On Post-Quantum Cryptography Authentication for Quantum Key Distribution

Juan Antonio Vieira Giestinhas[1]*, Timothy Spiller[1]

[1] University of York, YO10 5DD, York, United Kingdom

Abstract

The traditional way for a Quantum Key Distribution (QKD) user to join a quantum network is by authenticating themselves using pre-shared key material. While this approach is sufficient for small-scale networks, it becomes impractical as the network grows, due to the total quadratic increase in the number of pre-shared keys required. To address this scalability issue, Public Key Infrastructure (PKI) combined with Post-Quantum Cryptography (PQC) offers a more scalable solution, allowing users to authenticate the QKD traffic remotely to obtain information-theoretical secure (ITS) keys under the presented assumptions. Unlike traditional PKI, which relies on classical cryptographic algorithms such as RSA, the approach presented in this paper leverages PQC algorithms that are believed to be resistant to quantum attacks. Similarly to the SIGMA or TLS protocols, authentication, confidentiality, and integrity are achievable against bounded adversaries to ensure secure and scalable quantum networks.

1 Introduction

The development of quantum computers is becoming an increasing threat to the most commonly used asymmetric algorithms, such as Rivest-Shamir-Adleman (RSA) or Elliptic-Curve Cryptography (ECC) [1]. With the advent of Shor's algorithm [2], the feasibility of retrospective decryption (harvest now, decrypt later) could become possible sooner than expected, posing risks to sensitive data.

On one hand, Quantum Cryptography, notably Quantum Key Distribution (QKD) [3–5], can be used to distribute secret keys in an Information-Theoretical Secure (ITS) manner, with the condition that an authenticated channel is available. The classical channel for QKD is typically authenticated with pre-shared keys and ITS Message Authentication Codes (MAC) [6, 7]. However, this approach is not convenient as it forces honest parties to physically meet each other.

On the other hand, Post-Quantum Cryptography (PQC) is another alternative to asymmetric cryptography that is believed to be secure against quantum and classical algorithms. By combining both technologies, ITS keys given by QKD are obtainable through a remote classical channel authenticated by PQC. Once the first QKD iteration authenticated by PQC is effected, the shared keys thus obtained may be used to perform ITS authentication in further QKD iterations.

*Correspondence: qph504@york.ac.uk

This paper focuses on non-ITS authentication for QKD, specifically relying on PQC and Public Key Infrastructure (PKI), notably with certificates given by a Certificate Authority (CA), where parties do not have pre-shared key material to authenticate the classical channel, which is required to distill an authenticated shared secret key from a QKD protocol. A general case of authentication for QKD is presented. Two examples of PQC protocols are presented, a signature-based authentication protocol, and an alternative scheme using Key Encapsulation Mechanisms (KEM) for authentication without signatures. Furthermore, a shared-based key authentication example utilizing non-ITS Message Authentication codes (MAC) is also discussed. A bounded adversary is defined and considered, alongside a conditioned unbounded adversary, to check the ITS security of the final QKD keys.

The structure and strategy of this paper are as follows. Section 2 summarizes current work to date, along with further motivation on the need for and approaches to authentication for QKD. The strategy adopted in this paper is to present formal technical descriptions of various non-ITS authentication protocols (two that leverage PQC in section 5, and use of pre-shared key in section 6). This strategy must be built upon formal definitions of numerous entities, procedures, properties and quantities, which are provided in section 3. As the protocols leverage PQC, the required PQC is discussed in section 4. Sections 5 and 6 then present the non-ITS authentication protocols, as formal, time-ordered lists of operations, each accompanied by commentary and discussion. This formal approach provides numerous important consequences.

Examples, that don't heavily influence this work, are that it is straightforward to evaluate the time taken (in information round trip units) and computational effort required for one round of each protocol. Furthermore, it is straightforward to convert the protocol to a flow diagram, identifying the points where an honest party would abort, having determined they are interacting with a dishonest impostor. Such a flow diagram also enables careful development of pseudo-code and thus actual computer code, to facilitate a real implementation of the protocol. Such code-development should be undertaken formally and carefully, to avoid introducing information-leakage vulnerabilities into the actual implementation of what is a fundamentally secure protocol.

For this work, however, the really important consequence from a formal presentation of the authentication protocols is security analysis. Such security analysis can be undertaken through investigation of a game, or experiment, between a challenger and two precisely specified adversaries. The outcome of the game informs on the security of the protocol. This scenario requires formal definition and establishment of a key exchange framework, which is presented in section 7. Utilizing this framework, formal security analysis of the protocols is then presented in section 8.

Discussion of the security proof results along with a further specific example is given in section 9, followed by conclusions delivered in section 10.

2 Motivation, related literature and contribution

PQC signature-based, KEM-based, or even combined protocols have already been implemented to specifically authenticate a classical channel to effectively perform QKD. Benchmark parameters with standardized or variants of the National Institute of Standards and Technology (NIST) PQC algorithms [8], along with implemented or simulated QKD links, have been estimated in [9–11]. The basic idea underpinning these papers, and for this work, is to perform unauthenticated QKD and authenticate the post-processing information at some point after distilling QKD key material using PQC algorithms. An authenticated key exchange protocol with a security proof based on the “short-term” security of PQC and “long-term” security of QKD is also found in [12] and is strongly related to the implementation [10].

Care must be taken when authenticating keys with classical algorithms because impersonation or misbinding attacks may occur, as stated in the SIGMA paper [13]. This is why the PQC-based authentication protocols presented in this paper follow the SIGMA structure: computation of an unauthenticated ephemeral key, authentication (signature in SIGMA), and then binding of secrets with honest entities (MAC in SIGMA) using cryptographic primitives. This approach not only helps to avoid person-in-the-middle and misbinding attacks, but also provides the entity protection feature that SIGMA offers. By following the SIGMA and TLS structures [14], non-ITS but secure authentication, confidentiality and integrity for QKD is achieved. If QKD is achieved then ITS keys are obtainable.

A similar approach to combine PQC and QKD, proposed in the hybrid authenticated key exchange (HAKE) protocol Muckle series [15–18], is to mix the different key material to obtain a final authenticated hybrid key, where its security holds as long as one of the cryptographic primitives, classical, PQC or QKD, remains secure. The goal of this work is to discuss the approach where PQC is used to authenticate QKD, whilst also being able to obtain traffic secrets, whereas in the Muckle series, the goal is to achieve redundancy by applying different cryptographic primitives that present different strengths. However, by including the public QKD post-processing information within some Muckle steps, QKD may also be authenticated by following a protocol similar to the Muckle series. In fact, the protocols to authenticate QKD within this work are strongly inspired from the Muckle series and the KEMTLS protocol [19], in addition to the SIGMA and TLS protocols.

This work addresses the separate authentication requirement that QKD presents, particularly in remote scenarios. While pre-sharing keys to authenticate is not always convenient, PQC and PKI offers a scalable solution for establishing secure QKD networks remotely. The idea is to apply the PQC approach only in the first QKD iteration, as subsequent iterations can be authenticated with QKD key material obtained in prior iterations. However, the presented PQC protocols can be reiterated for different QKD protocols, as derived QKD keys and secret state parameters can be transferred from one protocol to another securely while taking into account the failure probability (equivalently the security) of said parameters. Taking inspiration from the SIGMA protocol, the Muckle series and the KEMTLS protocol, the schemes presented in this paper enable QKD protocols to securely distill a secret key through a classical channel, authenticated due to the “short-term” security for PQC algorithms assumption, defined in section 4. Additionally, confidentiality and integrity over the classical traffic, including information of the entities, is provided. The general case of mono-authentication for QKD is covered, with specific examples similar to those in related literature. The traffic protection feature is also discussed: this feature holds against bounded adversaries, even if it is assumed that the PQC secret keys are revealed to the public after the first QKD instantiation is finalized. The security of the presented protocols is evaluated using the HAKE framework introduced and reused in the Muckle series [15–18].

3 Definitions/Preliminaries

This section provides the required definitions that are used in the security proofs of this work. The bounded and conditioned unbounded adversaries considered, Quantum Key Distribution, dual pseudorandom function, a hash function, signature, key encapsulation mechanism, and message authentication code are all defined. Some of these definitions are accompanied by game-based security models, named experiments, which highlight the security feature of interest for these primitives.

Definition 1. (Hybrid Polynomial-Time Adversary). The bounded adversary considered within this work is both Probabilistic Polynomial-Time (PPT) and Quantum Polynomial-Time (QPT), making it Hybrid Polynomial-Time (HPT).

A PPT and QPT adversary represents an adversary with both classical and quantum bounded computational resources. Given an input of size n, a constant C_{PPT} for the classical computations, and a constant C_{QPT} for the quantum computations, the time complexity of this adversary is $\mathrm{O}(\mathrm{n}^{C_{\mathrm{PPT}}})$ for classical computations and $\mathrm{O}(\mathrm{n}^{C_{\mathrm{QPT}}})$ for quantum computations.

Since the considered adversary is both PPT and QPT, the adversary is able to run in parallel both QPT and PPT algorithms to solve a specific problem, if possible. If a specific problem can be solved in polynomial time by relying only on PPT or QPT algorithms independently, the time complexity of the adversary is the minimum between $\mathrm{O}(\mathrm{n}^{C_{\mathrm{PPT}}})$ and $\mathrm{O}(\mathrm{n}^{C_{\mathrm{QPT}}})$.

Furthermore, the adversary is able to combine PPT algorithms with QPT algorithms to perform HPT algorithms. In this case, given an input of size n, a constant $C_{T_{\mathrm{PPT}}}$ for the pertinent classical computations, a constant $C_{T_{\mathrm{QPT}}}$ for the pertinent quantum computations, and a constant C_{parallel} for the classical and quantum algorithms that can run in a parallel way, the total time complexity of the adversary is the summation of the classical algorithm time complexity, $\mathrm{O}(\mathrm{n}^{C_{T_{\mathrm{PPT}}}})$, with the quantum algorithm time complexity, $\mathrm{O}(\mathrm{n}^{C_{T_{\mathrm{QPT}}}})$, minus the time complexity where classical and quantum algorithms can run at the same time, $\mathrm{O}(\mathrm{n}^{C_{\mathrm{parallel}}})$. The idea behind the HPT algorithm is to solve a problem in polynomial time more efficiently, in terms of time complexity, than if an adversary is limited to only classical or quantum algorithms with their corresponding, and independent, limited resources. That can be expressed as:

$$\mathrm{O}(\mathrm{n}^{C_{T_{\mathrm{PPT}}}}) + \mathrm{O}(\mathrm{n}^{C_{T_{\mathrm{QPT}}}}) - \mathrm{O}(\mathrm{n}^{C_{\mathrm{parallel}}}) < \min(\mathrm{O}(\mathrm{n}^{C_{\mathrm{PPT}}}), \mathrm{O}(\mathrm{n}^{C_{\mathrm{QPT}}}))$$

Note that the HPT adversary can solve problems in polynomial time that have greater than polynomial time complexity on the classical side but not in the quantum side, by running QPT algorithms (and vice versa with PPT algorithms). A practical example could be the factorization of a composite integer to break RSA using Shor's algorithm [2].

The HPT adversary is allowed to interact with the protocols presented in this paper at any time. Once the protocols are not in an active state (either because they successfully completed, or they aborted), the HPT adversary is allowed to ask an unbounded adversary to obtain the QKD keys with the condition that they were not ITS. This unbounded adversary is also limited (termed a conditioned unbounded adversary from now on), since the goal of having such a threat is to prove that the QKD keys are ITS once authentication is finalized. In other words, the conditioned unbounded adversary is hypothetical and is used as a tool in the security analysis to prove the ITS property of the final QKD keys.

Definition 2. (Conditioned Unbounded Adversary). An unbounded adversary is useful when defining ITS security. Unbounded adversaries have infinite resources, infinite computational power, and have access to ideal quantum and classical technologies, including ideal quantum memories.

Since the goal of this work is to prove that the protocols presented are secure against the bounded HPT adversary and to prove that the final authenticated keys are ITS, the considered conditioned unbounded adversary is allowed to have total control of the quantum channel used to perform unauthenticated QKD (Definition 5), is able to read the public traffic used to distill the unauthenticated QKD keys but is not allowed to interact with anything else, i.e. with the authentication steps. Note that this conditioned unbounded adversary could be allowed to actively modify the classical traffic used to distill the QKD keys from the unauthenticated QKD step; however, that would be detected by the honest parties during the authentication step, making that degree of freedom ineffective.

The HPT adversary can interact with the conditioned unbounded adversary for two purposes. The first one is to obtain ephemeral QKD keys in the case the HPT adversary is not the one performing QKD as an honest party. This can be requested at any time, starting from the moment that the unauthenticated QKD has ended. The second purpose is to check the ITS security of QKD keys that have been successfully distilled from finalized protocols or iterations. Once the protocols finish running, the unbounded adversary is allowed to interact with the HPT adversary in a limited way. If the conditioned unbounded adversary has access to the distilled QKD keys, compromised during the unauthenticated QKD steps, then these are revealed to the HPT adversary, making the QKD keys not ITS. Otherwise, if the unbounded adversary is unable to compromise the ITS keys with the allowed interactions, then the HPT adversary receives nothing and thus learns that the QKD keys are ITS.

Furthermore, the HPT adversary is allowed to send the gathered traffic from the authentication steps to the unbounded adversary. Note that since the primitives that provide authentication, confidentiality and integrity are not ITS, the unbounded adversary can trivially break those, given the classical traffic. That is why the unbounded adversary is not allowed to transmit the compromised secrets to the HPT adversary but only the non-ITS distilled QKD keys.

Note that if authentication fails against the HPT adversary, the distilled QKD keys are not secure, thus not ITS.

Definition 3. (Quantum Key Distribution). A Quantum Key Distribution (QKD) protocol uses the properties of quantum mechanics to generate a shared secret key between honest parties [3–5]. Properties of quantum mechanics exploited to achieve secure quantum communication include the no-cloning theorem [20], the Heisenberg uncertainty principle [21], and quantum entanglement, depending on the QKD protocol [22]. QKD is divided into two parts: the preparation, sending, and measurement of qubits (quantum side) and a post-processing part for information reconciliation (classical side), typically divided into sifting, parameter estimation, error correction, and privacy amplification. The quantum side information is transmitted through an insecure quantum channel whereas the classical side information is transmitted through an authenticated classical channel. The quantum side involves QPT algorithms, while the classical side involves PPT algorithms. Since both sides are polynomial-time, the overall QKD protocol can be seen as a HPT algorithm, QKD = {KeyGen}, with an associated key set $\mathcal{K} = \{0,1\}^*$ of arbitrary length. The KeyGen algorithm is described as:

- KeyGen(κ, $*$) $\rightarrow \{(K_{\mathrm{QKD}}, \mathrm{m}_{\mathrm{QKD}}), \perp\}$: The algorithm outputs the shared authenticated key K_{QKD} of arbitrary size $* > 0$ between honest parties, given a security parameter κ. For practical purposes, the corresponding authenticated public information $\mathrm{m}_{\mathrm{QKD}}$ used to distill the secret key K_{QKD} is also an output. If QKD is aborted, then $\perp$ is returned.

The abortion condition strongly depends on the specific QKD protocol. For discrete-variable QKD (DV-QKD), this condition is often determined by the quantum bit error rate (QBER), which is obtained after the honest parties compare some of the assumed correlated sent and measured bits using the authenticated classical channel [3, 5]. For continuous-variable QKD (CV-QKD), this condition often depends on the signal-to-noise ratio [4]. The abortion condition also depends on whether the honest parties detect a tampering by an adversary within the authenticated classical channel. If an abortion occurs, the secret parameters and any revealed information from the public channel are discarded and never used again. Note that in practice, QKD may run continuously without ever aborting. However, when the keys are not secure, the final size of K_{QKD} would be zero.

Although different QKD protocols employ various methods for quantum state preparation, measurement, and post-processing, they all produce an authenticated shared secret key and the corresponding authenticated public information: K_{QKD} and $\mathrm{m}_{\mathrm{QKD}}$ are QKD protocol-agnostic.

A QKD iteration starts with honest parties exchanging and measuring quantum signals according to some agreed protocol and ends once privacy amplification is applied to their reconciled key, regardless of how many quantum signals, base sifting, parameter estimation, or other post-processing step but privacy amplification, are done in between.

Definition 4. **(Quantum Key Distribution Security).** The underlying problem behind the generation of the QKD keys relies on the bounds given by the laws of physics rather than the hardness of a mathematical problem, which makes the shared QKD secrets obtained from a QKD protocol between parties ITS from an ideal point of view. This means that the security of QKD does not depend on the computational power of the adversaries, implying that it is secure even against unbounded adversaries.

Ideally, an unbounded adversary $\mathcal{A}$ has no advantage in distinguishing between secrets distilled from a QKD protocol and random keys of the same length. The distinguishability game is defined with the experiment $\mathrm{Exp}^{\text{ind-theory}}_{\mathrm{QKD},\mathcal{A}}(*)$, illustrated in Experiment 1, where the unbounded adversary $\mathcal{A}$ is able to perform a query to the QKD oracle and execute any operation within the QKD oracle's quantum and authenticated classical channels, while the QKD oracle generates the QKD key.

```
$\mathrm{Exp}^{\text{ind-theory}}_{\mathrm{QKD},\mathcal{A}}(*)$:
    $b \xleftarrow{\$} \{0,1\}$
    if QKD does not abort:
        $(K_0, \mathrm{m}_{\mathrm{QKD}}) \leftarrow \mathrm{KeyGen}(*)$, $K_1 \xleftarrow{\$} \mathcal{K}$
    else:
        return $\perp \leftarrow \mathrm{KeyGen}(*)$
    $b' \leftarrow \mathcal{A}(K_b, \mathrm{m}_{\mathrm{QKD}})$
    if $b' = b$ return 1, else return 0
```

Experiment 1: Distinguishability security experiment for QKD.

Note that if the unbounded adversary tampers too much within the quantum channel or tampers within the authenticated classical channel, the QKD oracle will detect this and output the abort symbol. Continually tampering too much and making the QKD oracle to abort is equivalent to a denial of service (DoS) attack. Since the goal is to describe the security of QKD authenticated with PQC, DoS is not treated in this work.

In the ideal QKD scenario with ideal devices, the advantage of an unbounded adversary $\mathcal{A}$ to win the Experiment 1 is zero:

$$\text{Adv}^{\text{ind-theory}}_{\text{QKD},\mathcal{A}}(*) = \left| Pr[\text{Exp}^{\text{ind-theory}}_{\text{QKD},\mathcal{A}}(*) = 1] - \frac{1}{2} \right| = 0$$

However, the imperfections in implemented devices, the practical insecurity of the quantum channel and the information leakage and/or limitations from the usual implemented post-processing steps of QKD introduce a negligible exploitable advantage $\varepsilon_{QKD}(\kappa)$ [23, 24], which varies according to the considered specific QKD protocol, post-processing methods and the desired security level κ. In the same way, the authentication of the classical channel introduces a negligible exploitable advantage $\varepsilon_{\text{auth}}$. The authentication can be ITS, giving a $\varepsilon^{\text{ITS}}_{\text{auth}}$ security [25], or non-ITS but conditionally secure, giving a $\varepsilon^{\text{PQC}}_{\text{auth}}$ security.

For the case where an unbounded adversary is present when ITS authentication is performed, QKD is secure if the unbounded adversary has a negligible advantage to win the experiment $\text{Exp}^{\text{ind}}_{\text{QKD},\mathcal{A}}(\kappa, *)$, defined in the same way as the experiment 1, a result proved in [26]:

$$\text{Adv}^{\text{ind}}_{\text{QKD},\mathcal{A}}(\kappa, *) = \left| Pr[\text{Exp}^{\text{ind}}_{\text{QKD},\mathcal{A}}(\kappa, *) = 1] - \frac{1}{2} \right| \leq \varepsilon_{QKD}(\kappa) + \varepsilon^{\text{ITS}}_{\text{auth}}(\kappa)$$

Note that the unbounded adversary wins all the times, i.e. $\text{Adv}^{\text{ind}}_{\text{QKD},\mathcal{A}}(\kappa, *) = \frac{1}{2}$, if non-ITS authentication is performed, by realizing impersonation attacks.

In the same way, QKD is secure if all bounded HPT adversaries have a negligible advantage to win the experiment $\text{Exp}^{\text{ind}}_{\text{QKD},\mathcal{A}}(\kappa, *)$, defined in the same way as the experiment 1:

$$\text{Adv}^{\text{ind}}_{\text{QKD},\mathcal{A}}(\kappa, *) = \left| Pr[\text{Exp}^{\text{ind}}_{\text{QKD},\mathcal{A}}(\kappa, *) = 1] - \frac{1}{2} \right| \leq \varepsilon_{QKD}(\kappa) + \varepsilon_{\text{auth}}(\kappa)$$

Where $\varepsilon_{\text{auth}}(\kappa) \in \{\varepsilon^{\text{ITS}}_{\text{auth}}(\kappa), \varepsilon^{\text{PQC}}_{\text{auth}}(\kappa)\}$, and the non-ITS, but post-quantum, authentication holds as discussed in section 4.

For the rest of this paper, ε_{QKD} and $\varepsilon_{\text{auth}}$ depend implicitly on an arbitrary security parameter κ, i.e. $\varepsilon_{QKD} := \varepsilon_{QKD}(\kappa)$ and $\varepsilon_{\text{auth}} = \varepsilon_{\text{auth}}(\kappa)$.

To give a physical interpretation to the epsilons, the probability that the practical ITS QKD keys have not been compromised by any adversary is upper bounded by $(\varepsilon_{QKD} + \varepsilon_{\text{auth}})$, or in other words, the probability they are not secure as intended is bounded by $(\varepsilon_{\text{QKD}} + \varepsilon_{\text{auth}})$. A successful QKD protocol and the derived keys are typically assumed to have a security proportional to $(\varepsilon_{QKD} + \varepsilon_{\text{auth}})$-secure.

Definition 5. (Unauthenticated Quantum Key Distribution). Unauthenticated QKD is defined as the same as QKD from Definition 3 but the obtained shared key is distilled between two, not necessarily honest, parties through an insecure, unauthenticated, classical channel. This implies that the unauthenticated QKD key obtained from KeyGen, K_{QKD}, is not secure unless the corresponding public information m_{QKD} is authenticated in further steps by the honest parties, if they are indeed both honest.

