract}$ with the same value dk_0 (if collisions occur in the computation of the latter parameter by π and π'' respectively). Therefore, we have to count for at most 2 queries to Func (i.e. $q_{\mathrm{FC}} \leq 2$).

k_1 is now output by $\mathsf{RF}_{\mathsf{Extract}(dk_0,\cdot)}$.

Game$^{b)4)}$ *22.* PRF security of Expand keyed with k_1.

Even if the instances π and π'' do not share ss in stage t (by assumption they do not use the same value cid) and pss at beginning of stage t, it might happen that they key Expand with the same value k_1 (if collisions occur in the computation of the latter parameter by π and π'' respectively). In addition, the two instances may not use the same inputs to the random function that replaces $\mathsf{Expand}(k_1,\cdot)$. Therefore, we have to count for at most 2 queries to Func when computing hk_R, n_R, hk_I, n_I, hk'_R, n'_R. However, π and π'' use the same (public) input to compute dk_1. Therefore $q_{\mathrm{FC}} \leq 2 \times 6 + 1$.

hk_I and hk_R are now output by $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$.

Game$^{b)4)}$ *23.* In this game, the challenger aborts the experiment if the instance π receives a valid first AE ciphertext (c_I or c_R depending on whether π is a responder's or an initiator's instance) but no honest instance has used the same AE parameters (key, header, nonce) to compute the ciphertext as π to decrypt it.

By Game$^{b)4)}$ 22, $\mathsf{Expand}(k_1, \cdot)$ is replaced with a random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$. Therefore, the AE key (hk_I or hk_R) used to decrypt the ciphertext (c_I or c_R) is indistinguishable from a value sampled at random. Hence, this reduces straightforwardly to the ability for an adversary $\mathcal{B}_0$ to break the INT-CTXT security of the AE function.

During a given stage, $\mathcal{B}_0$ makes at most 1 call to its oracle Dec in order to decrypt c_I or c_R (i.e. $q_{\mathrm{D}} \leq 1$).

By Game$^{b)4)}$ 22, π'' is the only honest instance that might use the same random function $\mathsf{RF}_{\mathsf{Expand}(k_1,\cdot)}$ as π to compute the AE key used by π to decrypt the first received AE ciphertext. This might happen if collisions occur in the computation of the intermediary parameters (e.g. k_0, dk_0) that lead to the AE key. Indeed, even if π'' shares pss at beginning of stage t' and the ephemeral KEM shared secret ss, it does not share cid in stage t. Hence it does not share tr_0 nor tr_1 with π. Although, it is unlikely that π'' eventually computes the same AE key as π, we do not prevent such an event. Therefore, it might happen that $\mathcal{B}_0$ queries its oracle Enc to output the first AE ciphertext received by π (i.e. $q_{\mathrm{E}} \leq 1$).

Consequently, we have

$$\mathsf{adv}_{22}^{b)4)} \leq \mathsf{adv}_{23}^{b)4)} + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1)$$

The reasoning and conclusion are the same as those of $\mathsf{FreshIdentity}$ b)2). The adversary can do no better than guess the bit b_{id}, and $\mathsf{adv}_{23}^{b)4)} = 0$.

Collecting all the probabilities from Game$^{b)4)}$ 0 to Game$^{b)4)}$ 23 we have:

$$\begin{aligned}
\mathsf{adv}_0^{b)4)} \leq\ & (n_P \cdot n_S \cdot n_T)^2 \cdot \left(\frac{1}{2^{\mu_{\mathsf{KEM}}+1}} + \frac{1}{2^{\gamma_{\mathsf{KEM}}+1}} \right) + n_P \cdot n_S \cdot n_T \cdot \left(\delta_{\mathsf{KEM}} + \frac{1}{2^{\lambda}} \right) \\
& + \mathsf{adv}_{\mathsf{H}}^{\mathsf{cr}} \\
& + (n_P \cdot n_S)^2 \cdot \Big(n_T \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(2) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(13) + 2 \cdot \mathsf{adv}_{\mathsf{AE}}^{\mathsf{int\text{-}ctxt}}(1,1) \Big) \\
& \quad + (n_T(n_T-2)+1) \cdot \Big(2 \cdot \mathsf{adv}_{\mathsf{Extract}}^{\mathsf{prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(3) \Big) \\
& \quad + n_T(n_T-1) \cdot \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(7) \\
& \quad + (n_T(n_T-1)+1) \cdot \Big(\mathsf{adv}_{\mathsf{Extract}}^{\mathsf{swap\text{-}prf}}(1) + \mathsf{adv}_{\mathsf{Expand}}^{\mathsf{prf}}(2) \Big) \Big)
\end{aligned}$$

B Changes

We summarise below the main changes since version 20260109:090307 [51].

The corrections based on the remarks and questions from Sebastian Verschoor, Kathrin Hövelmanns, Daan Planken and Christian Schaffner [112] are the following:

- The security experiment is split into as many experiments as security properties (cf. Figures 5 and 6).
- Only one TEST query is allowed in a security experiment.
- A discrepancy is fixed between the definition of the LH property (cf. Figure 14) and the use of the encryption function of the AE scheme in the AKE run (cf. Figure 1).
- The partial intercept-resend scenario considered in several proofs (e.g. FreshAuth b)2), FreshKey c)2) in version 20260109:090307) is in fact already captured by the proof of Tomamichel and Leverrier [109] (through the smooth min-entropy and the parameter estimation bound ϵ_{pe}). Hence, the security Games and losses related to this attack scenario are removed from the proofs.

Other changes and corrections apply to:

- The INT-CTXT security experiment (cf. Figure 13).
- The computation of the AE nonces and keys (cf. Figure 1).
- The definition of the explicit authentication property (cf. Definition 8).
- The definition of the predicates FreshAuth, FreshKey, FreshIdentity (cf. Definitions 4, 5, 6).
- The addition of pseudo-codes describing the predicates FreshAuth, FreshKey, FreshIdentity (cf. Figures 7, 8, 9).
- Some reasoning and bounds in the proofs of explicit authentication, key indistinguishability and identity privacy (cf. Sections A.2, A.3, A.4).
- Some wording.