A conditioned unbounded adversary $\mathcal{A}$ is allowed to interact with the unauthenticated QKD, without targeting the authentication property of the classical channel (Definition 2). Let

$\mathrm{Exp}^{\text{ind-theory}}_{\text{unauth-QKD},\mathcal{A}}(\kappa, *)$ be the same Experiment 1 but this time the unbounded adversary is not allowed to target the authentication of the classical channel from the QKD oracle. Since authentication is not targeted, the QKD oracle can be replaced by an unauthenticated QKD oracle without changing the advantage of the adversary to win the distinguishability game. The advantage of the conditioned unbounded adversary $\mathcal{A}$ to win $\mathrm{Exp}^{\text{ind-theory}}_{\text{unauth-QKD},\mathcal{A}}(\kappa, *)$ is the already defined $\varepsilon_{QKD}(\kappa)$.

Definition 6. (From unauthenticated QKD to authenticated QKD). Let two parties perform unauthenticated, and $\varepsilon_{\mathrm{QKD}}$-secure, QKD to obtain an insecure shared key K_{QKD} and the corresponding public information $\mathrm{m}_{\mathrm{QKD}}$ used to distill K_{QKD}. If the transcript $\mathrm{m}_{\mathrm{QKD}}$ is authenticated by honest parties in further steps using an $\varepsilon_{\mathrm{auth}}$-secure composable authentication scheme, the unauthenticated QKD becomes equivalent to QKD, defined as in Definition 3, making the shared key, K_{QKD}, ITS and $(\varepsilon_{\mathrm{QKD}} + \varepsilon_{\mathrm{auth}})$-secure [27].

Assuming that the distilled and authenticated QKD key K_{QKD} has length L, K_{QKD} can be partitioned into n smaller keys of arbitrary size l_i with $L \geq \sum_{i=1}^{n} l_i$, such as $K_{\mathrm{QKD}} = ss_{\mathrm{QKD}_1}||\dots||ss_{\mathrm{QKD}_n}||ss_{rest}$ where $||$ denotes concatenation (and ss is short for shared secret), the length $|ss_{\mathrm{QKD}_i}| = l_i$ for $i = 1, ..., n$, and $|ss_{rest}| = L - \sum_{i=1}^{n} l_i$. Note that if any part of K_{QKD}, such as ss_{QKD_i}, is used for cryptographic purposes (e.g., renewing authentication material), that portion should not be reused for other cryptographic tasks, to avoid accidental or unexpected leakage of the secret. The remaining unused key material remains ITS and $(\varepsilon_{\mathrm{QKD}} + \varepsilon_{\mathrm{auth}})$-secure. The privacy amplification step, executed by all QKD protocols at the end, produces keys that are both uniformly random and independent with probability upper-bounded by $\varepsilon_{\mathrm{QKD}}$, typically following the leftover hash lemma [28]. Privacy amplification reduces any correlation between the key bits, as well as between the key bits and the adversary's potential information, to a negligible level, which is quantified inside ε_{QKD}. The combination of privacy amplification with an epsilon composable authentication scheme makes the final keys security upper-bounded by $(\varepsilon_{QKD} + \varepsilon_{\mathrm{auth}})$, where $\varepsilon_{QKD}(\kappa)$ includes the probability of failure given by the privacy amplification step and the security parameter κ. For example, let $K_{\mathrm{QKD}} = ss_{\mathrm{QKD}}||ss_{\mathrm{rest}}$, where ss_{QKD} is used for authentication with HPT secure primitives. Once the authentication is securely implemented, since the primitives used are HPT, an unbounded adversary could compromise ss_{QKD}, making the key non-ITS once used. However, the remaining part, ss_{rest}, remains ITS and $(\varepsilon_{QKD} + \varepsilon_{\mathrm{auth}})$-secure, as the individual bits from the QKD protocols are indistinguishable from truly random bits with at least $1 - \varepsilon_{QKD} - \varepsilon_{\mathrm{auth}}$ probability.

There could be instances where a single QKD iteration does not produce enough key material to run the HPT primitives within the protocols presented in sections 5 and 6. The concatenation of different QKD keys obtained through different QKD iterations or QKD protocols is possible. However, this implies that the security of the new concatenated key has also a new upper bound. Let K be the concatenation of "m" QKD keys, $K = K_1||K_2||\dots||K_m$. By Definition 4, each QKD key "i", $i \in \{0, 1, \dots, m\}$, is ε_i-secure. This implies that the concatenated key K is $(1 - \Pi_{i=1}^{m}(1 - \varepsilon_i))$-secure, which is upper bounded by $\Sigma_{i=1}^{m}\varepsilon_i$.

For simplicity, the epsilon security of any QKD step within the protocols presented in this paper is assumed to upper-bounded by the same value $\varepsilon_{QKD}(\kappa)$, dependent on an arbitrary security parameter κ.

Next, the definition for dual pseudorandom function, signature, message authentication code, and key encapsulation mechanism and their corresponding securities are presented, similarly to within the Muckle series [15–18].

Definition 7. (Dual Pseudorandom Function). The pseudorandom functions (PRFs) used in this work are assumed to be dual PRFs. That is:

For a PPT pseudorandom function PRF: $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{R}$ there exists the dual PPT PRF^{dual} defined as PRF^{dual}: $\mathcal{M} \times \mathcal{K} \rightarrow \mathcal{R}$ where PRF(k, m) = PRF^{dual}(m, k) for any bit string m $\in \mathcal{M}$ and key k $\in \mathcal{K}$. For a given security parameter κ, the key set $\mathcal{R}$ corresponds to the random output set $\{0,1\}^{\kappa}$. The sets $\mathcal{M}$ and $\mathcal{K}$ each separately depend on κ.

The security of the PRFs and their dual counterpart are based on the distinguishing game. For a HPT adversary $\mathcal{A}$ and security parameter κ, the corresponding advantage functions are defined as:

$$\text{Adv}^{\text{ind}}_{\text{PRF},\mathcal{A}}(\kappa) = \left| Pr[\text{Exp}^{\text{ind}}_{\text{PRF},\mathcal{A}}(\kappa) = 1] - \frac{1}{2} \right|$$

$$\text{Adv}^{\text{ind}}_{\text{PRF}^{\text{dual}},\mathcal{A}}(\kappa) = \left| Pr[\text{Exp}^{\text{ind}}_{\text{PRF}^{\text{dual}},\mathcal{A}}(\kappa) = 1] - \frac{1}{2} \right|$$

where the experiments $\text{Exp}^{\text{ind}}_{\text{PRF},\mathcal{A}}(\kappa)$ and $\text{Exp}^{\text{ind}}_{\text{PRF}^{\text{dual}},\mathcal{A}}(\kappa)$ are defined in Experiment 2, such that the HPT adversary $\mathcal{A}$ is able to perform a polynomial number of queries q to the PRF or PRF^{dual} oracles.

```
$\text{Exp}^{\text{ind}}_{\text{PRF}^*,\mathcal{A}}(\kappa)$:
    $b \xleftarrow{\$} \{0,1\}$
    k $\xleftarrow{\$} \mathcal{K}$
    for i $\leftarrow$ 1 to q:
        $\text{m}_i \leftarrow \mathcal{A}(r_1, \ldots, r_{i-1}) \in \mathcal{M}$
        if * = dual $\text{r}_{0_i} \leftarrow \text{PRF}^{\text{dual}}(\text{m}_i, \text{k})$, else $\text{r}_{0_i} \leftarrow \text{PRF}(\text{k}, \text{m}_i)$
        $r_{1_i} \xleftarrow{\$} \mathcal{R} \in \{0,1\}^{\kappa}$
    $b' \leftarrow \mathcal{A}(r_{b_1}, \ldots, r_{b_q})$
    if $b' = b$ return 1, else return 0
```

Experiment 2: Distinguishability security experiment for PRF or PRF^{dual}.

The pseudorandom functions PRF and PRF^{dual} are PPT, QPT or HPT secure if for all respective PPT, QPT or HPT adversaries $\mathcal{A}$, the advantages $\text{Adv}^{\text{ind}}_{\text{PRF},\mathcal{A}}(\kappa)$ and $\text{Adv}^{\text{ind}}_{\text{PRF}^{\text{dual}},\mathcal{A}}(\kappa)$ are negligible in the security parameter κ. That is, $\text{Adv}^{\text{ind}}_{\text{PRF},\mathcal{A}}(\kappa) \leq \text{negl}_{\text{PRF}}(\kappa)$ and $\text{Adv}^{\text{ind}}_{\text{PRF}^{\text{dual}},\mathcal{A}}(\kappa)$ $\leq \text{negl}_{\text{PRF}^{\text{dual}}(\kappa)}$, where $\text{negl}(\kappa)$ is a function that takes as input the security parameter κ and defined as: there exists $\kappa > \kappa_0$ such that $\text{negl}(\kappa_0) < \frac{1}{p(\kappa_0)}$ for every positive polynomial p.

Definition 8. (Hash function). The hash function H of this paper is assumed to be at least weakly collision resistant. That is:

Given a known bit string $m \in \mathcal{M}$, the PPT hash function H: $\mathcal{M} \rightarrow \mathcal{D}$, and the corresponding digest $\text{H}(m) \in \mathcal{D}$, an HPT adversary has negligible advantage $\text{Adv}_{\text{H},\mathcal{A}}(\kappa) \leq \text{negl}_H(\kappa)$ of finding $m' \in \mathcal{M}$ such that $\text{H}(m) = \text{H}(m')$, given a security parameter κ.

Definition 9. (Signature - Σ). Signatures are composed by three PPT algorithms Σ = {KeyGen, Sign, Ver}, with an associated bit string set $\mathcal{M}$, defined as:

- KeyGen(κ) $\rightarrow$ (sk, pk): The algorithm takes as input a security parameter κ and outputs a secret key sk with the corresponding public key pk.
- Sign(sk, m) $\rightarrow \sigma$: The algorithm takes as input a secret key sk and a bit string m $\in \mathcal{M}$ and outputs a signature tag σ.
- Ver(pk, m, σ) $\rightarrow \{0,1\}$: The algorithm takes as input a public key pk, a bit string m $\in \mathcal{M}$ and a signature tag σ and outputs a bit $\in \{0,1\}$.

Σ is correct if for all (sk, pk) $\leftarrow$ KeyGen(κ) and $\sigma \leftarrow$ Sign(sk, m) given any security parameter $\kappa \in \mathbb{N}$ and bit string m $\in \mathcal{M}$, then Ver(pk, m, σ) = 1:

$$\Pr\left[\mathrm{Ver}(\mathrm{pk}, \mathrm{m}, \mathrm{Sign}(\mathrm{sk},\mathrm{m})) = 1\right] = 1$$

Definition 10. (EUF-CMA security of Σ). For a HPT adversary $\mathcal{A}$, the advantage function for existential unforgeability under chosen message attacks (EUF-CMA) of Σ is:

$$\mathrm{Adv}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa) = \Pr\left[\mathrm{Exp}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa) = 1\right]$$

Where the experiment $\mathrm{Exp}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa)$ is defined in Experiment 3.

```
$\mathrm{Exp}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa)$:
    (sk, pk) $\leftarrow$ KeyGen($\kappa$), $\mathcal{Q} \leftarrow \emptyset$
    for i $\leftarrow$ 1 to q:
        $\mathrm{m}_i \leftarrow \mathcal{A}(\mathrm{pk}, \sigma_1, \ldots, \sigma_{i-1}) \in \mathcal{M}$
        $\sigma_i \leftarrow$ Sign(sk, $\mathrm{m}_i$), $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathrm{m}_i\}$
    (m*, $\sigma$*) $\leftarrow \mathcal{A}(\mathrm{pk}, \sigma_1, \ldots, \sigma_q)$
    if Ver(pk, m*, $\sigma$*) = 1 AND m* $\notin \mathcal{Q}$ return 1, else return 0
```

Experiment 3: EUF-CMA security experiment for signature Σ.

vThe signature scheme Σ is PPT, QPT or HPT EUF-CMA secure if for all respective PPT, QPT or HPT adversaries $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa)$ is negligible in the security parameter κ. As a formal inequality, $\mathrm{Adv}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa) \leq \mathrm{negl}_{\Sigma,\text{EUF-CMA}}(\kappa)$.

Definition 11. (Key Encapsulation Mechanism - KEM). Key Encapsulation Mechanisms (KEM) are composed by three PPT algorithms KEM = {KeyGen, Enc, Dec}, with an associated key set $\mathcal{K}$, defined as:

- KeyGen(κ) $\rightarrow$ (sk, pk): The algorithm takes as input a security parameter κ and outputs a secret key sk with the corresponding public key pk.
- Enc(pk) $\rightarrow$ (k, c): The algorithm takes as input a public key pk and outputs a key k $\in \mathcal{K}$ and the corresponding encapsulation ciphertext c.
- Dec(sk, c) $\rightarrow$ {k, $\perp$}: The algorithm takes as input a secret key sk and a ciphertext c and outputs the decapsulated k $\in \mathcal{K}$ or $\perp$ in case of failure.

KEM is correct if for all (sk, pk) $\leftarrow$ KenGen(κ) and (k, c) $\leftarrow$ Enc(pk) given any security parameter $\kappa \in \mathbb{N}$, then k $\leftarrow$ Dec(sk, c):

$$\Pr[\mathrm{Dec}(\mathrm{sk},\ \mathrm{c}) = \mathrm{k}] = 1$$

Definition 12. (IND-CPA and IND-CCA security of KEM). For a HPT adversary $\mathcal{A}$, the advantage functions for indistinguishability under chosen-ciphertext attacks (IND-CCA) and for indistinguishability under chosen-plaintext attacks (IND-CPA) of KEM are:

$$\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) = \left|\Pr\left[\mathrm{Exp}^{\text{IND-CPA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) = 1\right] - \frac{1}{2}\right|$$

$$\mathrm{Adv}^{\text{IND-CCA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) = \left|\Pr\left[\mathrm{Exp}^{\text{IND-CCA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) = 1\right] - \frac{1}{2}\right|$$

where the experiments $\mathrm{Exp}^{\text{IND-CPA}}_{\mathrm{KEM},\mathcal{A}}(\kappa)$ and $\mathrm{Exp}^{\text{IND-CCA}}_{\mathrm{KEM},\mathcal{A}}(\kappa)$ are defined in Experiment 4.

```
$\mathrm{Exp}^{\text{IND-*}}_{\mathrm{KEM},\mathcal{A}}(\kappa)$:
    $(\mathrm{sk}, \mathrm{pk}) \leftarrow \mathrm{KeyGen}(\kappa)$, $\mathcal{Q} \leftarrow \emptyset$, $b \xleftarrow{\$} \{0,1\}$
    for $\mathrm{i} \leftarrow 1$ to $\mathrm{q}$:
        $\mathrm{k}_i \leftarrow \mathcal{A}(\mathrm{pk}, \mathrm{c}_1, \dots, \mathrm{c}_{i-1}) \in \mathcal{K} := \{0,1\}^\kappa$
        $\mathrm{c}_i \leftarrow \mathrm{Enc}(\mathrm{pk}, \mathrm{k}_i)$, $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathrm{c}_i\}$
    $\mathrm{k}^*_0, \mathrm{k}^*_1 \leftarrow \mathcal{A}(\mathrm{c}_1, \dots, \mathrm{c}_q) \in \mathcal{K}$
    $\mathrm{c}^* \leftarrow \mathrm{Enc}(\mathrm{pk}, \mathrm{k}_b)$
    if $* = \mathrm{CCA}$:
        for $\mathrm{i} \leftarrow 1$ to $\mathrm{q}'$:
            $\mathrm{c}'_i \leftarrow \mathcal{A}(\mathrm{pk}, \mathrm{c}_1, \dots, \mathrm{c}_q, \mathrm{c}^*, \mathrm{k}'_1, \dots, \mathrm{k}'_{i-1})$
            $\mathrm{k}'_i \leftarrow \mathrm{Dec}(\mathrm{sk}, \mathrm{c}'_i)$, $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathrm{c}'_i\}$
        $b' \leftarrow \mathcal{A}(\mathrm{pk}, \mathrm{c}_1, \dots, \mathrm{c}_q, \mathrm{c}^*, \mathrm{k}'_1, \dots, \mathrm{k}'_{\mathrm{q}'})$
    else:
        $b' \leftarrow \mathcal{A}(\mathrm{pk}, \mathrm{c}_1, \dots, \mathrm{c}_q, \mathrm{c}^*)$
    if $b' = b$ AND $\mathrm{c}^* \notin \mathcal{Q}$ return 1, else return 0
```

Experiment 4: IND-CPA and IND-CCA security experiments for KEM.

The KEM scheme is PPT, QPT or HPT IND-CPA and/or IND-CCA secure if for all respective PPT, QPT or HPT adversaries $\mathcal{A}$, the advantages $\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{KEM},\mathcal{A}}(\kappa)$ and/or $\mathrm{Adv}^{\text{IND-CCA}}_{\mathrm{KEM},\mathcal{A}}(\kappa)$ are negligible in the security parameter κ. As a formal inequality, $\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) \leq \mathrm{negl}_{\mathrm{KEM},\text{IND-CPA}}(\kappa)$ and/or $\mathrm{Adv}^{\text{IND-CCA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) \leq \mathrm{negl}_{\mathrm{KEM},\text{IND-CCA}}(\kappa)$.

The KEM CPA security is not used for the rest of this paper and only the CCA security is relevant. However, the experiment is presented because Definition 16 makes use of it.

Definition 13. (Message Authentication Codes - MACs). MACs are composed by three PPT algorithms MAC = {KeyGen, Auth, Ver}, with an associated key set $\mathcal{K}$ and bit string set $\mathcal{M}$, defined as:

- $\mathrm{KeyGen}(\kappa) \to \mathrm{sk}$: The algorithm takes as input a security parameter κ and outputs a secret key $\mathrm{sk} \in \mathcal{K} := \{0,1\}^\kappa$.
- $\mathrm{Auth}(\mathrm{sk}, \mathrm{m}) \to \tau$: The algorithm takes as input a secret key $\mathrm{sk} \in \mathcal{K}$ and a bit string $\mathrm{m} \in \mathcal{M}$ and outputs an authentication tag τ.
- $\mathrm{Ver}(\mathrm{sk}, \mathrm{m}, \tau) \to \{0,1\}$: The algorithm takes as input a secret key $\mathrm{sk} \in \mathcal{K}$, a bit string $\mathrm{m} \in \mathcal{M}$ and an authentication tag τ and outputs a bit $\in \{0,1\}$.

MAC is correct if for all $(\mathrm{sk}, \mathrm{pk}) \leftarrow \mathrm{KeyGen}(\kappa)$ and $\tau \leftarrow \mathrm{Auth}(\mathrm{sk}, \mathrm{m})$ given any security parameter $\kappa \in \mathbb{N}$ and bit string $\mathrm{m} \in \mathcal{M}$, then $\mathrm{Ver}(\mathrm{sk}, \mathrm{m}\ \tau) = 1$:

$$\Pr\left[\mathrm{Ver}(\mathrm{sk},\ \mathrm{m},\ \mathrm{Auth}(\mathrm{sk},\mathrm{m})) = 1\right] = 1$$

Definition 14. (EUF-CMA security of MAC). For a HPT adversary $\mathcal{A}$, the advantage function for existential unforgeability under chosen message attacks (EUF-CMA) of MAC is:

$$\mathrm{Adv}^{\text{EUF-CMA}}_{\text{MAC},\mathcal{A}}(\kappa) = \Pr\left[\mathrm{Exp}^{\text{EUF-CMA}}_{\text{MAC},\mathcal{A}}(\kappa) = 1\right]$$

Where the experiment $\mathrm{Exp}^{\text{EUF-CMA}}_{\text{MAC},\mathcal{A}}(\kappa)$ is defined in Experiment 5.

```
$\mathrm{Exp}^{\text{EUF-CMA}}_{\text{MAC},\mathcal{A}}(\kappa)$:
    sk $\leftarrow$ KeyGen($\kappa$), $\mathcal{Q} \leftarrow \emptyset$
    for i $\leftarrow$ 1 to q:
        $\mathrm{m}_i \leftarrow \mathcal{A}(\tau_1, \ldots, \tau_{i-1}) \in \mathcal{M}$
        $\tau_i \leftarrow$ Auth(sk, $\mathrm{m}_i$), $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\mathrm{m}_i\}$
    ($\mathrm{m}^*$, $\tau^*$) $\leftarrow \mathcal{A}(\tau_1, \ldots, \tau_q)$
    if Ver(sk, $\mathrm{m}^*$, $\tau^*$) = 1 AND $\mathrm{m}^* \notin \mathcal{Q}$ return 1, else return 0
```

Experiment 5: EUF-CMA security experiment for MAC.

The MAC scheme is PPT, QPT or HPT EUF-CMA secure if for all respective PPT, QPT or HPT adversaries $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{EUF-CMA}}_{\text{MAC},\mathcal{A}}(\kappa)$ is negligible in the security parameter κ. As a formal inequality, $\mathrm{Adv}^{\text{EUF-CMA}}_{\text{MAC},\mathcal{A}}(\kappa) \leq \mathrm{negl}_{\text{MAC,EUF-CMA}}(\kappa)$.

An ITS MAC is a MAC that is secure for all unbounded adversaries $\mathcal{A}$: the advantage $\mathrm{Adv}^{\text{EUF-CMA}}_{\text{MAC},\mathcal{A}}(\kappa)$ is negligible in the security parameter κ.

Definition 15. (Authenticated Encryption with Associated Data - AEAD). An AEAD scheme is composed of three PPT algorithms AEAD = {KeyGen, Enc, Dec}, with an associated key space $\mathcal{K}$, message space $\mathcal{M}$, associated data space $\mathcal{AD}$, and ciphertext space $\mathcal{C}$, defined as follows:

- KeyGen(κ) $\rightarrow$ k: The algorithm takes as input a security parameter κ and outputs a key k $\in \mathcal{K}$ chosen uniformly at random.
- Enc(k, A, m) $\rightarrow$ c: The encryption algorithm takes as input a key k $\in \mathcal{K}$, associated data A $\in \mathcal{AD}$, and a message m $\in \mathcal{M}$, and outputs a ciphertext c $\in \mathcal{C}$.
- Dec(k, A, c) $\rightarrow$ {m, $\perp$}: The decryption algorithm takes as input a key k $\in \mathcal{K}$, associated data A $\in \mathcal{AD}$, and a ciphertext c $\in \mathcal{C}$, and outputs the corresponding message m $\in \mathcal{M}$ if c is valid, $\perp$ otherwise.

The AEAD scheme is said to be correct if for all keys k output by KeyGen(κ), for all m $\in \mathcal{M}$ and A $\in \mathcal{AD}$, it holds that:

$$\mathrm{Dec}(\mathrm{k},\ \mathrm{A},\ \mathrm{Enc}(\mathrm{k},\ \mathrm{A},\ \mathrm{m})) = \mathrm{m}$$

Definition 16. (IND-CPA and INT-CTXT Security of AEAD). For any HPT adversary $\mathcal{A}$, the advantage functions for indistinguishability under chosen plaintext attacks (IND-CPA) and for ciphertext integrity (INT-CTXT) of an AEAD scheme are defined as:

$$\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) = \left|\Pr\left[\mathrm{Exp}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) = 1\right] - \frac{1}{2}\right|$$

$$\mathrm{Adv}^{\text{INT-CTXT}}_{\text{AEAD},\mathcal{A}}(\kappa) = \Pr\left[\mathrm{Exp}^{\text{INT-CTXT}}_{\text{AEAD},\mathcal{A}}(\kappa) = 1\right]$$

The experiment $\mathrm{Exp}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa)$ is the same as Experiment 4 but there is no public key involved. The $\mathrm{Exp}^{\text{INT-CTXT}}_{\text{AEAD},\mathcal{A}}(\kappa)$ Experiment is defined in Experiment 6.

```
$\mathrm{Exp}^{\text{INT-CTXT}}_{\text{AEAD},\mathcal{A}}(\kappa)$ :
    $\mathrm{k} \leftarrow \mathrm{KeyGen}(\kappa)$, $\mathcal{Q} \leftarrow \emptyset$
    for $i \leftarrow 1$ to q:
        $(\mathrm{A}_i, \mathrm{m}_i) \leftarrow \mathcal{A}(\mathrm{A}_1, \mathrm{m}_1, \mathrm{c}_1, \ldots, \mathrm{A}_{i-1}, \mathrm{m}_{i-1}, \mathrm{c}_{i-1})$
        $\mathrm{c}_i \leftarrow \mathrm{Enc}(\mathrm{k}, \mathrm{A}_i, \mathrm{m}_i)$, $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathrm{A}_i, \mathrm{c}_i)\}$
    $(\mathrm{A}^*, \mathrm{c}^*) \leftarrow \mathcal{A}(\mathrm{A}_1, \mathrm{m}_1, \mathrm{c}_1, \ldots, \mathrm{A}_q, \mathrm{m}_q, \mathrm{c}_q) \in \mathcal{AD} \times \mathcal{C}$
    $\mathrm{m}^* \leftarrow \mathrm{Dec}(\mathrm{k}, \mathrm{A}^*, \mathrm{c}^*)$
    if $\mathrm{m}^* \neq \perp$ and $(\mathrm{A}^*, \mathrm{c}^*) \notin \mathcal{Q}$ return 1, else return 0.
```

Experiment 6: INT-CTXT security experiment for AEAD.

An AEAD scheme is PPT, QPT or HPT IND-CPA secure, and INT-CTXT secure, if for all respective PPT, QPT or HPT adversaries $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa)$, and the advantage $\mathrm{Adv}^{\text{INT-CTXT}}_{\text{AEAD},\mathcal{A}}(\kappa)$, are negligible in the security parameter κ. As a formal inequality, $\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) \leq \mathrm{negl}_{\text{AEAD,IND-CPA}}(\kappa)$ and $\mathrm{Adv}^{\text{INT-CTXT}}_{\text{AEAD},\mathcal{A}}(\kappa) \leq \mathrm{negl}_{\text{AEAD,INT-CTXT}}(\kappa)$.

Note that if an AEAD algorithm is both IND-CPA secure and INT-CTXT secure then the AEAD algorithm is also IND-CCA secure. Note that IND-CCA security implies IND-CPA security but not INT-CTXT security [29].

In the next section, useful assumptions on the PQC algorithms used to authenticate QKD are presented.

4 Post-quantum cryptography and assumptions made in this paper

4.1 Post-quantum security

Post-quantum cryptographic algorithms are designed to be secure against HPT adversaries and are believed to withstand attacks from any classical, quantum or hybrid algorithm that can run in polynomial time. Practically, this means there is no known classical or quantum algorithm that can break a PQC algorithm within a feasible time frame, typically considered to be at least 30 years.

Post-quantum secure signatures, KEMs and other asymmetric cryptographic algorithms, often referred to as PQC algorithms, are relatively recent technologies undergoing standardization by relevant organizations such as NIST [8]. As mentioned earlier, they are believed to be HPT secure. PQC can rely on public key infrastructure (PKI) to manage keys and certificates delivered by a certificate authority (CA), specially for authentication purposes. In the schemes presented in this paper, PQC is used to authenticate honest parties, relying on PKI and post-quantum CAs. The topic of creating a post-quantum secure CA is out of scope.

PRFs, MACs and other symmetric cryptographic PPT algorithms are considered to be quantum resistant. However, they are not typically included when discussing PQC. The primary quantum algorithm found so far that decreases the security of these algorithms, in this case quadratically, is Grover's algorithm [30]. To maintain the security of the symmetric cryptographic PPT algorithms against Grover's algorithm, the size of the secret material must be at least doubled. For the rest of this paper, symmetric cryptographic algorithms are assumed to be HPT secure.

4.2 Practical security

Before discussing why PQC is suitable for authenticating QKD users, it is important to note that PQC and QKD encompass a wide variety of protocols that offer different methods for establishing secure communications between honest parties. Both technologies should complement each other, rather than displace one another. Furthermore, classical cryptography, known to be not post-quantum, can be integrated with PQC and QKD to provide redundant security. Although this topic is out of scope of this paper, examples of hybridization involving QKD, PQC and classical cryptography can be found in the Muckle series [15–18].

As the security of PQC algorithms relies on the belief that the hardness of certain mathematical problems holds against both classical and quantum adversaries, there is no proof that this will remain true for the future. There is always uncertainty about when a classical, quantum, or hybrid algorithm might be developed that could break the hardness of the mathematical problems underlying PQC algorithms. An example of this happening in the past is Shor's algorithm, which, along with advancements in quantum computing technology, motivated the standardization of PQC algorithms.

However, since PQC is built on the assumed hardness of certain mathematical problems, there is a potential gap that could be exploited by honest parties to perform cryptographic operations, such as using PQC for authentication in QKD. The argument relies on the premise that if honest parties can perform their authentication operations and exchanges faster than an HPT adversary can compromise the secret material used in the PQC algorithms by targeting and processing the public material, such as public keys and ciphers, then the honest parties can be assured that no malicious activity targeting authentication has been effectuated by any HPT adversary. The secret keys obtained through a QKD protocol, where the authentication of the classical channel is secured through a PQC protocol, are ITS and $(\varepsilon_{QKD} + \varepsilon_{\text{auth}}^{\text{PQC}})$-secure if honest parties are sure that no successful efficient attack by any HPT adversary has been possible within the time required to authenticate the classical information needed to distill the QKD keys.

A general and conservative runtime condition for the honest parties to be sure that no attack targeting authentication obtained through a PQC protocol has been possible is presented in equation 1.

$$\mathrm{T}_A + \mathrm{T}_B + \mathrm{T}_T < \mathrm{T}_{HPT} \tag{1}$$

Where T_A and T_B are the runtimes required for honest parties A and B to perform the computational operations involving the authentication PQC protocol, T_T is the time accounting for transmission, network delays and other non-trivial times, and T_{HPT} is the time required for an HPT adversary to break the PQC algorithm in question. Note that fractions of T_A, T_B and/or T_T may overlap and such overlap is not taken in account. T_{HPT} is the time required for the adversary to obtain a non-negligible advantage function, defined by the corresponding security experiments/games, which accounts for the fact that the adversaries gain some (negligible or non-negligible) information about the secret material after every query. The adversary time T_{HPT} starts counting once any kind of public information generated by the PQC algorithms from the honest parties is obtained.

While the runtimes from the left side of equation 1, T_A, T_B and T_T, are more or less feasible to estimate via benchmarking, time complexity theory and risk assessments, the adversary runtime T_{HPT} is far from being trivial for several reasons. One reason is that honest parties have no clue when an adversary has taken possession of any public information computed through PQC algorithms by them. Another reason is that the runtime T_{HPT} may be overestimated, as the more public information the HPT adversary is capable of gathering, the more resources it has to break the PQC algorithms faster than predicted. Furthermore, it is hard to know exactly what the computational capacities of the adversary are. Unpredictable advancements in classical and quantum computation fields, as well as unknown attacks, further complicate the task of characterizing the adversary runtime. After characterization of the HPT adversary runtime T_{HPT}, this could be expressed in T_{HPT}-term CAs that would revoke provided certificates once their lifetime surpasses a bound, dependent on an estimation of the HPT adversary runtime T_{HPT}.

However, since PQC (or rather hybrid classical and PQC schemes) are becoming the norm of modern PPT cryptography, it is with the same legitimacy that the assumption that equation 1 holds for some justified HPT runtime T_{HTP}, a time that could range from seconds to days. After that time has passed, a new pair of secret and public keys would need to be generated along with a T_{HPT}-term certificate to validate the identity of the honest parties behind them.

Thus, the PQC algorithms used in the following sections are considered to be T_{HPT}-term HPT secure, according to equation 1 and Definition 17 given below.

Definition 17. ($\mathbf{T}_{HPT}$-term HPT security). An algorithm is T_{HPT}-term HPT secure (or T_{HPT}-term secure) if the advantage function, as defined by the corresponding experiments/games, remains negligible in the relevant security parameter against a HPT adversary for a specified time T, $\mathrm{T} < \mathrm{T}_{HPT}$. Said advantage function becomes no longer negligible once $\mathrm{T} \approx \mathrm{T}_{HPT}$.

Additionally, the symmetric algorithms and the dual PRF used in the following sections are considered to be long-term HPT secure according to the Definition 18.

Definition 18. (Long-term HPT security). An algorithm is long-term HPT secure if the advantage function, as defined by the corresponding experiments/games, remains negligible in the relevant security parameter against a HPT adversary for an arbitrarily long time, typically ranging from 10 to 30 years. Note that if T_{HPT} is estimated to be large enough, Definition 17 and Definition 18 are equivalent.

To prevent the threat from Grover's algorithm [30], the secret keys used inside the long-term HPT secure symmetric primitives, namely dual PRF, AEAD and MAC, have to be at least doubled compared to the usual sizes that resist a non-quantum search threat. For example, to have a bit security close to 256 bits, the secret input length of the PRF has to be 512 bits or more.

5 PQC authentication for QKD

A crucial component that ensures QKD is ITS and $(\varepsilon_{\text{QKD}} + \varepsilon_{\text{auth}})$-secure is the presence of authenticated channels. Specifically, it is essential that honest parties can authenticate the classical traffic used to distill QKD keys.

As shown in the security proof, in section 8, if the authentication algorithms are T_{HPT}-term HPT secure, following Definition 17, honest parties have a time window, strictly smaller than T_{HPT}, during which they are confident that no adversary can execute any attack targeting authentication and integrity. Consequently, as shown in section 8, the following PQC authentication protocols for QKD would remain secure as long as Definition 17 holds, ensuring that the distilled QKD keys are ITS, $(\varepsilon_{\text{QKD}} + \varepsilon_{\text{auth}}^{\text{PQC}})$-secure, and shared by honest parties.

Stronger security for authentication can be adopted once the PQC authentication for QKD protocols have been executed at least once. Given that honest parties will share a secret at that point, ITS authentication can then be achieved using the commonly employed Carter-Wegman MAC construction, which is based on strongly universal hash functions [6, 7, 31, 32]. A practical, lightweight implementation can be constructed following [7], where the type of MAC is defined and optimized to require a minimal amount of secret key material for authentication in each QKD iteration, given an arbitrary authentication security $\varepsilon_{\text{auth}}^{\text{ITS}}$. ITS authentication is not the scope of this work and the topic is mostly omitted in the following sections.

This section discusses the general case where PQC primitives, alongside symmetric primitives, are used to provide authentication, confidentiality and integrity to QKD. Some examples of PQC authentication for QKD are given. The security of the protocols presented is discussed in the subsequent section.

5.1 General case

One of the most used authentication mechanisms for key exchange at the time of writing this paper is the Transport Layer Security (TLS) protocol. The latest version, TLS 1.3 [14], implements Diffie-Hellman (DH) exchanges, with authentication inspired from the SIGMA protocol, using digital certificates. The authenticated key exchange (AKE) steps from TLS 1.3 and SIGMA may be generalized as the creation of an authenticated key and the authentication step as shown in figure 1 (left).

The first step of an AKE corresponds to the creation of an unauthenticated key between two parties, often referred as an ephemeral key since its generation and usage is typically attributed to a specific task making them have a short lifespan. The ephemeral key may be computed through different methods such as a DH exchange, elliptic-curve DH protocol, KEM exchange, among other protocols. The second step of the AKE is to verify the entities behind the key exchange while keeping confidentiality and the integrity of the traded messages. This is often achieved by relying on cryptographic primitives such as signatures, MACs and/or KEMs, along with related tools like pre-shared key (PSK) material or certificates. Specifically, if honest parties have PSK material, authentication typically relies on symmetric primitives such as MACs, which can be designed to be ITS if desired [6, 25, 32]. Otherwise, parties have to rely on asymmetric cryptography, such as signatures and KEMs, and trust-based mechanisms, such as public key infrastructure (PKI) along with certificate authorities (CA), which bind public keys to entities.

This AKE architecture can be used to authenticate classical information to make a QKD

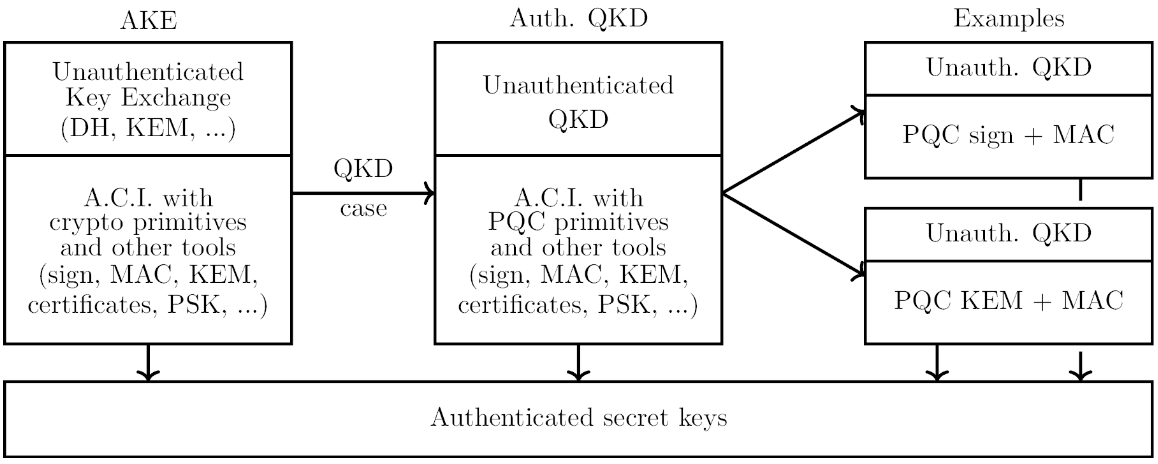

AKE
Unauthenticated Key Exchange (DH, KEM, ...)
A.C.I. with crypto primitives and other tools (sign, MAC, KEM, certificates, PSK, ...)
QKD case
Auth. QKD
Unauthenticated QKD
A.C.I. with PQC primitives and other tools (sign, MAC, KEM, certificates, PSK, ...)
Examples
Unauth. QKD
PQC sign + MAC
Unauth. QKD
PQC KEM + MAC
Authenticated secret keys

Figure 1: Authenticated key exchange general steps (left); Post-quantum AKE applied to QKD (center); Examples of AKE based on QKD (right). A.C.I. = Authentication, Confidentiality and Integrity.

protocol and the corresponding distilled keys secure, by replacing the traditional unauthenticated key exchange by an unauthenticated QKD key obtained through an unauthenticated QKD protocol following Definition 5 (Figure 1, center). A QKD protocol can be executed by two parties without really needing to immediately authenticate the public information required to distill a secret QKD key, the post-processing information. Certainly, this would open the person-in-the-middle vulnerability if no authentication process occurs after an unauthenticated QKD protocol is finalized. The idea behind the authentication is to rely on PKI, along with CA and certificates, and T_{HPT}-term HPT secure primitives (equivalent to T_{HPT}-term secure PQC), such as signatures and KEMs. The information utilized by the honest parties to distill a QKD key through an unauthenticated QKD protocol, such as the post-processing information, is authenticated during the authentication step with an $\varepsilon_{\text{auth}}^{\text{PQC}}$ scheme and by Definition 6, the QKD keys become $(\varepsilon_{\text{QKD}} + \varepsilon_{\text{auth}}^{\text{PQC}})$-secure. The process of authenticating the required information at the end of a scheme is known as "mono-authentication", and is also found in [7] and [10], for the QKD case.

The other properties of interest are confidentiality and integrity. By using the ephemeral keys, along with the derived classical keys through the AKE scheme, honest parties derive key material to encrypt and decrypt the messages exchanged between them, often relying on symmetric algorithms such as the Authenticated Encryption with Associated Data (AEAD) primitive [33], on deriving keys to compute MAC tags to keep in check the integrity of the traded messages. Confidentiality over the entities' information is often called entity protection. The goal of entity protection is to encrypt and conceal the exchanged information between honest parties, such as certificates and ciphers, so that adversaries or eavesdroppers cannot determine the identities behind the traded messages. The entity protection feature within the AKE schemes can be optional by giving up on confidentiality.

The entity protection feature usually provides security against passive adversaries for both parties, and against active adversaries for one of the parties. A passive adversary is limited to eavesdropping on the communications between honest parties, whereas an active adversary can interact with the traded data and actively engage in communications by sending and receiving information. Once an AKE protocol with the entity protection feature is successfully completed between honest parties, it is safe to assume that the concerned entities are protected with the same security as the underlying symmetric protocol used to hide the entities (and/or tags), regardless of whether the adversaries are passive or active.

In the context of a new user wanting to join an existing quantum network, one could argue that it is best to provide entity protection against active adversaries to the existing network user, rather than to the new user. In this way, active adversaries are unable to identify the entities within the network. However, if an adversary knows that a new user is attempting to join the network, the adversary could potentially identify the entity behind the new user and do so for every user joining the network, thereby revealing the entities within the network as it grows. A potential solution for this issue is to provide entity protection against active adversaries to the new user and use a generic entity name for the users within the network. Once the new user is within the network, the real identities can be exchanged by following subsequent procedures that are not commented in this work.

The presented protocols in this work provide entity protection in the same way that SIGMA or TLS does [13, 14]. This relates to the known traffic secrets that TLS and similar protocols such the Muckle series present. The symmetric encryption of "message information" using primitives such as AEAD with a key "key", mainly used to provide entity protection (and tag protection), is noted as $\{\text{message information}\}_{key}$ within the protocols. Note that the messages "m_i" within the digest $\text{H}(\text{m}_i)$ correspond to the message before encryption if "$\{\text{message information}\}_{key}$" is sent, or decrypted if $\{\text{message information}\}_{key}$ is received, "message information" rather than the entire encrypted bit string $\{\text{message information}\}_{key}$. For example, given an exchanged message m_i: $\{\text{message information}\}_{key}$ between parties, $\text{H}(\text{m}) = \text{H}(\text{message information})$ is used, rather than $\text{H}(\{\text{message information}\}_{key})$. Note that "message information" may contain optional traffic other than that presented in the protocol, but the security of the protocol is not affected by this optional traffic.

Examples of authentication methods for QKD, which are described in more detail below, are T_{HPT}-term HPT secure PQC signatures and MAC, or T_{HPT}-term HPT secure KEM and MAC (figure 1, right). The signature then MAC is similar to the SIGMA protocol (and closely related to TLS 1.3) but using an ephemeral key obtained through unauthenticated QKD, while the KEM then MAC process is similar to the KEMTLS protocol [19] but with the same ephemeral unauthenticated QKD key.

The presented schemes assume that honest parties start interacting with each other without any pre-shared keys. To secure the first iteration of QKD, PQC is used for authentication at least for the initial QKD iteration. Once sufficient QKD key material has been obtained and authenticated using PQC, the authentication for QKD can rely on HPT-secure MAC or even ITS-MAC, as mentioned before. However, the presented PQC authentication for QKD examples can be used successively by keeping continuity of the secret state parameter, denoted *SecState* in the protocols.

5.2 QKD authenticated with signature then MAC

Despite the wide variety of QKD protocols [4, 5], all can rely on PQC for authentication for the first instantiation. An unauthenticated QKD may be seen as a black box that outputs unauthenticated ephemeral keys that have been obtained through some public information that must be authenticated at some point to make the whole QKD scheme secure, following Definitions 3, 5 and 6. Regarding the QKD black box, special care has to be taken in terms of the order of exchange of the encoding and decoding bases to sift the raw QKD keys. The initiator sends its encoding bases only when the already used decoding bases from the responder have been received: this is to avoid attacks where quantum memories are involved. The order of the exchanged public information needed to distill a QKD key must follow the steps as if the classical channel was authenticated. The QKD protocol used and how the unauthenticated QKD key is distilled are not discussed in this paper, as the authentication (PQC), confidentiality, (PRF and AEAD) and integrity (PQC and/or MAC) approaches are agnostic regarding this matter.

For the rest of this paper, the QKD protocols are seen as black boxes that provide an unauthenticated shared secret $K_{QKD} = \mathrm{ss}_{QKD_1}||\ldots||ss_{QKD_n}||\mathrm{ss}_{rest}$ and the traded public information to be authenticated m_{QKD}, as specified in Definitions 3 and 5. If the bit length of the key K_{QKD} is not sufficient to run the HPT primitives given by the protocols, honest parties can perform QKD again to produce K'_{QKD} and the corresponding classical traffic m'_{QKD}. The authentication algorithms will authenticate both transcripts, m_{QKD} and m'_{QKD}, to make the individual keys, K_{QKD} and K'_{QKD}. Since this would make the epsilon security of the concatenated key more complex, let us assume that the QKD protocols always produce keys long enough to run the HPT primitives. This assumption is realistic because honest parties have a prior estimate of the minimum key length required to be extracted from QKD in order to authenticate, and even to store for further applications. In this case, the QKD keys are always $\varepsilon_{QKD}(\kappa)$. The intention of honest parties is to derive more key material than what is used in the authentication process, hence they will perform authentication only after distilling enough unauthenticated QKD keys/bits, which is feasible as QKD typically post-processes blocks of 10^5 bits or more.

Within the figures below, the QKD exchange is illustrated by a black double-headed arrow that connects the QKD initiator Alice and the QKD responder Bob. For the rest of the paper, Alice is set as the initiator, whereas Bob is the responder. The authentication provided by the signature then MAC scheme is illustrated in figure 2. As typical in the current state of the art, the single headed arrows correspond to the transmission of classical information through a public and non-authenticated channel.

Step by step. The steps for the signature then MAC protocol are presented below. The security analysis is presented in section 8.

0: Before starting the protocol, the initiator Alice and the responder Bob have known parameters. Alice and Bob have their corresponding T_{HPT}-term HPT secret key, sk_A or sk_B, from the PQC signature algorithm and the corresponding public key, pk_A or pk_B, within a digital certificate, cert_A or cert_B, provided by a legitimate CA. Entity parameters such as certificates are placed inside the parameters A and B for Alice and Bob, respectively. Both parties also share the secret state, *SecState*, that is updated at the end of each execution of the protocol and carry on from one application of the protocol to another. This secret parameter, *SecState*, is initialized to an arbitrary value: it could be initialized at 0 or even take the value given by $\mathrm{PRF}(ss_{QKD}, \text{label}||\mathrm{H}())$ with some bit string that composes the label, for example. Note that the secret keys, sk_A and sk_B, are revealed to no one, including honest parties. The entity parameters, A and B, may be public in the case that entity concealment is not required, otherwise they are secret and only traded following the presented protocol. As discussed in section 5.1, the entity protection feature is secure against passive adversaries for both parties, and secure against active adversaries for one of the parties.

1: The protocol starts with the initiator Alice and the responder Bob performing an unauthenticated QKD protocol following Definition 5. Both parties obtain the QKD traffic to authenticate, m_{QKD}, which is public information, and the corresponding unauthenticated QKD bit string $K_{QKD} = \mathrm{ss}_{QKD}||\mathrm{ss}_{rest}$, which is secret data. In this example of a protocol, the first key, ss_{QKD}, acts as an ephemeral key.

If the unauthenticated QKD is badly performed between honest parties, the condition un-

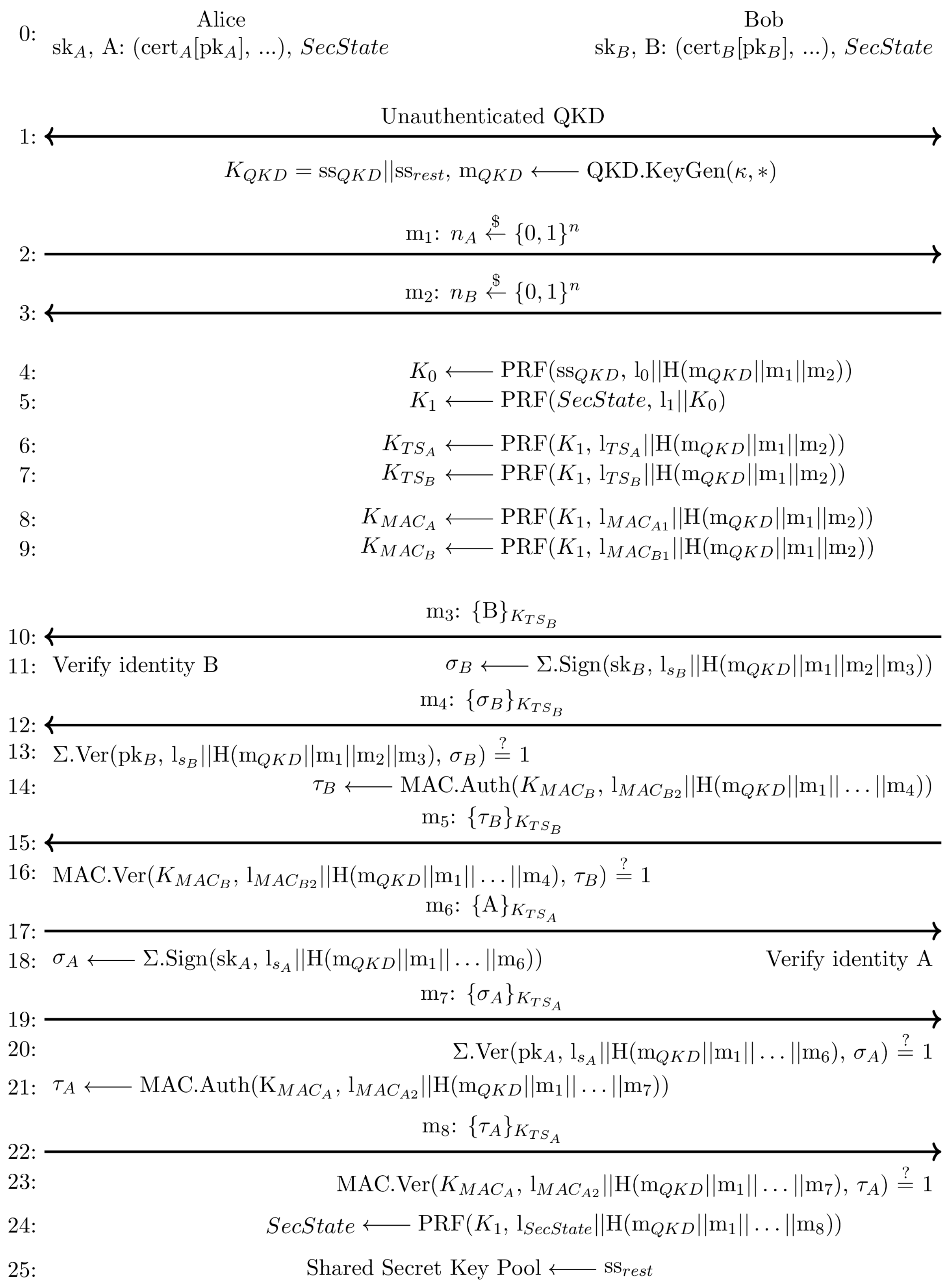

0: Alice sk_A, A: $(\text{cert}_A[\text{pk}_A], ...)$, $\mathit{SecState}$
Bob sk_B, B: $(\text{cert}_B[\text{pk}_B], ...)$, $\mathit{SecState}$
1: Unauthenticated QKD
$K_{QKD} = \text{ss}_{QKD}||\text{ss}_{rest}$, $\text{m}_{QKD} \longleftarrow$ QKD.KeyGen$(\kappa, *)$
2: m_1: $n_A \overset{\$}{\leftarrow} \{0,1\}^n$
3: m_2: $n_B \overset{\$}{\leftarrow} \{0,1\}^n$
4: $K_0 \longleftarrow \text{PRF}(\text{ss}_{QKD}, \text{l}_0||\text{H}(\text{m}_{QKD}||\text{m}_1||\text{m}_2))$
5: $K_1 \longleftarrow \text{PRF}(\mathit{SecState}, \text{l}_1||K_0)$
6: $K_{TS_A} \longleftarrow \text{PRF}(K_1, \text{l}_{TS_A}||\text{H}(\text{m}_{QKD}||\text{m}_1||\text{m}_2))$
7: $K_{TS_B} \longleftarrow \text{PRF}(K_1, \text{l}_{TS_B}||\text{H}(\text{m}_{QKD}||\text{m}_1||\text{m}_2))$
8: $K_{MAC_A} \longleftarrow \text{PRF}(K_1, \text{l}_{MAC_{A1}}||\text{H}(\text{m}_{QKD}||\text{m}_1||\text{m}_2))$
9: $K_{MAC_B} \longleftarrow \text{PRF}(K_1, \text{l}_{MAC_{B1}}||\text{H}(\text{m}_{QKD}||\text{m}_1||\text{m}_2))$
10: m_3: $\{\text{B}\}_{K_{TS_B}}$
11: Verify identity B
$\sigma_B \longleftarrow \Sigma.\text{Sign}(\text{sk}_B, \text{l}_{s_B}||\text{H}(\text{m}_{QKD}||\text{m}_1||\text{m}_2||\text{m}_3))$
12: m_4: $\{\sigma_B\}_{K_{TS_B}}$
13: $\Sigma.\text{Ver}(\text{pk}_B, \text{l}_{s_B}||\text{H}(\text{m}_{QKD}||\text{m}_1||\text{m}_2||\text{m}_3), \sigma_B) \overset{?}{=} 1$
14: $\tau_B \longleftarrow \text{MAC.Auth}(K_{MAC_B}, \text{l}_{MAC_{B2}}||\text{H}(\text{m}_{QKD}||\text{m}_1||\ldots||\text{m}_4))$
15: m_5: $\{\tau_B\}_{K_{TS_B}}$
16: $\text{MAC.Ver}(K_{MAC_B}, \text{l}_{MAC_{B2}}||\text{H}(\text{m}_{QKD}||\text{m}_1||\ldots||\text{m}_4), \tau_B) \overset{?}{=} 1$
17: m_6: $\{\text{A}\}_{K_{TS_A}}$
18: $\sigma_A \longleftarrow \Sigma.\text{Sign}(\text{sk}_A, \text{l}_{s_A}||\text{H}(\text{m}_{QKD}||\text{m}_1||\ldots||\text{m}_6))$
Verify identity A
19: m_7: $\{\sigma_A\}_{K_{TS_A}}$
20: $\Sigma.\text{Ver}(\text{pk}_A, \text{l}_{s_A}||\text{H}(\text{m}_{QKD}||\text{m}_1||\ldots||\text{m}_6), \sigma_A) \overset{?}{=} 1$
21: $\tau_A \longleftarrow \text{MAC.Auth}(\text{K}_{MAC_A}, \text{l}_{MAC_{A2}}||\text{H}(\text{m}_{QKD}||\text{m}_1||\ldots||\text{m}_7))$
22: m_8: $\{\tau_A\}_{K_{TS_A}}$
23: $\text{MAC.Ver}(K_{MAC_A}, \text{l}_{MAC_{A2}}||\text{H}(\text{m}_{QKD}||\text{m}_1||\ldots||\text{m}_7), \tau_A) \overset{?}{=} 1$
24: $\mathit{SecState} \longleftarrow \text{PRF}(K_1, \text{l}_{SecState}||\text{H}(\text{m}_{QKD}||\text{m}_1||\ldots||\text{m}_8))$
25: Shared Secret Key Pool $\longleftarrow \text{ss}_{rest}$

Figure 2: Example of QKD mono-authenticated with PQC signature then MAC.

bounded adversary is able to obtain a non-negligible amount of information of the unauthenticated QKD bit string, $K_{QKD} = \text{ss}_{QKD}||\text{ss}_{rest}$, out of the public QKD traffic m_{QKD}. By consequence, the final QKD keys do not hold the promised ε_{QKD} security, even if authentication is successful, and thus are not secure.

2, 3: Parties exchange random and unique nonces, n_A and n_B. The nonces are required to prevent replay attacks, misbinding attacks, and to avoid derived vulnerabilities [13].

4: Parties derive key material from the QKD material ss_{QKD} thanks to a dual PRF. The derived key K_0 binds the unauthenticated QKD key ss_{QKD} to the public information m_{QKD} and the exchanged nonces. A hash function H is applied to the classical information to compress large bit strings into sizes that fit the dual PRF's input requirements. This adds another potential layer of vulnerability, as an adversary/Eve could tamper with the classical messages such as to arrange that the public information that Alice has, m_{QKD_A}, and the public information that Bob has, m_{QKD_B}, do not match, but their digests do, for example, $\mathrm{H}(\mathrm{m}_{QKD_A}||\mathrm{m}_1||\mathrm{m}_2) = \mathrm{H}(\mathrm{m}_{QKD_B}||\mathrm{m}_1||\mathrm{m}_2)$. The tampering would allow the HPT adversary to gain some information about intermediate keys of the protocols, the output of the protocol or even the output K_{QKD} of the QKD protocol. This potential vulnerability is discussed further in the security analysis, section 8. The considered hash function H follows Definition 8 and does not require a secret key to work. On the same note, if the HPT adversary is passive and does not obtain the shared secret ss_{QKD}, a guess of the PRF output or a tentative forgery is the only way to obtain the intermediate key K_0, since the secret ss_{QKD} is not known, as also discussed in the security proof.

The labels l_i, $i \in \{0, 1, TS_A, TS_B, MAC_{A1}, MAC_{B1}, s_A, s_B, MAC_{A2}, MAC_{B2}, SecState\}$, presented inside the cryptographic primitives, correspond to unique (from each other) bit strings that are public information. The labels allow for domain separation for each intermediate key, provide security against reuse attacks and bind context to the derived keys, as commented in this NIST recommendation strongly related to PRFs [34].

5: The obtained key K_0 is combined with the secret state *SecState* using dual PRFs to derive the intermediate key K_1. The adversary would require to know both of the parameters, *SecState* and K_0, to be able to derive K_1, otherwise a forgery or a guess is the only way to obtain it, for which the success probability is low since PRFs are assumed to be long-term HPT secure (Definition 18).

The parameter *SecState* is known at the first iteration of authentication and gets updated at the very end of the protocol, once the authentication gets accepted by both parties. The inclusion and update of the secret state parameter *SecState* provides a weak form of post-compromise security, in the sense that if an active adversary learns all secrets of a session, security can be recovered only if honest parties complete at least one session in which the adversary has become passive.

6, 7: If entity protection or traffic protection is enabled, the traffic secrets, K_{TS_A} and K_{TS_B}, are derived from K_1. The HPT adversary requires to know K_1 in order to derive the traffic secrets.

Traffic protection, which includes entity protection, against passive attackers is achieved for the initiator and responder at all times. If the secret state *SecState* is a known value to an eavesdropper, either at the initialization step or because it was leaked, and depending on how messages are traded between the initiator and the responder, entity protection against active attacks is achieved for either the initiator or the responder, but not both. In this case, traffic protection against active adversaries for both honest parties is not possible to achieve, as the first traffic protection encryption is always performed with an unauthenticated key, and the parameter *SecState* is known by the adversary: anyone impersonating can compute K_1 if the shared parameter *SecState* is not secret.

The presented protocol 2 achieves active entity protection only for the initiator, but this can be changed to the responder if the order of who reveals their identity first is reversed. An eavesdropper/adversary Eve would be required to have QKD equipment in order to realize an active attack and learn the identity of one of the parties: her presence would be noticed afterwards, when parties require the PQC secret keys to execute PQC algorithms and the protocol would be aborted by the honest party. In the case that the shared parameter *SecState* is secret, then entity protection against active adversaries is achieved by both honest parties.

8, 9: The MAC keys, K_{MAC_A} and K_{MAC_B}, are derived from K_1. These keys are used to create and verify MAC tags. The HPT adversary requires to know K_1 in order to derive the MAC keys.

10: If entity protection is enabled, the responder Bob AEAD encrypts the entity information B with the key K_{TS_B} and the output, $\{\mathrm{B}\}_{K_{TS_B}}$, is sent to the initiator Alice. Otherwise, the entity information B is public knowledge.

11: The initiator Alice verifies the identity B thanks to public information, such as the public key from the responder Bob's certificate. The responder Bob computes the signature tag σ_B by signing the public information $\mathrm{l}_{s_B}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\mathrm{m}_2||\mathrm{m}_3)$ with the secret key sk_B. Note that the HPT adversary would have a negligible chance to create such tag, and thus be detected by Alice in the verification step.

12: If traffic protection is enabled, the responder Bob AEAD encrypts σ_B with the key K_{TS_B} and sends the output, $\{\sigma_B\}_{K_{TS_B}}$, to the initiator Alice. Otherwise, σ_B is sent directly.

13: The initiator Alice (decrypts the received output with the corresponding traffic secret and) verifies the received signature tag with public information $\mathrm{l}_{s_B}||$ $\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\mathrm{m}_2||\mathrm{m}_3)$ and the responder Bob's public key pk_B. Here is where Alice detects HPT adversaries that have tried to forge a signature tag pretending to be Bob. Furthermore, this verification step is what provided entity protection against active adversaries since Alice will abort if the verification is not successful, thus never sending her identity.

14: The responder Bob computes the authentication tag τ_B of the public information $\mathrm{l}_{MAC_{B2}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_4)$ with the derived secret key K_{MAC_B}.

15: If traffic protection is enabled, the responder Bob AEAD encrypts τ_B with the traffic secret K_{TS_B} and sends the output, $\{\tau_B\}_{K_{TS_B}}$, to the initiator Alice. Otherwise, τ_B is sent directly.

16: The initiator Alice (decrypts the received output with the corresponding traffic secret and) verifies the received authentication tag, τ_B, with the public information $\mathrm{l}_{MAC_{B2}}$ $||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_4)$ and the derived secret key K_{MAC_B}. If the adversary does not know K_{MAC_B}, then any forgery attempt of τ_B is detected here with high chance, as taken in account within the security proof.

17: If the verification of the responder Bob's identity, the received signature and the received authentication tag are successful, then the initiator Alice AEAD encrypts the entity information A with the key K_{TS_A}, in the case that entity protection is enabled. The output $\{\mathrm{A}\}_{K_{TS_A}}$ is sent to the responder Bob. If entity protection is not enabled, the entity information A is sent directly.

18: The responder Bob (decrypts the received output with the corresponding traffic secret and) verifies the identity A, utilizing public information such as the public key from the initiator Alice's certificate. The initiator Alice computes the signature tag σ_A by signing the public information $l_{s_A}||H(m_{QKD}||m_1||\dots||m_6)$ with the secret key sk_A. Note that the HPT adversary would have a negligible chance to create such tag, and thus be detected by Bob in the verification step.

19: If traffic protection is enabled, the initiator Alice AEAD encrypts σ_A with the key K_{TS_A} and sends the output, $\{\sigma_A\}_{K_{TS_A}}$, to the responder Bob. Otherwise, σ_A is sent directly.

20: The responder Bob (decrypts the received output with the corresponding traffic secret and) verifies the received signature tag with public information l_{s_A} $||H(m_{QKD}||m_1||\dots||m_6)$ and the initiator Alice's public key pk_A. Here is where Bob detects HPT adversaries that have tried to forge a signature tag pretending to be Alice.

21: The initiator Alice computes the authentication tag τ_A of the public information $l_{MAC_{A2}}||H(m_{QKD}||m_1||\dots||m_7)$ with the derived secret key K_{MAC_A}.

22: If traffic protection is enabled, the initiator Alice AEAD encrypts τ_A with the key K_{TS_A} and sends the output ,$\{\tau_A\}_{K_{TS_A}}$, to the responder Bob. Otherwise, τ_A is sent directly.

23: The responder Bob (decrypts the received output with the corresponding traffic secret and) verifies the received authentication tag with the public information $l_{MAC_{A2}}$ $||$ $H(m_{QKD}||m_1||\dots||m_7)$ and the derived secret key K_{MAC_A}. If the adversary does not know K_{MAC_A}, then any forgery attempt of τ_A is detected here with high chance, as taken in account within the security proof of section 8.

Once both authentication tags have been verified (step 23), the PQC secrets, sk_A and sk_B, can be revealed to any HPT adversary without compromising the security of the now authenticated QKD material ss_{rest}, ss_{QKD}, or the concealment of traffic, including the information of entities.

Alice does not have feedback on Bob's final MAC verification as the protocol is. Thus, denial of service of the final interaction would force Alice to update the final keys whereas Bob would have the precedent values. This could be prevented by adding an authenticated acknowledgment message along with a timeout mechanism. The guarantee of liveness (i.e. the protocol finalizes with honest parties having the same values, despite an adversary interference) will not be discussed further in this work.

After this step 23 is successful, parties can derive session secrets out of the intermediate K_1 to be used in other cryptographic purposes, similar to in TLS [14] or Muckle [15, 17, 18], following the same approach the secret state *SecState* is updated, by using more unique labels. This is not explicitly written down in the protocol, since the goal is to provide authentication, confidentiality, and integrity to the classical traffic of QKD protocols in order to obtain keys given by QKD. These final keys are ITS with an epsilon security to be calculated according to the QKD and authentication security levels. The final security of the keys is given within the security proof of section 8.

24: Both parties update the secret state *SecState* by deriving it from the intermediate key K_1 and the classical traffic traded so far, using a PRF.

25: Honest parties keep the unused QKD bits, ss_{rest}, in a pre-agreed shared secret key pool.

Once the protocol is finalized, the authenticated key ss_{rest} is ready to be used in other cryptographic applications or can even be used to derive additional secret key material.

If authentication is secure, the quantum nature of QKD provides naturally forward security, i.e. compromising all the secrets of a session, including the T_{HPT}-term PQC secrets, sk_A and sk_B, the secret state, *SecState*, and the kept shared bits, ss_{rest}, does not compromise the rest of the keys that are stored inside the shared secret key pool in the previous iterations of the protocol.

As shown in the security analysis, in section 8, the secret material returned by the PQC authentication protocol 2, namely $SecState$ and ss_{rest}, are known only to the authenticated parties (with high probability, 1 - a negligible quantity determined in the security proof), the initiator Alice and the responder Bob, if the T_{HPT}-term HPT security assumption (Definition 17) applied to the PQC algorithm holds while the authentication protocol is in an active state against an HPT adversary. In this case, Definition 6 makes the initial QKD key K_{QKD} and its segments ITS and $(\varepsilon_{\mathrm{QKD}} + \varepsilon^{\mathrm{PQC}}_{\mathrm{auth}})$-secure. Because the assumption that AEAD is long-term HPT (Definition 18) secure holds, the traffic protection tags are long-term HPT secure as well. This implies that an unbounded adversary would be able to compromise the traffic secrets, K_{TS_A} and K_{TS_B}, and trace back until compromising the old, already used, secret state $SecState$ (step 0) and ss_{QKD} from K_1 and K_0, respectively. Additionally, this compromises the updated secret state $SecState$ as well (step 24). In other words, after the protocol is finalized, the key material ss_{QKD}, K_0, K_1 and all derived keys, including the secret state $SecState$, are not ITS but rather long-term HPT-secure. Note that this does not affect the ITS security of the final keys derived in the present and past sessions, stored inside the shared secret key pool (step 25).

The PQC signature authentication process shown in figure 2 requires 1.5 round-trip times (RTT) and could be initiated within the last interactions of the post-processing procedure, within the unauthenticated QKD step, depending on which party has to send the certificate first, which is pre-agreed according to who needs the most entity protection against active adversaries.

Additionally, the presented protocol could be modified to cover the case where both parties have knowledge of the certificates, or when only one party needs to authenticate, as is the case for some client-server interactions.

Since the protocol takes as input the ephemeral key ss_{QKD}, the minimum QKD key length required to execute 2 is exactly the size of the PRF input. On the other hand, if ITS MAC authentication is desired, some shared secret key pool of considerable size should be anticipated and kept, as ITS MAC consumes key material every time an authentication tag has to be sent, on top of having enough keys to not exhaust the key pool in case of denial of service attack.

Additional redundant protection may be added, for example by initiating the protocol with an ephemeral PQC key obtained through a PQC KEM exchange. The ephemeral PQC key can be used to provide an additional layer of confidentiality to the traded QKD public information, in the same way traffic protection works within the authentication protocol from figure 2. If PQC redundancy is added, an eavesdropper would need to execute PQC algorithms, possess QKD equipment, and know the secret state parameter, $SecState$, to carry out an active attack on the entity protection feature and identify the vulnerable party. Eventually, the active attacker would be detected with high probability by the honest party, once the PQC secret material is required to perform authentication.

The next subsection provides an example of authentication that does not rely on signatures.

5.3 QKD authenticated with KEM then MAC

Previous implementations have found that PQC HAKE protocols based on certain KEM algorithms run with fewer computational cycles and with lower memory requirements, compared to the signature-based protocols, but with the additional cost of an extra RTT [18, 19]. Following the same motivation, this subsection presents an alternative approach to mono-authenticate QKD with PQC KEM. Note that KEM and signature protocols could be combined to achieve redundant protection where both of the PQC algorithms used to authenticate are built on different mathematical problems [8].

The PQC KEM-based authentication protocol follows a similar structure to the sign then MAC protocol from figure 2. The authentication mechanism relies on T_{HPT}-term HPT secure KEM algorithms, rather than signatures. The protocol is presented in figure 3.

Step by step. The steps for the KEM then MAC protocol are presented below. The security analysis is presented in section 8.

0, 1, 2, 3, 4, 5, 6, 7, 8: Same as steps 0, 1, 2, 3, 4, 5, 6, 7, 10, respectively, from the signature then MAC case (figure 2). In this case, the T_{HPT}-term HPT secret keys, sk_A and sk_B, are associated with a PQC KEM algorithm. Alice and Bob can have two different KEM algorithms, denoted as KEM_A and KEM_B in figure 3. The choice of the KEMs is arbitrary and depends on the needs of the honest parties, including security levels, diversification of PQC algorithms, performance efficiency or memory requirements.

The labels l_i, $i \in \{0, 1, TS_{A1}, TS_{B1}, 2, TS_{A2}, TS_{B2}, 3, MAC_{A1}, MAC_{B1}, MAC_{A2}, MAC_{B2}\}$ presented inside the PRFs, correspond to unique bit strings that are public information. The labels allow for domain separation for each intermediate key, provides security against reuse attacks and binds context to the derived keys, as commented in this NIST recommendation strongly related to PRFs [34].

9: The initiator Alice verifies the identity B using public information, such as the public key from the responder Bob's certificate. If the verification is successful, the initiator Alice computes the encapsulation c_B of a random key k_B by using the encapsulation algorithm KEM_B.Enc along with the responder Bob's public key pk_B.

10: If traffic protection is enabled, the initiator Alice AEAD encrypts c_B with the key $K_{TS_{A1}}$ and sends the output, $\{\mathrm{c}_B\}_{K_{TS_{A1}}}$, to the responder Bob. Otherwise, c_B is sent directly.

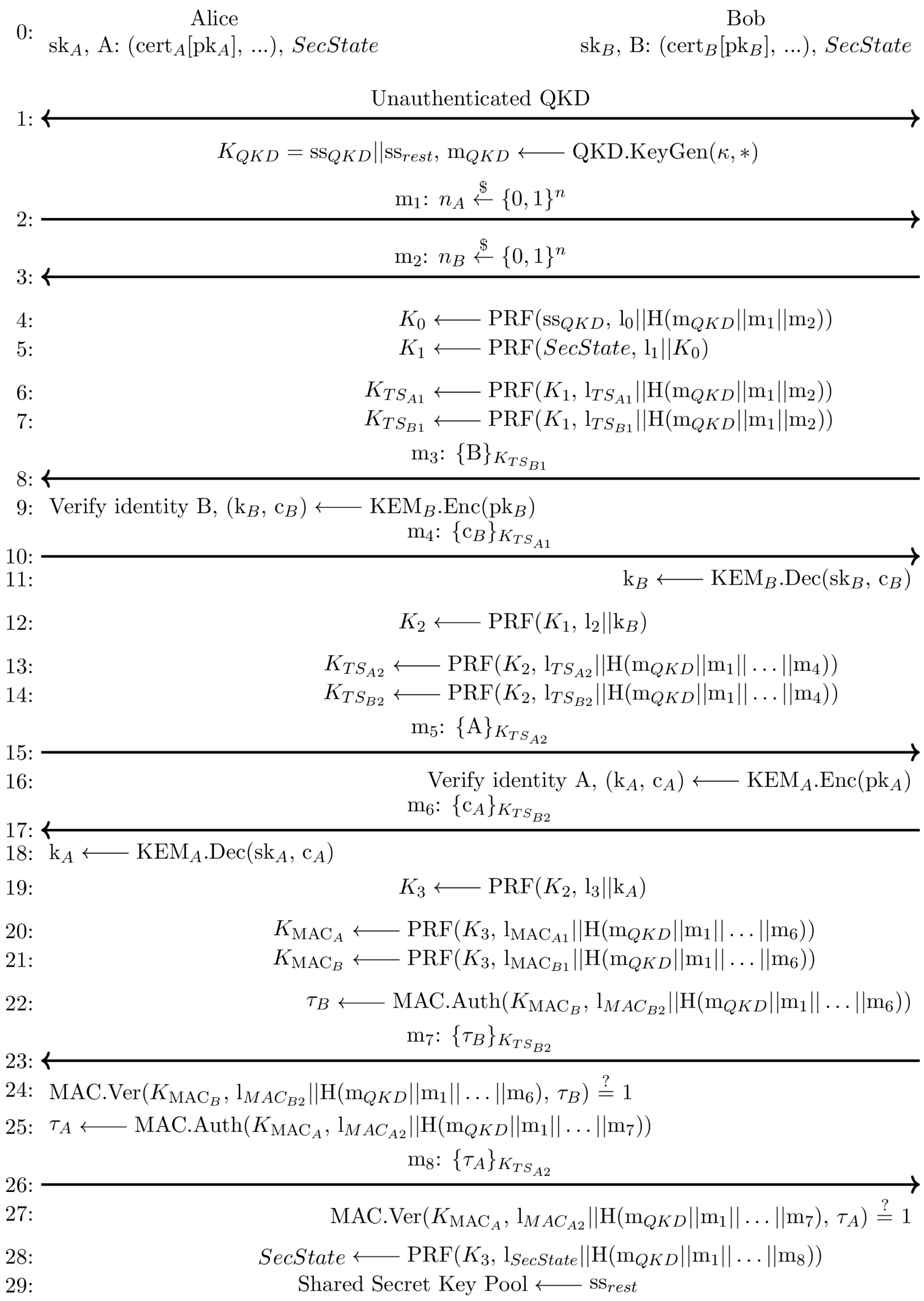
11: The responder Bob (decrypts the received output with the corresponding traffic secret and) decapsulates the random secret k_B from the encapsulation cipher c_B using the decapsulation algorithm KEM_B.Dec along with the secret key sk_B. Without the knowledge of the secret key sk_B, an HPT adversary is not able to decapsulate the random secret k_B with high probability as taken into account within the security proof.

12: The shared key k_B is combined with the intermediate key K_1 using a dual PRF to obtain the intermediate key K_2. An adversary without k_B has negligible probability of forging K_2.

13, 14: If the entity protection feature is enabled, the entity protection keys, $K_{TS_{A1}}$ and $K_{TS_{B1}}$, are derived from K_2. An adversary without K_2 has negligible probability of forging both.

15: If entity protection is enabled, the initiator Alice AEAD encrypts the entity information A with the key $K_{TS_{A2}}$ and sends the output, $\{\mathrm{A}\}_{K_{TS_{A2}}}$, to the responder Bob. Otherwise, the entity information A is public knowledge.

16: The responder Bob (decrypts the received output with the corresponding traffic secret and) verifies the identity A using public information, such as the public key from the initiator Alice's certificate. If the verification is successful, the responder Bob computes the encapsulation c_A of a random key k_A using the encapsulation algorithm KEM_A.Enc



0: Alice: sk_A, A: ($\mathrm{cert}_A[\mathrm{pk}_A]$, ...), *SecState* — Bob: sk_B, B: ($\mathrm{cert}_B[\mathrm{pk}_B]$, ...), *SecState*

1: $\longleftrightarrow$ Unauthenticated QKD

$K_{QKD} = \mathrm{ss}_{QKD}||\mathrm{ss}_{rest}$, $\mathrm{m}_{QKD} \longleftarrow$ QKD.KeyGen$(\kappa, *)$

2: $\longrightarrow$ m_1: $n_A \overset{\$}{\leftarrow} \{0,1\}^n$

3: $\longleftarrow$ m_2: $n_B \overset{\$}{\leftarrow} \{0,1\}^n$

4: $K_0 \longleftarrow \mathrm{PRF}(\mathrm{ss}_{QKD},\ \mathrm{l}_0||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\mathrm{m}_2))$

5: $K_1 \longleftarrow \mathrm{PRF}(\mathit{SecState},\ \mathrm{l}_1||K_0)$

6: $K_{TS_{A1}} \longleftarrow \mathrm{PRF}(K_1,\ \mathrm{l}_{TS_{A1}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\mathrm{m}_2))$

7: $K_{TS_{B1}} \longleftarrow \mathrm{PRF}(K_1,\ \mathrm{l}_{TS_{B1}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\mathrm{m}_2))$

8: $\longleftarrow$ m_3: $\{\mathrm{B}\}_{K_{TS_{B1}}}$

9: Verify identity B, $(\mathrm{k}_B,\ \mathrm{c}_B) \longleftarrow \mathrm{KEM}_B.\mathrm{Enc}(\mathrm{pk}_B)$

10: $\longrightarrow$ m_4: $\{\mathrm{c}_B\}_{K_{TS_{A1}}}$

11: $\mathrm{k}_B \longleftarrow \mathrm{KEM}_B.\mathrm{Dec}(\mathrm{sk}_B,\ \mathrm{c}_B)$

12: $K_2 \longleftarrow \mathrm{PRF}(K_1,\ \mathrm{l}_2||\mathrm{k}_B)$

13: $K_{TS_{A2}} \longleftarrow \mathrm{PRF}(K_2,\ \mathrm{l}_{TS_{A2}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_4))$

14: $K_{TS_{B2}} \longleftarrow \mathrm{PRF}(K_2,\ \mathrm{l}_{TS_{B2}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_4))$

15: $\longrightarrow$ m_5: $\{\mathrm{A}\}_{K_{TS_{A2}}}$

16: Verify identity A, $(\mathrm{k}_A,\ \mathrm{c}_A) \longleftarrow \mathrm{KEM}_A.\mathrm{Enc}(\mathrm{pk}_A)$

17: $\longleftarrow$ m_6: $\{\mathrm{c}_A\}_{K_{TS_{B2}}}$

18: $\mathrm{k}_A \longleftarrow \mathrm{KEM}_A.\mathrm{Dec}(\mathrm{sk}_A,\ \mathrm{c}_A)$

19: $K_3 \longleftarrow \mathrm{PRF}(K_2,\ \mathrm{l}_3||\mathrm{k}_A)$

20: $K_{\mathrm{MAC}_A} \longleftarrow \mathrm{PRF}(K_3,\ \mathrm{l}_{\mathrm{MAC}_{A1}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_6))$

21: $K_{\mathrm{MAC}_B} \longleftarrow \mathrm{PRF}(K_3,\ \mathrm{l}_{\mathrm{MAC}_{B1}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_6))$

22: $\tau_B \longleftarrow \mathrm{MAC.Auth}(K_{\mathrm{MAC}_B},\ \mathrm{l}_{MAC_{B2}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_6))$

23: $\longleftarrow$ m_7: $\{\tau_B\}_{K_{TS_{B2}}}$

24: $\mathrm{MAC.Ver}(K_{\mathrm{MAC}_B},\ \mathrm{l}_{MAC_{B2}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_6),\ \tau_B) \overset{?}{=} 1$

25: $\tau_A \longleftarrow \mathrm{MAC.Auth}(K_{\mathrm{MAC}_A},\ \mathrm{l}_{MAC_{A2}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_7))$

26: $\longrightarrow$ m_8: $\{\tau_A\}_{K_{TS_{A2}}}$

27: $\mathrm{MAC.Ver}(K_{\mathrm{MAC}_A},\ \mathrm{l}_{MAC_{A2}}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_7),\ \tau_A) \overset{?}{=} 1$

28: $\mathit{SecState} \longleftarrow \mathrm{PRF}(K_3,\ \mathrm{l}_{SecState}||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\ldots||\mathrm{m}_8))$

29: Shared Secret Key Pool $\longleftarrow \mathrm{ss}_{rest}$

Figure 3: Example of QKD mono-authentication with PQC KEM then MAC.

along with the initiator Alice's public key pk_A. Note that in the case Eve/adversary is impersonating Bob, since K_2 has negligible chance to be forged, so is the traffic secret

$K_{TS_{A2}}$, giving the entity protection against active adversaries, discussed in section 5.1.

17: If traffic protection is enabled, the responder Bob AEAD encrypts c_A with the key $K_{TS_{A2}}$, and sends the output, $\{c_A\}_{K_{TS_{A2}}}$, to the initiator Alice. Otherwise, c_A is sent directly.

18: The initiator Alice (decrypts the received output with the corresponding traffic secret and) decapsulates the random secret k_A from the encapsulation cipher c_A using the decapsulation algorithm KEM_A.Dec along with the secret key sk_A. Without the knowledge of the secret key sk_A, an HPT adversary is not able to decapsulate the random secret k_A with high probability, as taken into account within the security proof.

19: The shared key k_A is combined with the intermediate key K_2 using a dual PRF to obtain the intermediate key K_3. An adversary without k_A has negligible probability of forging K_3.

20, 21: The MAC keys, K_{MAC_A} and K_{MAC_B}, are derived from K_3. An adversary without K_3 has negligible probability of forging both.

22: The responder Bob computes the authentication tag τ_B of the public information $l_{MAC_{B2}}||H(m_{QKD}||m_1||\dots||m_6)$ with the derived secret key K_{MAC_B}.

23: If traffic protection is enabled, the responder Bob AEAD encrypts τ_B with the key $K_{TS_{B2}}$ and sends the output, $\{\tau_B\}_{K_{TS_{B2}}}$, to the initiator Alice. Otherwise, τ_B is sent directly.

24: The initiator Alice (decrypts the received output with the corresponding traffic secret and) verifies the received authentication tag with the public information $l_{MAC_{B2}} || H(m_{QKD}||m_1||\dots||m_6)$ and the derived secret key K_{MAC_B}. If an HPT adversary tried to forge the MAC tag, Alice would detect this within the verification step with a high probability, as taken into account within the security proof.

25: The initiator Alice computes the authentication tag τ_A of the public information $l_{MAC_{A2}}||H(m_{QKD}||m_1||\dots||m_7)$ with the derived secret key K_{MAC_A}.

26: If traffic protection is enabled, the initiator Alice AEAD encrypts τ_A with the key $K_{TS_{A2}}$ and sends the output, $\{\tau_A\}_{K_{TS_{A2}}}$, to the responder Bob. Otherwise, τ_A is sent directly.

27: The responder Bob (decrypts the received output with the corresponding traffic secret and) verifies the received authentication tag with the public information $l_{MAC_{A2}}||H(m_{QKD}$ $||m_1||\dots||m_7)$ and the derived secret key K_{MAC_A}. If an HPT adversary tried to forge the MAC tag, Bob would detect this within the verification step with a high probability, as taken into account within the security proof.

28: Both parties update the secret state *SecState* by deriving it from the intermediate key K_3 and the classical traffic traded so far.

29: Honest parties keep the unused QKD bits, ss_{rest}, in a pre-agreed shared secret key pool.

Entity protection is provided in the same way as in the signature-based protocol. The comments specified inside the signature then MAC version also apply here, since they are agnostic of which PQC algorithm is used to authenticate.

As shown in the security analysis, in section 8, the secret material returned by the PQC authentication protocol 2, namely *SecState*, and ss_{rest}, are known only to the authenticated parties (with high probability), the initiator Alice and the responder Bob, if the T_{HPT}-term HPT security assumption (Definition 17) applied to the PQC algorithm holds while the authentication protocol is in an active state against HPT adversaries, as shown in the security proof, section 8. In this case, Definition 6 makes the initial QKD key and its segments ITS and $(\varepsilon_{\text{QKD}} + \varepsilon_{\text{auth}}^{\text{PQC}})$-secure. Given the assumption that AEAD is long-term HPT (Definition 18) secure holds, the entity protection tags are long-term HPT secure. This implies that an unbounded adversary would be able to compromise the entity protection keys, $K_{TS_{A1}}$ and $K_{TS_{B1}}$, and trace back until compromising the old, already used, secret state *SecState* (step 0) and ss_{QKD} from K_1 and K_0, respectively. In other words, after the protocol is finalized, the QKD key ss_{QKD}, and the secret state *SecState* (step 0), are not ITS but rather long-term HPT-secure. The remaining intermediate keys, K_2 and K_3, are long-term HPT secure whereas the PQC secrets k_A and k_B are T_{HPT}-term HPT secure. Since the intermediate key K_3 is long-term secure, the derived secret state *SecState* (step 28) is long-term secure too. Note that this does not affect the ITS security of the final keys derived in the present and past sessions, stored inside the shared secret key pool (step 29).

Similarly with the signature protocol, if ITS MAC authentication is desired for the next QKD iterations, the shared secret key pool has to be filled according to some agreed size between the honest parties. Indeed, once the shared secret key pool is filled sufficiently, (ITS or non-ITS) authentication without relying on PQC in posterior QKD iterations may be undertaken.

Since this work focuses rather on non-ITS primitives, the next section provides an example on how to achieve authentication using a non-ITS MAC algorithm by taking a strong inspiration from the original Muckle protocol [15].

6 Further QKD iterations: shared key scenario

Once parties have performed PQC authentication for QKD in the first iteration successfully, authentication can stop relying on asymmetric cryptography since the shared secret key pool contains some ITS and $(\varepsilon_{\text{QKD}} + \varepsilon_{\text{auth}})$-secure keys stored inside. Authentication relying on symmetric but non-ITS, long-term HPT-secure, primitives such as most MAC functions is possible. In the same way, authentication relying on ITS primitives such as ITS-MAC algorithms is achievable. This section discusses an example for long-term HPT-secure authentication (non-ITS).

Theorem 23, obtained after performing the security analysis, in section 8, shows that the security of QKD keys, that are stored within the shared secret key pool, decreases in a polynomial manner with the number of parties, sessions and iterations. This implies that the keys generated in previous QKD iterations have an advantage to be used for further authentication steps regarding the keys generated in later QKD iterations.

Further QKD iterations can be mono-authenticated securely thanks to the now non-empty shared secret key pool and the long-term HPT-secure MAC algorithms. An example protocol is presented in figure 4.

Step by step. The steps for the shared key-based mono-authentication for QKD using long-term HPT-secure MAC are presented below. The security analysis is presented in section 8.

0: The initiator Alice and the responder Bob hold a non-empty ε-secure ITS shared secret key pool, where ε is estimated in previous steps, and the secret state parameter *SecState*, given by previous iterations, or initialized at a known, and public, value with the condition that the shared secret key pool has enough key material to run the presented MAC

0: Alice — Shared Secret Key Pool, *SecState*, *ctr* | Bob — Shared Secret Key Pool, *SecState*, *ctr*

1: $\xleftrightarrow{\text{Unauthenticated QKD}}$

$K_{QKD} = \mathrm{ss}_{rest},\ \mathrm{m}_{QKD} \longleftarrow \text{QKD}$

2: $K_0 \longleftarrow \text{Shared Secret Key Pool}$

3: $K_1 \longleftarrow \mathrm{PRF}(SecState,\ \mathrm{l}_1||ctr||\mathrm{K}_0)$

4: $K_{TS_A} \longleftarrow \mathrm{PRF}(K_1,\ \mathrm{l}_{TS_A}||ctr||\mathrm{H}(\mathrm{m}_{QKD}))$

5: $K_{TS_B} \longleftarrow \mathrm{PRF}(K_1,\ \mathrm{l}_{TS_B}||ctr||\mathrm{H}(\mathrm{m}_{QKD}))$

6: $K_{MAC_A} \longleftarrow \mathrm{PRF}(K_1,\ \mathrm{l}_{MAC_{A1}}||ctr||\mathrm{H}(\mathrm{m}_{QKD}))$

7: $K_{MAC_B} \longleftarrow \mathrm{PRF}(K_1,\ \mathrm{l}_{MAC_{B1}}||ctr||\mathrm{H}(\mathrm{m}_{QKD}))$

8: $\tau_A \longleftarrow \mathrm{MAC.Auth}(K_{MAC_A},\ \mathrm{l}_{MAC_{A2}}||ctr||\mathrm{H}(\mathrm{m}_{QKD}))$

9: $\xrightarrow{\mathrm{m}_1:\ \{\tau_A\}_{K_{TS_A}}}$

10: $\mathrm{MAC.Ver}(K_{MAC_A},\ \mathrm{l}_{MAC_{A2}}||ctr||\mathrm{H}(\mathrm{m}_{QKD}),\ \tau_A) \stackrel{?}{=} 1$

11: $\tau_B \longleftarrow \mathrm{MAC.Auth}(K_{MAC_B},\ \mathrm{l}_{MAC_{B2}}||ctr||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1))$

12: $\xleftarrow{\mathrm{m}_2:\ \{\tau_B\}_{K_{TS_B}}}$

13: $\mathrm{MAC.Ver}(K_{MAC_B},\ \mathrm{l}_{MAC_{B2}}||ctr||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1),\ \tau_B) \stackrel{?}{=} 1$

14: $SecState \longleftarrow \mathrm{PRF}(K_1,\ \mathrm{l}_{SecState}||ctr||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1||\mathrm{m}_2))$

15: $ctr \longleftarrow ctr + 1$

16: $\text{Shared Secret Key Pool} \longleftarrow \mathrm{ss}_{rest}$

Figure 4: Example of shared key-based long-term HPT-secure mono-authentication for QKD.

protocol. They also share a public counter *ctr*, initialized at 0. In this occasion public key infrastructure is not required since honest parties share secrets that no adversary has (with negligible probability), hence there is no need to have certificates.

1: The protocol starts with the initiator Alice and the responder Bob performing an unauthenticated QKD protocol following Definition 5. Both parties obtain the public information m_{QKD} and the corresponding unauthenticated QKD bit string $K_{QKD} = \mathrm{ss}_{rest}$.

2: Honest parties agree on a shared secret K_0, taken from the shared secret key pool. Each key has a security level associated and most likely an unique ID to differentiate them easily. Earlier QKD keys present a better security than the ones distilled in later QKD iterations: this could be a selection criteria.

3: The obtained key K_0 is combined with the secret state *SecState* using dual PRFs to derive the intermediate keys K_1. An HPT adversary without the key K_0 has negligible probability to forge the intermediate key K_1.

The labels l_i, $i \in \{1, TS_A, TS_B, MAC_{A1}, MAC_{B1}, MAC_{A2}, MAC_{B2}, SecState\}$, presented inside the cryptographic primitives, correspond to unique (with each other) bit strings that are public information. The labels, combined with the counter *ctr*, allow for domain separation for each intermediate key, provide security against reuse attacks and bind context to the derived keys, as commented in this NIST recommendation strongly related to PRFs [34].

4, 5: If traffic protection is enabled, the traffic secrets, K_{TS_A} and K_{TS_B}, are derived from K_1. An HPT adversary without the intermediate key K_1 has negligible probability to forge both traffic secrets.

Traffic secrets are needed when honest parties want confidentiality regarding the message structure or metadata. Otherwise, the derivation of traffic secrets is optional as the MAC tags, τ_A and τ_B, along with the shared keys, K_0 and K_1, are sufficient to provide integrity and authenticity.

Traffic protection is maintained against active HPT adversaries for both honest parties since knowledge of K_0 and the secret state parameter is required to be able to forge the traffic secrets.

6, 7: The MAC keys, K_{MAC_A} and K_{MAC_B}, are derived from K_1. An HPT adversary without the intermediate key K_1 has negligible probability to forge both MAC secrets.

8: The initiator Alice computes the long-term HPT-secure authentication tag τ_A of the public information $\mathrm{l}_{MAC_{A2}}||ctr||$ $\mathrm{H}(\mathrm{m}_{QKD})$ with the derived secret key K_{MAC_A}. An HPT adversary without the MAC key K_{MAC_A} has negligible probability to forge the MAC tag.

9: If traffic protection is enabled, the initiator Alice AEAD encrypts τ_A with the key K_{TS_A} and sends the output, $\{\tau_A\}_{K_{TS_A}}$, to the responder Bob. Otherwise, τ_A is sent directly.

10: The responder Bob (decrypts the received output with the corresponding traffic secret and) verifies the received authentication tag with the public information $\mathrm{l}_{MAC_{A2}}||ctr$ $||\mathrm{H}(\mathrm{m}_{QKD})$ and the derived secret key K_{MAC_A}. An impersonation from an HPT adversary that tried to forge the MAC tag τ_A is detected with high probability, as taken into account within the security proof.

11: The responder Bob computes the long-term HPT-secure authentication tag τ_B of the public information $\mathrm{l}_{MAC_{B2}}||ctr||\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1)$ with the derived secret key K_{MAC_B}. An HPT adversary without the MAC key K_{MAC_B} has negligible probability to forge the MAC tag.

12: If traffic protection is enabled, the responder Bob AEAD encrypts τ_B with the key K_{TS_B} and sends the output, $\{\tau_B\}_{K_{TS_B}}$, to the initiator Alice. Otherwise, τ_B is sent directly.

13: The initiator Alice (decrypts the received output with the corresponding traffic secret and) verifies the received authentication tag with the public information $\mathrm{l}_{MAC_{B2}}||ctr||$ $\mathrm{H}(\mathrm{m}_{QKD}||\mathrm{m}_1)$ and the derived secret key K_{MAC_B}. An impersonation from an HPT adversary that tried to forge the MAC tag τ_B is detected with high probability, as taken in account within the security proof.

14: Both parties update the secret state $SecState$ by deriving it from the intermediate key K_1 and the classical traffic traded so far.

15: The counter ctr is updated with $ctr + 1$.

Since the key extracted from the shared secret key pool provides randomness, uniqueness has to be included to avoid replay attacks. The counter ctr provides the uniqueness required, so nonces are not necessary in this scheme.

16: Honest parties keep the now authenticated QKD bits, ss_{rest}, in the pre-agreed shared secret key pool.

Note that in this protocol, there is no involvement of ephemeral key. Adding an ephemeral key would add another layer of protection regarding post-compromise security. If it is assumed that the HPT adversary Eve has knowledge of the shared secret key pool and the secret state parameter $SecState$ in one iteration "a", and if she becomes passive for at least an iteration "b" > "a", then the parameter $SecState$ would be updated by the honest parties to a value unknown to Eve, because of the ephemeral key. Thus the adversary Eve would be detected at the MAC verification step by honest parties.

The choice of the non-ITS MAC function may depend on a security level parameter and the secret key consumption constraint, if there is any such constraint. The obvious choice is to use a MAC function with a time complexity that scales well with the length input, in addition to requiring little secret key material to operate. Possible candidates would be Hash-based MAC (HMAC) [35], Cipher-based Message Authentication Code (CMAC) [36, 37] or Galois Message Authentication Code (GMAC) [38, 39].

These MAC algorithms allow reuse of the shared secret key K_0 without compromising the security of the schemes. In that scenario, the key eventually expires, either because it has been too long since the last update, or because it has been used too many times. Renewal of the key is done by simply repeating step 2: a new QKD key is extracted from the shared secret key pool. For the security proof of the non-ITS MAC-based protocol, K_0 is assumed to be refreshed every time the protocol is run, to facilitate some post-compromise security properties and freshness. If the shared secret key pool has insufficient bits, the PQC-based protocol would need to be run again.

7 Hybrid Authenticated Key Exchange Framework

To show that the presented examples (Figures 2, 3 and 4) are secure, an adapted version of the HAKE framework introduced in [15] and recalled in [17, 18] is used. This section recalls and adapts the tools required to execute the HAKE framework in the context of this work. The algorithmic description of the security model, $\mathrm{Exp}_{\Pi,n_P,n_S,n_T}^{\mathrm{HAKE,clean},\mathcal{A}}(\kappa)$, is reused and is not recalled within this work, however it can be found in [15], figure 5 of Appendix C. To investigate security, the indistinguishability experiment is played between an HPT challenger $\mathcal{C}$ and two adversaries $\mathcal{A} = \{\mathcal{A}_{\mathcal{HPT}}, \mathcal{A}_{unbounded}\}$, where $\mathcal{A}_{\mathcal{HPT}}$ and $\mathcal{A}_{unbounded}$ correspond to an HPT (Definition 1) and conditioned unbounded (Definition 2) adversary, respectively.

The presented examples do not use classical cryptography (non-PQC), however the tools to deal with classical primitives are also presented for completeness. Protocols that combine classical cryptography with different PQC algorithms follow a practice to prevent that if one of the algorithms believed to be secure is found to be compromised, then said combined protocol's security would still hold. This is useful, especially nowadays where PQC algorithms are under intense scrutiny and testing to check their practical security.

7.1 Secret Key Generation

Recall and adaptation of the HAKE framework. The HAKE framework categorizes secret key generation into long-term and ephemeral. Long-term keys are generated once and reused in every execution of a protocol, while ephemeral keys are freshly generated at certain stages within each protocol execution. Furthermore, these are divided into the following sub-categories:

- Post-quantum asymmetric secret generation. This includes all public-secret key pair generation given by any PQC algorithm. The algorithm that generates these secrets for a long-term manner is denoted as LQKeyGen, while EQKeyGen denotes the ephemeral version.

- Classical asymmetric secret generation. This includes all public-secret key pair generation given by any classical algorithm, known to be insecure against quantum algorithms. The algorithm that generates these secrets for a long-term manner is denoted as LCKeyGen, while ECKeyGen denotes the ephemeral version.
- Symmetric secrets. This includes pre-shared keys and keys distilled from an already working QKD network. The algorithm that generates these secrets for a long-term manner is denoted as LSKeyGen, while ESKeyGen denotes the ephemeral version.

The set of algorithms that produces the key material are denoted as:

- XYKeyGen(κ) $\overset{\$}{\rightarrow}$ (pk, sk), where X $\in \{L, E\}$, Y $\in \{Q, C\}$, pk is public key, sk is secret key and κ is a security parameter.
- XSKeyGen(κ) $\overset{\$}{\rightarrow}$ (ss, ssid), where X $\in \{L, E\}$, ss is shared secret (or symmetric secret), ssid is the identity parameter of the shared secret ss and κ is a security parameter.

Adaptations, added conditions and comments. Long-term Post-Quantum secret key generation is indirectly replaced by T_{HPT}-term Post-Quantum secret key generation, following Definition 17. This distinction evokes that the security of the PQC algorithm secrets has a temporal bound T_{HPT} that relates to the bounded resources of the considered HPT adversaries, required to make the QKD iterations secure. Additionally, the presented protocols do not use ephemeral Post-Quantum secrets, thus EQKeyGen is never executed (or returns $\perp$). However, ephemeral asymmetric post-quantum algorithms can be implemented to achieve redundant security properties, similarly to the approach in the Muckle series [15–18].

Furthermore, classical asymmetric secret generation is never used within the presented protocols, thus LCKeyGen and ECKeyGen are never executed (or return $\perp$). However, classical asymmetric algorithms can be added to provide redundant security properties, as discussed at the start of this section.

The unauthenticated QKD keys from the first steps of the PQC-based protocols are considered symmetric ephemeral keys, thus, if ephemeral QKD keys are involved, then ESKeyGen is executed via QKD.KeyGen in the presented protocols.

Since the protocols presented here are designed to require no pre-shared key, LSKeyGen is not executed (or returns $\perp$). Once any of the presented protocols (Figures 1, 2, 3 and 4) are securely finalized, parties obtain a non empty shared secret key pool given by the unused QKD bits. These QKD bits stored inside the shared secret key pool are considered to be equivalent to session keys. Additionally, if the session keys are used to perform the MAC-based protocol, the used keys are no longer considered as session keys, but rather as a different element (shared secret key pool keys), with the condition that the session keys were securely derived.

Next, the HAKE execution environment is recalled and adapted.

7.2 Execution Environment

Recall and adaptation of the HAKE framework. The environment is composed of n_P parties $P_1, \ldots, P_{n_P}$ capable of interacting with each other. Each party is able to run n_S sessions of a general key-exchange protocol denoted by Π. Each session consists of n_T stages: i.e., for a new stage, the protocol Π is run once again while maintaining the per-session parameters from the previous stage.

The per-session parameters are denoted as π and described as:

- $\rho \in \{\text{init}, \text{resp}\}$: The role of the party in the current session, i.e., initiator or responder.
- $pid \in \{1, \ldots, n_P, \star\}$: The intended communication partner, where $\star$ denotes unspecified. For the unspecified case, the parameter can be updated once to a specified party $\in \{1, \ldots, n_P\}$.
- $stid \in [n_T]$: The current or most recently completed stage of the session.
- $\alpha \in \{\text{active}, \text{accept}, \text{reject}, \perp\}$: The status of the session, initialized with $\perp$.
- $m_i[stid] \in \{0,1\}^* \cup \{\perp\}$, $i \in \{s, r\}$: Array of the concatenation of messages sent (if $i = s$) or received (if $i = r$) by the session in each stage. Initialized with $\perp$ and indexed by the stage identifier *stid*.
- $k[stid] \in \{0,1\}^* \cup \{\perp\}$: Array of the session keys from each stage. The used session keys $k[i]$ to run the MAC-based protocol Π_{MAC} are removed and deleted, where the index i is the closest and non-used integer to *stid* such that $i < stid$. Initialized with $\perp$ and indexed by the stage identifier *stid*.
- $qk[stid] \in \{0,1\}^* \cup \{\perp\}$: Array of the T_{HPT}-term post-quantum secrets. Initialized with $\perp$ and indexed by the stage identifier *stid*
- $exk[stid] \in \{0,1\}^* \cup \{\perp\}$, $x \in \{q, c, s\}$: Array of the ephemeral post-quantum (if $x = q$), classical (if $x = c$) and symmetric (if $x = s$) secret keys used by the session in each stage. Initialized with $\perp$ and indexed by the stage identifier *stid*.
- $pss[stid] \in \{0,1\}^* \cup \{\perp\}$: Per-stage secret state (*SecState*) that is established during protocol execution for use in the following stage. Initialized at an arbitrary, and public, value and indexed by the stage identifier *stid*.
- $sskp[stid] \in \{0,1\}^* \cup \{\perp\}$: Array of the keys extracted from $k[i]$ to run the MAC-based protocol (Π_{MAC}, defined below), where the index i is the closest and non-used integer to *stid* such that $i < stid$. Initialized with $\perp$ and indexed by the stage identifier *stid*.
- $st[stid] \in \{0,1\}^*$: Any additional state used by the session in each stage. Indexed by the stage identifier *stid*.

The identifier of the s-th instance of the protocol Π being run by party P_i, and the collection of per-session variables maintained for the s-th instance of the protocol Π being run by party P_i, are both denoted by π_i^s in the following.

In the context of this work, three examples of protocols can be run in an alternative manner. The signature-based, the KEM-based and the MAC-based protocols from figures 2, 3 and 4 respectively, are denoted as Π_Σ, Π_{KEM} and Π_{MAC} respectively. In the following discussion, Π corresponds to any element of the set $\{\Pi_\Sigma, \Pi_{KEM}, \Pi_{MAC}\}$. Note that the first iteration of QKD has to be run with a PQC-based protocol, since the shared secret key pool is empty, thus rendering insecure the usage of the MAC-based protocol.

The key-exchange protocol Π is represented as a tuple of algorithms (f, *EQKeyGen*, *ECKeyGen*, *ESKeyGen*, *LQKeyGen*, *LCKeyGen*, *LSKeyGen*), where:

- $f(\kappa,\ \mathrm{pk}_i,\ \mathrm{sk}_i,\ \mathrm{ssid}_i,\ \mathrm{ss}_i,\ \pi,\ m) \to (m',\ \pi')$ is a probabilistic algorithm that takes a security parameter κ, the set of long-term asymmetric key pairs pk_i, sk_i of the party P_i, a collection of per-session variables π and an arbitrary bit string $m \in \{0,1\}^* \cup \{\emptyset\}$ as input, and then outputs a response $m' \in \{0,1\}^* \cup \{\emptyset\}$ and an updated per-session state π', behaving as an honest protocol implementation.
- The KeyGen algorithms are the ones described in subsection 7.1.

To investigate security, the challenger $\mathcal{C}$ runs a total number of n_P times the pertinent secret key generation algorithms, to generate public-secret key pairs (pk_i, sk_i) and/or symmetric keys with the corresponding identifiers (ss, ssid), for each party $P_i \in \{P_1, \dots, P_{n_P}\}$. All the public keys and symmetric key identifiers, pk_i and ssid, are shared with the adversary $\mathcal{A}$. The challenger $\mathcal{C}$ then samples a random bit $b \xleftarrow{\$} \{0,1\}$ and interacts with the adversary following the adversarial queries defined below. After terminating, the adversary $\mathcal{A}$ outputs a guess d of the challenger bit b. The adversary wins the HAKE indistinguishability experiment, defined in Figure 5 of the Appendix C of [15], if $d = b$, and if the test session π satisfies a cleanness predicate, also recalled below.

Since the PQC secrets are T_{HPT}-term secure, the challenger is allowed to run LQKeyGen again when required to obtain new PQC secret key material. Additionally, the adversaries are allowed to ask for the PQC secrets once the sessions are not in an active status. The challenger eventually only runs Π with certain PQC secrets if CorruptQK, defined in subsection 7.3, has not been issued before reaching an accept status.

Furthermore, the challenger $\mathcal{C}$ manages a set of corruption registers that keeps track of which secrets have been corrupted. Most of the registers are formally defined in Appendix C of [15] and new ones would follow the same structure.

Adaptations, added conditions and comments. In the context of the protocols presented in this paper, the session parameter $k[stid]$ (the array containing the session keys) corresponds to the QKD bits that are stored in the shared secret key pool once the protocols end without aborting. Note that once Π_{MAC} is run, the extracted session keys, K_0, are moved to a different array, $sskp[stid]$ to prove some post-compromise security property and to avoid re-usage of the same key.

The array $qk[stid]$ is also added, to express not only the idea that the PQC secrets are T_{HPT}-term secure, but also that the adversaries are allowed to request the PQC secrets once the test sessions reach the accept status. Eventually, the challenger creates new PQC secret material when required before making the sessions to start a new stage.

In the same way, the arrays containing the ephemeral secret keys, $exk[stid]$, update depending on which protocols have been executed for the different stages:

- If the signature-based protocol (Figure 2) is executed in stage *stid*, $esk[stid]$ is updated with the symmetric ephemeral QKD key ss_{QKD}.
- If the KEM-based protocol (Figure 3) is executed in stage *stid*, $eqk[stid]$ is updated with the post-quantum ephemeral keys, k_A and k_B, and $esk[stid]$ is updated with the symmetric ephemeral QKD key ss_{QKD}.
- If the MAC-based protocol (Figure 4) is executed in stage *stid*, $exk[stid]$ is updated with some character that specifies that there is no ephemeral key computed in stage

stid. This is because the provided example does not use ephemeral keys, however, as mentioned before, an ephemeral QKD key ss_{QKD} may be derived to provide a stronger version of post-compromise security.

The session parameter $st[stid]$ is used to contain the information regarding which protocol has been run in stage *stid*; either Π_Σ, Π_{KEM} or Π_{MAC} in this case. If Π_{MAC} is run in stage *stid*, this session parameter also provides information to determine which key K_0 from $k[stid]$ was extracted to execute the protocol: this could be done by storing some key identity parameter. This array is not relevant to the security proof and is assumed to be known by all adversaries $\mathcal{A}$.

Now that the execution environment has been recalled and adapted, the adversarial interactions are recalled and adapted next.

7.3 Adversarial Interaction

The considered adversary able to interact with the execution environment is an HPT adversary. Additionally, the conditioned unbounded adversary is allowed to interact in a limited way with both the execution environment and the HPT adversary.

Recall and adaptation of the HAKE framework. The adversaries $\mathcal{A} = \{\mathcal{A}_{\mathcal{HPT}}, \mathcal{A}_{unbounded}\}$ are allowed to act as stated in Definitions 1 and 2. That is, the HPT adversary has complete control of the classical and quantum communication network, able to modify, inject, delete or delay classical or quantum messages/signals at all times. The unbounded adversary has complete control of the quantum communication network, able to modify, inject, delete or delay quantum signals at all times. Additionally, the unbounded adversary is able to only read and delay the classical traffic.

Following these conditions, the HPT adversary $\mathcal{A}_{\mathcal{HPT}}$ can interact with the challenger $\mathcal{C}$ following the queries:

- Create$(i, j, role) \to \{(s), \perp\}$: The adversary initializes a new session owned by party P_i, with role *role*, and intended partner party P_j. If a session π_i^s has already been created, the challenger returns $\perp$. Otherwise (s) is returned.
- Send$(i, s, m) \to \{m', \perp\}$: The adversary sends a message m to an active session ($\pi_i^s.\alpha$ = active) for protocol execution and receives the honest output m' from the challenger, computed with $\Pi.f(\kappa, \text{pk}_i, \text{sk}_i, \text{ssid}_i, \text{ss}_i, \pi, m)$. The previous local (challenger side) information of the session π_i^s is updated to $\pi_i^{s'}$.
- Reveal(i, s, t): The adversary has access to the session keys computed in session π_i^s ($\pi_i^s.k[t]$) where the targeted session is in the accepted state ($\pi_i^s.\alpha$ = accept). Otherwise the challenger returns $\perp$. The challenger updates the corruption registers accordingly.
- Test$(i, s, t) \to \{k_b, \perp\}$: The adversary is provided by the challenger with the real session key ($b = 1$) or a random session key ($b = 0$), where the targeted session is in the accepted state ($\pi_i^s.\alpha$ = accept). Otherwise the challenger returns $\perp$.
- CorruptXY$(i, s, t) \to \{key, \perp\}$: The adversary is provided by the challenger with the long-term XY $\in \{SK, QK, CK\}$ keys from party P_i. The parameters s and stage t are indicated only for XY $= QK$. If the keys have been corrupted previously, then $\perp$ is returned. Specifically:
  - CorruptSK(i): Reveals the long-term symmetric secrets (if available) of party P_i. This also affects the peer party of P_i.

  - CorruptQK(i, s, t): Reveals the post-quantum T$_{HPT}$-term keys used in stage t, $\pi_i^s.qk[t]$ (if available).
  - CorruptCK(i): Reveals the classical long-term keys (if available) of party P_i.

- CompromiseXY$(i, s, t) \rightarrow \{key, \perp\}$: The adversary is provided by the challenger with the ephemeral XY $\in \{QK, CK, SK, SS, KP\}$ keys created during the session π_i^s prior to stage t. If the ephemeral key has already been compromised, then $\perp$ is returned. Specifically:
  - CompromiseQK(i, s, t): Reveals the ephemeral post-quantum keys, $\pi_i^s.eqk[t]$ (if available).
  - CompromiseCK(i, s, t): Reveals the ephemeral classical keys, $\pi_i^s.eck[t]$ (if available).
  - CompromiseSK(i, s, t): Reveals the ephemeral symmetric keys, $\pi_i^s.esk[t]$ (if available).
  - CompromiseSS(i, s, t): Reveals the per-session state, $\pi_i^s.pss[t]$ ($SecState$).
  - CompromiseKP(i, s, t): Reveals the symmetric secret extracted from the session keys (shared secret key pool) to execute Π_{MAC}, $\pi_i^s.sskp[t]$ (K_0).

Additionally, when the Test query is issued, the HPT adversary $\mathcal{A}$ interacts with the conditioned unbounded adversary $\mathcal{A}_{unbounded}$ with the following query:

- RevealITS$(i, s, t) \rightarrow \{\text{Reveal}(i, s, t), \perp\}$: If the conditioned unbounded adversary finds a way to compromise the QKD key K_{QKD}, distilled in session π_i^s at stage t, the conditioned unbounded (or the HPT, does not matter) adversary $\mathcal{A}$ is allowed to issue the Reveal(i, s, t) query to the challenger $\mathcal{C}$. Otherwise, $\perp$ is returned.

The algorithmic description of RevealITS is an if statement that activates the Reveal query in the case the conditioned unbounded adversary compromises the ITS security of the QKD keys; else nothing is returned. This RevealITS query gives virtual access to unbounded resources for the HPT adversary in order to test the ITS security of the final QKD keys, also named as the session keys in this context.

The query RevealITS returns the session keys if the QKD step has failed in the $\varepsilon_{QKD}(\kappa)$ sense. In other words, the probability that RevealITS returns the session keys is $\varepsilon_{QKD}(\kappa)$. Furthermore, if the unauthenticated has been bad implemented, RevealITS reveals always the session keys (QKD keys). Note that the conditioned unbounded adversary is able to interact with the quantum channel and with the public QKD traffic but not with the authentication steps.

Furthermore, the probability that the QKD keys are not ITS is equivalent to the probability that authentication is not secure (named ε_{auth}) or QKD fails (named ε_{QKD}), giving equation 2.

$$\Pr(\text{QKD keys not ITS}) = \varepsilon_{auth} + \varepsilon_{QKD} - \varepsilon_{auth}\varepsilon_{QKD} \leq \varepsilon_{auth} + \varepsilon_{QKD} \tag{2}$$

Where ε_{auth} is indirectly estimated in the security analysis, in section 8, and ε_{QKD} is defined in Definition 4.

Note that there is not a single way to test the ITS security of the final QKD keys. For example, the unbounded adversary could be replaced by the challenger itself who is honest regarding the ITS security. In that case, the query RevealITS would be requested to the challenger $\mathcal{C}$, rather than the conditioned unbounded adversary.

Adaptations, added conditions and comments. The traffic protection keys and the corresponding encryption are not effective against the unbounded adversary, because the confidentiality property of the presented protocols is not ITS, but is effective against bounded HPT adversaries. Since authentication, confidentiality and integrity are being tested against HPT adversaries, the conditioned unbounded adversary does not provide with the trivially compromised HPT-secure keys to the HPT adversary once the RevealITS query is issued.

The Corrupt and Compromise queries can be called for the three presented protocols Π_{Σ}, Π_{KEM} and Π_{MAC}, yielding the following:

- Π_{Σ} :
  - CorruptQK reveals the T_{HPT}-term secrets of the PQC signature.
  - CorruptSK and CorruptCK reveal nothing.
  - CompromiseSK reveals the ephemeral QKD key.
  - CompromiseSS reveals the SecState parameter.
  - CompromiseQK, CompromiseSK and CompromiseKP reveal nothing.
- Π_{KEM} :
  - CorruptQK reveals the T_{HPT}-term secrets of the PQC KEMs.
  - CorruptSK and CorruptCK reveal nothing.
  - CompromiseSK reveals the ephemeral QKD key.
  - CompromiseSS reveals the SecState parameter.
  - CompromiseQK, CompromiseSK and CompromiseKP reveal nothing.
- Π_{MAC} :
  - CorruptQK, CorruptSK and CorruptCK reveal nothing.
  - CompromiseKP reveals the symmetric secret K_0.
  - CompromiseSS reveals the SecState parameter.
  - CompromiseSK, CompromiseQK and CompromiseKP reveal nothing.

When the CompromiseSK query cannot be issued to the challenger $\mathcal{C}$ caused by a cleanness predicate condition, the HPT adversary $\mathcal{A}_{\mathcal{HPT}}$ is allowed to interact with the conditioned unbounded adversary once the unauthenticated QKD step has finalized, as stated in Definition 2, and obtain the ephemeral QKD keys with $\varepsilon_{QKD}(\kappa)$ probability.

Note that the query CorruptQK is always successfully issued after waiting the estimated time T_{HPT}.

If the Test and Reveal queries are issued together at any time, then the adversary trivially compromises the security of the protocol, as that is equivalent to explicit issue of the shared secret key pool. To avoid this kind of trivial compromise, the cleanness predicates are defined in subsection 7.5, after recalling the definitions regarding partnering.

7.4 Partnering Definition

The matching sessions [40] and origin session [41] definitions are recalled. This defines how sessions are partnered.

Definition 19. (Matching sessions). Let π_i^s and π_j^r be two sessions such that $\pi_i^s.pid = j$, $\pi_j^r.pid = i$ and $\pi_i^s.\rho \neq \pi_j^r.\rho$. Session π_i^s matches session π_j^r in stage t if $\pi_i^s.m_r[t] = \pi_j^r.m_s[t]$ and $\pi_i^s.m_s[t] = \pi_j^r.m_r[t]$, where $\pi_i^s.m_s[t]$ and $\pi_i^s.m_r[t]$ correspond to the concatenation of all messages sent and received by the session π_i^s in stage t, respectively.
Additionally, the session π_i^s prefix-matches session π_j^r in stage t if $\pi_j^r.m_s[t] = \pi_i^s.m_r[t]'$, where $\pi_i^s.m_r[t]'$ is $\pi_i^s.m_r[t]$ truncated to the length of $\pi_j^r.m_s[t]$.

Note that π_i^s having a matching or prefix-matching session with π_j^r is equivalent to π_j^r undertaking a matching or prefix-matching session with π_i^s. If the sessions π_i^s and π_j^r are matching, then the sessions are also prefix-matching.

Definition 20. (Origin sessions). Let π_i^s and π_j^r be two sessions such that $\pi_i^s.pid = j$, $\pi_j^r.pid = i$ and $\pi_i^s.\rho \neq \pi_j^r.\rho$. Session π_i^s has an origin session with π_j^r if π_i^s matches or prefix-matches session with π_j^r.

A session π_i^s that has an origin session with π_j^r implies that π_i^s has received the messages that π_j^r intended to send. However, that does not imply that π_j^r is undertaking an origin session with π_i^s. Additionally, the sessions match if all messages sent and received by both sessions are identical: for this case, both sessions are undertaking an origin session with each other. The origin session distinction is necessary to include the case scenarios where an adversary starts injecting, deleting, or substituting the classical traffic for at least one of the sessions, where both sessions were matching beforehand.

The cleanness predicates are defined next.

7.5 Cleanness Predicates

The cleanness predicates define the conditions for which an adversary $\mathcal{A}$ does not break the security of Π. The main goal is to show that the presented examples of protocols are secure against an HPT adversary. Furthermore, once any of the protocols are in an accept state (or failed state, different from active state), the HPT adversary is allowed to issue RevealITS to an unbounded adversary, to verify that the obtained QKD keys are ITS secure, where the unbounded adversary was able to interfere with the unauthenticated QKD step and only observe the authentication steps. Note that if an adversary is permanently unbounded, the presented algorithms are not secure: the only viable way of making QKD secure in this case would be to properly use ITS-MACs with pre-shared key. Thus, the unbounded adversary is defined such as being able to read all classical data and returns either the Reveal and CompromiseSK queries, or nothing, without revealing more to the HPT adversary, in function of if the sessions keys are ITS or not.

In this case, the cleanness predicates can be reduced to "if authentication is secure, then the session keys are secure". Thus, secrecy of at least one of the secrets involving authentication has to be guaranteed for when a protocol Π is in an active state.

The cleanness predicates also capture perfect forward secrecy. The nature of a secure QKD protocol implies direct perfect forward secrecy, since the last post-processing step of QKD, privacy amplification, ensures that the individual QKD bits contain a min-entropy close to unity. Thus, revealing all the secrets but the actual QKD bits (and the QKD secrets such as the raw data encoded and decoded by respectively Alice and Bob) does not reveal the QKD bits distilled from previous stages, i.e. the protocols have perfect forward secrecy, secure against unbounded adversaries that interact with the public material once the protocols are finalized (in an accepted state). This is expressed in the condition 3 (below) of the cleanness predicate, by allowing the adversary to compromise all of the secret keys once the test sessions reach an *accept* state.

Post-compromise security is also achieved for all HPT adversaries in the sense that the adversary can compromise all ephemeral secrets of a particular stage without compromising the security of the protocols, as long as there exists some previous stage that has not had the QKD ephemeral secret compromised and the adversary has been passive in all stages between the "Test" stage and the previous "clean" stage [42], expressed in condition 3 of the cleanness predicate definition below as well.

Definition 21. (clean$_{HPT}$). Let $\pi_i^s[t]$ be a $\Pi \in \{\Pi_\Sigma, \Pi_{KEM}, \Pi_{MAC}\}$ session in stage t such that $\pi_i^s.\alpha[t] = accept$ and $\pi_i^s.pid = j$. The Π session π_i^s in stage t is clean$_{HPT}$ against an HPT adversary if all of the following conditions hold:

1. Reveal(i, s, t) has not been issued.
2. For all $(j, r, t) \in n_P \times n_S \times n_T$ such that π_i^s matches π_j^r in stage t, Reveal(j, r, t) has not been issued.
3. If there exists no $(j, r, t) \in n_P \times n_S \times n_T$ such that π_j^r is an origin session of π_i^s in stage t, then one of the following set of queries has not been issued before $\pi_i^s.\alpha[t] \leftarrow accept$:
   - CorruptQK(i, s, t) and CorruptQK(j, r, t).
   - CompromiseKP(i, s, t) and CompromiseKP(j, r, t).
   - CompromiseSK(i, s, t'), CompromiseSK(j, r, t') or CompromiseKP(i, s, t'), CompromiseKP(j, r, t') and CompromiseSS(i, s, u), CompromiseSS(j, r, u), with $t' \leq u < t$, and π_i^s matches π_j^r in stages u.

   If there exists a $(j, r, t) \in n_P \times n_S \times n_T$ such that π_j^r is an origin session of π_i^s in stage t, then one of the following set of queries has not been issued before $\pi_j^r.\alpha[t] \leftarrow accept$:
   - CorruptQK(i, s, t) and CorruptQK(j, r, t).
   - CompromiseKP(i, s, t) and CompromiseKP(j, r, t).
   - CompromiseSK(i, s, t'), CompromiseSK(j, r, t') or CompromiseKP(i, s, t'), CompromiseKP(j, r, t') and CompromiseSS(i, s, u), CompromiseSS(j, r, u), with $t' \leq u < t$, and π_i^s matches π_j^r in stages u.
4. The protocol Π_{MAC} is run only when $\pi_i^s.k$ has enough key material to extract K_0, where π_i^s is a matching session of π_j^r in stages $t' < t$ and $\pi_i^s.k[t'] = \pi_j^r.k[t']$.

Condition 1. prevents the session keys being revealed to the considered adversary via the Reveal query, since this trivially compromises the security of any protocol Π.

Condition 2. additionally prevents that the session keys of the matching session are targeted by the Reveal query as well, since matching sessions share the same session keys at the end of the accepted protocols.

Conditions 1. and 2. impose that the challenger $\mathcal{C}$ never issues the Reveal query to the HPT adversary $\mathcal{A}$. However, note that the query RevealITS can still be issued by the HPT adversary to the unbounded adversary, bypassing conditions 1. and 2., making the session $\pi_i^s[t]$ not clean in the case that QKD keys are not ITS in the first place.

Condition 3. ensures that at least one of the secrets used to authenticate within the protocols is not known to the HPT adversary $\mathcal{A}$. This makes authentication secure against substitution and impersonation attacks. Recall that after the physical time T_{HPT} has elapsed, the T_{HPT}-term PQC secrets are revealed by the challenger $\mathcal{C}$, which is equivalent to the HPT adversary $\mathcal{A}$ issuing a CorruptQK query. If CorruptQK is issued once the sessions reach accept status, the parties generate new PQC material in order to proceed to the next stage.

Condition 4. ensures that the protocol Π_{MAC} is run only when the shared secret key pool has enough key material to extract the key K_0. Since the shared secret key pool is empty for stage t, a PQC-based protocol (Π_Σ or Π_{KEM}) has to be realized until the shared secret key pool has enough key material to realize successfully Π_{MAC}. If the adversary decides to perform Denial of Service (DoS) attacks, this could exhaust the shared secret key pool from session π_i^s, since K_0 is assumed to be fresh every time the protocol Π_{MAC} is executed. In that case, the PQC-based protocol has to be run again to replenish the key pool once again. If the goal is to rely as little as possible on PQC, then ideally it would be desirable to have sufficient distilled key at stage $t = 1$, i.e. at the first QKD iteration, to essentially prevent DoS attacks. Once Π_{MAC} is finalized, more keys are stored inside the key pool that can in turn be used to execute Π_{MAC} again. The party behind session π_i^s would also need to communicate to its peer π_j^r to delete the used keys from the shared secret key pool. This could be done by exchanging authenticated identity parameters of the used keys that now have to be discarded. Despite being an extensive subject, no more discussion is provided regarding the DoS attack in this paper.

Now that the cleanness predicate clean_{HAKE} is defined, it is convenient to recall the advantage of an algorithm $\mathcal{A}$ in winning the HAKE key indistinguishability experiment:

Definition 22. (HAKE key indistinguishability). Let Π by a key-exchange protocol, and n_P, n_S, $n_T \in \mathbb{N}$. For a particular given predicate clean, and an algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the HAKE key-indistinguishability is defined as:

$$\mathrm{Adv}_{\Pi,n_P,n_S,n_T}^{\mathrm{HAKE,clean},\mathcal{A}}(\kappa) = 2\left|\Pr\left[\mathrm{Exp}_{\Pi,n_P,n_S,n_T}^{\mathrm{HAKE,clean},\mathcal{A}}(\kappa) = 1\right] - \frac{1}{2}\right| ,$$

where the experiment $\mathrm{Exp}_{\Pi,n_P,n_S,n_T}^{\mathrm{HAKE,clean},\mathcal{A}}(\kappa)$ is defined in Appendix C, figure 5 of the Muckle paper [15] and κ is a security parameter, and where RevealITS, defined in this work, also enters in play to test ITS security of the session keys. Π is HPT HAKE-secure if, for all HPT adversaries $\mathcal{A}$, $\mathrm{Adv}_{\Pi,n_P,n_S,n_T}^{\mathrm{HAKE,clean},\mathcal{A}}(\kappa)$ is negligible in the security parameter κ.

As the HPT adversaries have access to the RevealITS query in the presented framework, if Π is HPT HAKE-secure, then the session keys (QKD keys) are ITS too.

8 Security Analysis

The security of the presented protocols $\Pi \in \{\Pi_\Sigma, \Pi_{KEM}, \Pi_{MAC}\}$ is evaluated through the cleanness predicate clean_{HPT} and Definition 22.

Furthermore, the confidentiality property given by the traffic keys and AEAD is also taken into account. The advantages for the HAKE key-indistinguishability game and the advantages related to confidentiality are summed as given by the union bound. The following theorem 23 accounts for both properties.

Theorem 0. The multi-stage and dynamic $\Pi \in \{\Pi_\Sigma, \Pi_{KEM}, \Pi_{MAC}\}$ key exchange protocol is HAKE-secure with cleanness predicate clean_{HPT}, and confidential secure, where unauthenticated QKD follows Definition 5, PRF is a dual PRF algorithm (Definition 7), H is a weak collision resistant hash function (Definition 8), Σ is a EUF-CMA secure signature algorithm (Definitions 9 and 10), KEM is IND-CCA secure KEM algorithm (Definitions 11 and 12) and MAC is a EUC-CMA secure MAC algorithm (Definition 13 and 14). That is, for any adversary $\mathcal{A}$ (Definitions 1 and 2) against the HAKE key-indistinguishability game (Definition 22), $\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,n_T}(\kappa)$ is negligible in the security parameter κ, where:

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,n_T}(\kappa) \;\leq\;& \\
& 2n_P^2 n_S n_{T_\Sigma}\left[\mathrm{Adv}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa) + 6\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\right] \\
+\;& 2n_P^2 n_S^2 n_{T_\Sigma}^2\left[\varepsilon_{QKD}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa)\right. \\
+\;& (1+n_{T_\Sigma})\left\{5\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\right. \\
+\;& \left.\left.\mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 6\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\right\}\right] \\[1ex]
+\;& 2n_P^2 n_S n_{T_{KEM}}\left[\mathrm{Adv}^{\text{IND-CCA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa)\right. \\
+\;& 4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
+\;& \left.\mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 5\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\right] \\
+\;& 2n_P^2 n_S^2 n_{T_{KEM}}^2\left[\varepsilon_{QKD}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa)\right. \\
+\;& (1+n_{T_{KEM}})\left\{\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + 2\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\right. \\
+\;& \left.\left.2\mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 5\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\right\}\right] \\[1ex]
+\;& 2n_P^2 n_S n_{T_{MAC}}\left[\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,(n'_T|K_0\text{ derivation stage})}(\kappa)\right. \\
+\;& \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) + 3\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
+\;& \left.\mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\right] \\
+\;& 2n_P^2 n_S^2 n_{T_{MAC}}^2\left[\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,(n'_T|K_0\text{ derivation stage})}(\kappa)\right. \\
+\;& \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) + 4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
+\;& \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) \\
+\;& n_{T_{MAC}}\left\{5\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\right. \\
+\;& \left.\left.\mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\right\}\right] \\[1ex]
+\;& n_P^2 n_S^2 n_T\left[2\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \varepsilon_{QKD}(\kappa)\right] \\
+\;& n_P^2 n_S^2 n_{T_{KEM}} 2\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)
\end{aligned}$$

Proof. As performed in the Muckle series, the proof regarding the HAKE key-indistinguishability game is divided into three separate cases where the query $\mathrm{Test}(i, s, t)$ has been issued:

1. The session π_i^s (where $\pi_i^s.\rho = \mathit{init}$) has no origin session in stage t.
2. The session π_i^s (where $\pi_i^s.\rho = \mathit{resp}$) has no origin session in stage t.

3. The session π_i^s in stage t has a matching session.

Cases 1 and 2 verify that Π remains secure against attacks that force sessions to remain unmatched. Both cases encapsulate the idea of an active adversary substituting, deleting or injecting messages. Case 3 ensures that Π is secure for sessions where the transcripts fully match or prefix-match. This case takes into account a passive adversary that is only reading the traffic. The three cases together cover all adversarial scenarios relevant to the protocol, protecting against a wide variety of attacks, including both impersonation and substitution attacks, and thus provide comprehensive security guarantees for the scheme.

Additionally, since Π could involve three different protocols, Π_Σ, Π_{KEM} and Π_{MAC}, the security proof has to take into account the fact that a session can choose between one of the three algorithms at any time (with the conditions that at stage $t = 1$ and condition 4. from clean_{HPT}, Π_{MAC} is not run). This implies that the security of the three protocols have to be evaluated individually at some point of the proof. This is done by dividing the number of stages n_T into n_{T_Σ}, $n_{T_{KEM}}$ and $n_{T_{MAC}}$, where $n_{T_\Sigma} + n_{T_{KEM}} + n_{T_{MAC}} = n_T$. This division takes into account the number of stages where each protocol has been executed. Hence, each case out of the three cases that decomposes the HAKE key-indistinguishability game present three more sub-cases: one for each protocol in Π.

It follows that:

$$\begin{aligned}
\text{Adv}_{\Pi,n_P,n_S,n_T}^{\text{HAKE,clean}_{HPT},\mathcal{A}}(\kappa) &= \text{Adv}_{\Pi,n_P,n_S,n_T}^{\text{HAKE,clean}_{HPT},\mathcal{A},C_1}(\kappa) \\
&+ \text{Adv}_{\Pi,n_P,n_S,n_T}^{\text{HAKE,clean}_{HPT},\mathcal{A},C_2}(\kappa) \\
&+ \text{Adv}_{\Pi,n_P,n_S,n_T}^{\text{HAKE,clean}_{HPT},\mathcal{A},C_3}(\kappa)
\end{aligned}$$

Where $\text{Adv}_{\Pi,n_P,n_S,n_T}^{\text{HAKE,clean}_{HPT},\mathcal{A},C_\#}(\kappa)$ corresponds to the advantage of the HPT adversary $\mathcal{A}$ in winning the key-indistinguishability game in case #. Furthermore,

$$\begin{aligned}
\text{Adv}_{\Pi,n_P,n_S,n_T}^{\text{HAKE,clean}_{HPT},\mathcal{A},C_\#}(\kappa) &= \text{Adv}_{\Pi,n_P,n_S,n_{T_\Sigma}}^{\text{HAKE,clean}_{HPT},\mathcal{A},C_\#}(\kappa) \\
&+ \text{Adv}_{\Pi,n_P,n_S,n_{T_{KEM}}}^{\text{HAKE,clean}_{HPT},\mathcal{A},C_\#}(\kappa) \\
&+ \text{Adv}_{\Pi,n_P,n_S,n_{T_{MAC}}}^{\text{HAKE,clean}_{HPT},\mathcal{A},C_\#}(\kappa)
\end{aligned}$$

In total, nine advantages have to be evaluated (three cases, for each of three protocols). However, since case 1 and case 2 present the same bounds given a protocol and case 3 can be easily treated in the same way for the three protocols, four total advantages are actually evaluated (one case for three protocols and one case for one protocol).

Here is an example of how to evaluate the advantages for different protocols. Assume that two parties are running a session with four stages where they decided to run the protocols Π_{KEM}, Π_{MAC}, Π_Σ and Π_{MAC} in stages 1, 2, 3 and 4 respectively. The total security levels for the different stages are:

- Stage 1: $\text{Adv}_{\Pi,n_P=2,n_S=1,n_T=1}^{\text{HAKE,clean}_{HPT},\mathcal{A}}(\kappa) = \text{Adv}_{\Pi,n_P=2,n_S=1,n_{T_{KEM}}=1}^{\text{HAKE,clean}_{HPT},\mathcal{A}}(\kappa)$
- Stage 2: $\text{Adv}_{\Pi,n_P=2,n_S=1,n_T=2}^{\text{HAKE,clean}_{HPT},\mathcal{A}}(\kappa) = \text{Adv}_{\Pi,n_P=2,n_S=1,n_{T_{KEM}}=1}^{\text{HAKE,clean}_{HPT},\mathcal{A}}(\kappa)$ $+\text{Adv}_{\Pi,n_P=2,n_S=1,n_{T_{MAC}}=1}^{\text{HAKE,clean}_{HPT},\mathcal{A}}(\kappa)$

- Stage 3: $\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_T=3}(\kappa) = \mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_{T_{KEM}}=1}(\kappa)$ $+\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_{T_{MAC}}=1}(\kappa) + \mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_{T_\Sigma}=1}(\kappa)$
- Stage 4: $\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_T=4}(\kappa) = \mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_{T_{KEM}}=1}(\kappa)$ $+\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_{T_{MAC}}=2}(\kappa) + \mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_{T_\Sigma}=1}(\kappa)$

The security proof is performed by starting with the original HAKE security game (the presented protocols as they are) and aiming for an unwinnable game (by modifying little by little the presented protocols with sound replacements), reached by game-hopping [43, 44], whilst making sure that the session is clean$_{HPT}$ according to the defined cleanness predicates. This is typically done by stating what key material can be compromised and then evaluate the chance of the defined adversary to win the modified games by replacing the non-compromised key material by uniformly random bit strings; the capacity of an adversary to differentiate between Games is given by the defined cryptographic primitives advantages, given in the Definitions section 3. Additionally, confidentiality of the traffic is also evaluated and included within the security parameters.

Since the unauthenticated QKD step is assumed to follow Definition 5, i.e. the unauthenticated QKD protocol is the same as a regular QKD protocol that has a secure authenticated channel, it suffices to show that authentication is ε_{auth}-secure to make the final QKD keys ITS and $(\varepsilon_{auth} + \varepsilon_{QKD})$-secure, according to Definition 6. Additionally, ε_{auth} contains the security that keeps confidentiality and integrity intact, as given by the traffic protection keys and the MAC tags, respectively.

A reminder that the PQC secrets are used when the physical time is less than an estimation of T$_{HPT}$. Once the physical time has surpassed T$_{HPT}$ or the CorruptQK query has been issued when the test sessions reach the accept state, new PQC secrets have to be generated.

In all cases and sub-cases, the game-hopping proof starts with the same Game 0 defined as:

Game 0: This is the original HAKE security game.

$$\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,n_{T_\Sigma}}(\kappa) = \Pr(break_0)$$

8.1 Case 1: Test *init* session without origin session

As performed the in Muckle series, the game hops will converge to a game that takes all the potential vulnerabilities of the cryptographic primitives used, where the adversary $\mathcal{A}$ has a negligible chance to win, to afterwards hop to a final unwinnable game, where the challenger $\mathcal{C}$ always reject, regardless of the correctness of the cryptographic primitives. Specifically, Case 1 (and Case 2) security analysis shows that $\mathcal{A}$ has negligible chance in causing the test sessions π_i^s (or π_j^r respectively) to reach an accept state without an origin session. Both Cases take into account substitution, injection and deleting attacks, meaning that the test sessions π_i^s may or may not already have started undertaking a matching session with an intended partner session π_j^r at stage $t' < t$, but in stage t the adversary stops the matching session by interacting with the traffic such that the test sessions no longer have an origin session.

Game 1 is the same for all of the sub-cases for Case 1 and Case 2, unless mentioned otherwise:

Game 1: The indices (i, s, t), corresponding to the test session π_i^s, the stage stage t, and the intended partner j are guessed. If the query Test(i', s', t') is issued to a session $\pi_{i'}^{s'}$, where $\pi_{i'}^{s'}.pid = j'$, and if $(i, s, t, j) \neq (i', s', t', j')$, then the game aborts, yielding the following in the worst case scenario:

$$\Pr(break_0) \leq n_P^2 n_S n_T \Pr(break_1)$$

This game takes into account not only the scenarios where the adversary wants to impersonate party j from the start, but also replay and substitution attacks. This is because the security against the mentioned attacks relies solely on the cryptographic primitives and not on the timing of the adversarial attack. The collision resistance probability is also taken into account for all sub-cases but the MAC case, since the MAC input presents a counter that provides uniqueness for the input of the cryptographic primitives. Furthermore, entity protection and cipher protection are also taken in account, regardless of if this compromises or not the final session key (QKD keys).

8.1.1 Case 1: Sub-case Σ, CorruptQK is not issued at stage t before *accept*

The security of the Π_Σ protocol is evaluated in the case that the CorruptQK query has not been issued.

Games 0 and 1: Defined right before and right after subsection 8.1 respectively.

Game 2: If the targeted test sessions reach the reject state, $\pi_i^s.\alpha[t] = reject$, then this game aborts. In Game 1, if the targeted test sessions reach a reject state, then $\perp$ is returned by the Test query, making Game 1 and Game 2 equivalent. Since the adversary $\mathcal{A}$ gains no additional advantage compared to Game 1, it yields that:

$$\Pr(break_1) = \Pr(break_2)$$

Game 3: If the targeted sessions reach the accept state, $\pi_i^s.\alpha[t] = accept$, then this game aborts. Note that the probability of winning Game 3 is exactly zero since the Test query always returns $\perp$, caused by the constant abortion. Furthermore, Game 2 and Game 3 are identical when the adversary $\mathcal{A}$ fails to successfully forge or distinguish any of the cryptographic primitives or key material, thus by the Difference Lemma (Lemma 1 from [43]), this yields:

$$|\Pr(break_2) - \Pr(break_3)| \leq \Pr(abort)$$
$$\Rightarrow \Pr(break_2) \leq \Pr(abort)$$

where $\Pr(break_3) = 0$ and $\Pr(abort)$ is the probability that the session aborts, i.e. $\pi_i^s.\alpha[t] = accept$. The following Game(s) replace(s) the vulnerable primitives or key material by uniformly random and independent values to upper-bound $\Pr(abort)$. On this occasion, only the signature tag is targeted since the rest of the primitives and tags are trivially vulnerable since the HPT adversary $\mathcal{A}_{\mathcal{HPT}}$ is allowed to issue the pertinent queries while keeping the test session clean.

Game 4: The T_{HPT}-term secret sk_B is replaced with a uniformly random and independent value, $\widetilde{\text{sk}_B}$, of the same length. The signature tag σ_B is then computed with $\Sigma.\text{Sign}(\widetilde{\text{sk}_B}, \text{l}_{s_B}||\text{H}(\text{m}_{QKD}||\text{m}_1))$. Since sk_B is itself uniformly random and independent, and the HPT adversary $\mathcal{A}$ has not issued either $\text{CorruptQK}(i, j)$ or $\text{CorruptQK}(j, i)$ queries, this change is a sound replacement. If the session π_i^s reaches the accept state, then the adversary must have forged a valid signature tag σ_B (following Experiment 3 from Definition 10). Additionally, the adversary is allowed to reuse requested signature tags but changing the message to be authenticated, to try to bypass the burden of finding the secret key $\widetilde{\text{sk}_B}$. In that case, if the session π_i^s reaches the accept state, then a hash collision must has been found. Actually, the adversary can target six different digests: $\text{H}(\text{m}'_{QKD}, \text{m}_1 \text{ to } \text{m}_i)$, where m'_{QKD} corresponds to the QKD traffic created by the authenticated interaction between the test party and the adversary and $i \in \{2, 3, 4, 6, 7, 8\}$. To prevent any attack that targets the digests, a conservative bound is taken where all lines involving the hash function are considered. According to Definitions 8 and 10 and the union bound, it follows that:

$$\Pr(abort) \leq \text{Adv}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa)$$

$$\Pr(break_0) \leq n_P^2 n_S n_{T_\Sigma} \left[\text{Adv}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa) + 6\,\text{Adv}_{\text{H},\mathcal{A}}(\kappa)\right]$$

The conditioned unbounded adversary cannot interact (replace or substitute) with the nonces since this is not part of the unauthenticated QKD step. Additionally, if the nonces are concatenated to the labels, instead of putting them inside the hash function, the collision vulnerability becomes nonexistent.

Note that the adversaries $\mathcal{A}$ can issue the query CorruptQK at any time, as long as the test sessions reach the accept state. Since the PQC secrets are used to authenticate the QKD traffic without affecting the QKD step itself, issuing CorruptQK under the accept state condition does not affect the final advantage of winning the HAKE security game.

8.1.2 Case 1: Sub-case Σ, CompromiseSK are not issued at stage t' and CompromiseSS are not issued at stages u, $t' \leq u < t$

Game 0: Defined immediately before subsection 8.1.

Game 1: Same as Game 1 defined immediately after subsection 8.1, but the adversary has to additionally guess the session π_j^r in which the test session π_i^s has matched in stages u, and taking into account the guessed stage t' where the adversary $\mathcal{A}$ has not issued $\text{CompromiseSK}(i, s, t')$ or $\text{CompromiseSK}(j, r, t')$, and $\text{CompromiseSS}(i, s, u)$ or $\text{CompromiseSS}(j, r, u)$ have been issued, where the test session π_i^s matches session π_j^r in stages u, where $t' \leq u < t$:

$$\Pr(break_0) \leq n_P^2 n_S^2 n_{T_\Sigma}^2 \Pr(break_1)$$

Games 2 and 3: Same as sub-case Σ, CorruptQK is not issued at stage t before *accept*, section 8.1.1.

$$\Pr(break_1) \leq \Pr(break_2) \leq \Pr(abort)$$

Game 4: The ephemeral QKD key is replaced with a uniformly random and independent value, $\widetilde{ss_{QKD}}$, of the same length. The replacement is performed with an unauthenticated QKD challenger that replaces the public traffic m_{QKD} with m'_{QKD}, which is used to distill $\widetilde{ss_{QKD}}$. Since π_i^s matches session π_j^r in stage t', the replaced traffic m'_{QKD} is received by both sessions without modification. The detection of this replacement by $\mathcal{A}$ implies having an efficient distinguishing algorithm against the indistinguishability security of the unauthenticated QKD, where the algorithm does not target authentication.

$$\Pr(abort) \leq \varepsilon_{QKD}(\kappa) + \Pr(break_4)$$

Game 5: The intermediate key $K_0 \leftarrow \mathrm{PRF}(\widetilde{ss_{QKD}}, l_0||\mathrm{H}(\mathrm{m})$ is replaced with a uniformly random and independent value, $\widetilde{K_0}$, of the same length. The replacement is performed with a PRF challenger after querying $l_0||\mathrm{H}(\mathrm{m})$, where m and H are the pertinent public data. Since $\widetilde{ss_{QKD}}$ is uniformly random and independent by Game 4, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the indistinguishability security of the PRF, thus:

$$\Pr(break_4) \leq \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_5)$$

Game 6: The intermediate key $K_1 \leftarrow \mathrm{PRF}(SecState, l_1||\widetilde{K_0})$ is replaced with a uniformly random and independent value, $\widetilde{K_1}$, of the same length. The replacement is performed with a dual PRF challenger after querying $SecState$ and l_1. Since $\widetilde{K_0}$ is uniformly random and independent by Game 5, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the indistinguishability security of the dual PRF, thus:

$$\Pr(break_5) \leq \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) + \Pr(break_6)$$

Games 7 and 8: The traffic key $K_{TS_B} \leftarrow \mathrm{PRF}(\widetilde{K_1}, l_{TS_B}||\mathrm{H}(\mathrm{m})$ and the MAC key $K_{MAC_B} \leftarrow \mathrm{PRF}(\widetilde{K_1}, l_{MAC_B}||\mathrm{H}(\mathrm{m})$ are replaced with uniformly random and independent values, $\widetilde{K_{TS_B}}$ and $\widetilde{K_{MAC_B}}$, of the same length, respectively. These replacements are performed with a PRF challenger after querying $l_{TS_B}||\mathrm{H}(\mathrm{m})$ and $l_{MAC_B}||\mathrm{H}(\mathrm{m})$, where m and H are the pertinent public data. Since $\widetilde{K_1}$ is uniformly random and independent by Game 6, these are sound replacements. The detection of these replacements by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the indistinguishability security of the PRF, thus:

$$\Pr(break_6) \leq 2\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_8)$$

Since confidentiality is also desired, the AEAD advantages regarding IND-CPA are taken into account. Additionally, the adversary not only has to forge what is inside the AEAD ciphertext, but also has to generate a valid AEAD ciphertext that the decryption process accepts (related to INT-CTXT security).

Game 9: Same as Game 7 but with the traffic key K_{TS_A}.

$$\Pr(break_8) \leq \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_9)$$

Games 10 and 11: The AEAD ciphers $\{\mathrm{B}, \sigma_B, \tau_B\}_{K_{TS_B}}$ and $\{\mathrm{A}, \sigma_A, \tau_A\}_{K_{TS_A}}$ are replaced with uniformly random and independent values, $\widetilde{\{\mathrm{B}, \sigma_B, \tau_B\}_{K_{TS_B}}}$ and $\widetilde{\{\mathrm{A}, \sigma_A, \tau_A\}_{K_{TS_A}}}$, of the same length, respectively. The replacement of $\{\mathrm{A}, \sigma_A, \tau_A\}_{K_{TS_A}}$ is performed with a IND-CPA AEAD challenger with query $\mathrm{A}, \sigma_A, \tau_A$, whereas the replacement of $\{\mathrm{B}, \sigma_B, \tau_B\}_{K_{TS_B}}$ is performed with a INT-CTXT AEAD challenger with query $\mathrm{B}, \sigma_B, \tau_B$. Since $\widetilde{K_{TS_B}}$ and $\widetilde{K_{TS_A}}$ are uniformly random and independent by Games 7 and 8, respectively, these are sound replacements. The detection of these replacements by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the INT-CTXT or IND-CPA security of the AEAD algorithm, thus:

$$\Pr(break_9) \leq \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \Pr(break_{11})$$

Game 12: If the targeted session π_i^s accepts without an origin session in stage t, then this game aborts. The abortion condition is met once the final MAC tag given by no honest session π_j^r is accepted by π_i^s. The accept condition occurs when a EUF-CMA MAC challenger computes the MAC tag $\tau_B \leftarrow \mathrm{MAC.Auth}(\widetilde{K_{MAC_B}}, l_{MAC_B}||\mathrm{H(m)})$ for π_i^s by querying the pertinent message m. Since $\widetilde{K_{MAC_B}}$ is uniformly random and independent by Game 8, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT forging algorithm against the EUF-CMA security of the MAC algorithm, thus:

$$\Pr(break_{11}) \leq \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + \Pr(break_{12})$$

Game 13: The secret state $SecState \leftarrow \mathrm{PRF}(\widetilde{K_1}, l_{SecState}||\mathrm{H(m)})$ is replaced with a uniformly random and independent value, $\widetilde{SecState}$, of the same length. The replacement is performed with a PRF challenger after querying $l_{SecState}||\mathrm{H(m)}$, where m and H are the pertinent public data. Since $\widetilde{K_1}$ is uniformly random and independent by Game 6, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the indistinguishability security of the PRF, thus:

$$\Pr(break_{12}) \leq \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_{13})$$

The next game hops involving the secret state $SecState$ are repeated $(t - t')$ times in each consecutive stage u where $t' < u \leq t$. Recall that there is a matching session π_j^r with π_i^s for all the stages following stage t'.

Game 14: The intermediate key $K_1 \leftarrow \mathrm{PRF}(\widetilde{SecState}, l_1||K_0)$ is replaced with a uniformly random and independent value, $\widetilde{K_1}$, of the same length. The replacement is performed with a PRF challenger after querying $l_1||K_0$. Since $\widetilde{SecState}$ is uniformly random and independent by the previous game, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the indistinguishability security of the PRF, thus:

$$\Pr(break_{13}) \leq n_T\{\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_{14})\}$$

Games 15 to 21 : Same as Games 7 to 13 respectively.

$$\Pr(break_{14}) \leq 4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)$$

To include potential attacks where collision of the digests could be exploited, as done in the sub-case Σ, CorruptQK is not issued at stage t before *accept*, section 8.1.1, the union bonds of the advantages related to the collision probability are also included. That is:

$$\begin{aligned}
\Pr(break_{13}) &\leq 6\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_{14}) \\
\Pr(break_{21}) &\leq 6\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)
\end{aligned}$$

Since there are no more primitives or key material that the adversary can target, the final advantage is bounded.

Through combining all the inequalities together, inequality 3 follows.

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A},C_1}_{\Pi,n_P,n_S,n_{T_\Sigma}}(\kappa) \leq\;& n_P^2 n_S n_{T_\Sigma}\big[\mathrm{Adv}^{\text{EUF-CMA}}_{\Sigma,\mathcal{A}}(\kappa) + 6\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\big] \\
+\;& n_P^2 n_S^2 n_{T_\Sigma}^2\big[\varepsilon_{QKD}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) \\
+\;& (1+n_{T_\Sigma})\{5\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
+\;& \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 6\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\}\big]
\end{aligned} \tag{3}$$

The sub-case for Π_{KEM} is treated next.

8.1.3 Case 1: Sub-case KEM, CorruptQK is not issued at stage t before *accept*

The security analysis follows analogously to what has been performed with the signature case.

Games 0 and 1: Defined immediately before and immediately after subsection 8.1, respectively.

$$\Pr(break_0) \leq n_P^2 n_S n_{T_{KEM}} \Pr(break_1)$$

Games 2 and 3: Same as sub-case Σ, CorruptQK is not issued at stage t before *accept*, section 8.1.1, respectively.

$$\Pr(break_1) \leq \Pr(break_2) \leq \Pr(abort)$$

Game 4: The T_{HPT}-term secret k_B is replaced with a uniformly random and independent value, $\widetilde{\mathrm{k}_B}$, of the same length. The replacement is performed with a IND-CCA KEM challenger where the challenger's public key pk'_B and the cipher output c'_B takes the place of the public key pk_B and cipher c_B in messages m_3 and m_4, respectively. If the session π_i^s reaches the accept state, then the adversary must have successfully decapsulated $\widetilde{\mathrm{k}_B}$. As the queries CorruptQK(i,j) or CorruptQK(j,i) have not been issued, it follows that (following Experiment 4 from Definition 12):

$$\Pr(abort) \leq \mathrm{Adv}^{\text{IND-CCA}}_{\text{KEM},\mathcal{A}}(\kappa) + \Pr(break_4)$$

Game 5: The intermediate key $K_2 \leftarrow \mathrm{PRF}(K_1, l_2||\widetilde{\mathrm{k}_B})$ is replaced with a uniformly random and independent value, $\widetilde{K_2}$, of the same length. The replacement is performed with a dual PRF challenger after querying K_1 and l_2. Since $\widetilde{\mathrm{k}_B}$ is uniformly random and independent by the previous game, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the indistinguishability security of the dual PRF, thus:

$$\Pr(break_4) \leq \mathrm{Adv}^{\text{ind}}_{\text{PRF}^{\text{dual}},\mathcal{A}}(\kappa) + \Pr(break_5)$$

Games 6 and 7: The traffic keys $K_{TS_{A2}} \leftarrow \mathrm{PRF}(\widetilde{K_1}, l_{TS_{A2}}||\mathrm{H(m)})$ and $K_{TS_{B2}} \leftarrow \mathrm{PRF}(\widetilde{K_1}, l_{TS_{B2}}||\mathrm{H(m)})$ are replaced with uniformly random and independent value, $\widetilde{K_{TS_{A2}}}$ and $\widetilde{K_{TS_{B2}}}$, of the same length, respectively. The replacements are performed with a PRF challenger after querying $l_{TS_{A2}}||\mathrm{H(m)}$ and $l_{TS_{B2}}||\mathrm{H(m)}$, where m and H are the pertinent public data. Since $\widetilde{K_1}$ is uniformly random and independent by the previous game, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the indistinguishability security of the PRF, thus:

$$\Pr(break_5) \leq 2\mathrm{Adv}^{\text{ind}}_{\text{PRF},\mathcal{A}}(\kappa) + \Pr(break_7)$$

Games 8 and 9: The AEAD ciphers $\{\mathrm{c}_A, \tau_B\}_{\widetilde{K_{TS_{B2}}}}$ and $\{\mathrm{c}_B, \tau_A\}_{\widetilde{K_{TS_{A2}}}}$ are replaced with uniformly random and independent values, $\widetilde{\{\mathrm{c}_A, \tau_B\}_{K_{TS_{B2}}}}$ and $\widetilde{\{\mathrm{c}_B, \tau_A\}_{K_{TS_{A2}}}}$, of the same length, respectively. The replacement of $\{\mathrm{c}_B, \tau_A\}_{K_{TS_{A2}}}$ is performed with a IND-CPA AEAD challenger with query c_B, τ_A, whereas the replacement of $\{\mathrm{c}_A, \tau_B\}_{K_{TS_{B2}}}$ is performed with a INT-CTXT AEAD challenger with query c_A, τ_B. Since $\widetilde{K_{TS_{B2}}}$ and $\widetilde{K_{TS_{A2}}}$ are uniformly random and independent by Games 6 and 7, this is a sound replacement. The detection of these replacements by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the INT-CTXT or IND-CPA security of the AEAD algorithm, thus:

$$\Pr(break_7) \leq \mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\text{AEAD},\mathcal{A}}(\kappa) + \Pr(break_9)$$

Game 10: The intermediate key $K_3 \leftarrow \mathrm{PRF}(\widetilde{K_2}, l_3||\mathrm{k}_A)$ is replaced with a uniformly random and independent value, $\widetilde{K_3}$, of the same length. The replacement is performed with a PRF challenger after querying $l_3||\mathrm{k}_A$. Since $\widetilde{K_2}$ is uniformly random and independent by the previous game, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the indistinguishability security of the PRF, thus:

$$\Pr(break_9) \leq \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_{10})$$

Game 11: The MAC key $K_{MAC_B} \leftarrow \mathrm{PRF}(\widetilde{K_3}, l_{MAC_{B1}}||\mathrm{H(m)})$ is replaced with a uniformly random and independent value, $\widetilde{K_{MAC_B}}$, of the same length. The replacement is performed with a PRF challenger after querying $l_{MAC_{B1}}||\mathrm{H(m)}$. Since $\widetilde{K_3}$ is uniformly random and independent by the previous game, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT distinguishing algorithm against the indistinguishability security of the PRF, thus:

$$\Pr(break_{10}) \leq \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_{11})$$

Game 12: If the targeted session π_i^s accepts without an origin session in stage t, then this game aborts. The abortion condition is met once the final MAC tag given by no honest session π_j^r is accepted by π_i^s. The accept condition occurs when a EUF-CMA MAC challenger computes the MAC tag $\tau_B \leftarrow \mathrm{MAC.Auth}(\widetilde{K_{MAC_{B2}}}, l_{MAC_B}||\mathrm{H(m)})$ for π_i^s by querying the pertinent message m. Since $\widetilde{K_{MAC_{B2}}}$ is uniformly random and independent by Game 8, this is a sound replacement. The detection of this replacement by $\mathcal{A}_{\mathcal{HPT}}$ implies having an efficient HPT forging algorithm against the EUF-CMA security of the MAC algorithm, thus:

$$\Pr(break_{11}) \leq \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + \Pr(break_{12})$$

Where $\Pr(break_{12}) = 0$ since there are no more components to target that would allow the test session to reach the accept state or abort caused by the MAC verification failure. Additionally, as performed with the signature case, the hash collision advantage is also taken into account for all traded messages, to take into account potential attacks in regard of that. Hence:

$$\begin{aligned}\Pr(break_0) \leq\ & n_P^2 n_S n_{T_{KEM}} \big[\mathrm{Adv}^{\mathrm{IND\text{-}CCA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa)\\ &+ 4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\ &+ \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 5\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\big]\end{aligned}$$

8.1.4 Case 1: Sub-case KEM, CompromiseSK are not issued at stage t' and CompromiseSS are not issued at stages u, $t' \leq u < t$

Game 0: Defined immediately before subsection 8.1.

Game 1: Same as the sub-case Σ, CompromiseSK is not issued at stage t' and CompromiseSS is not issued at stages u, $t' \leq u < t$, section 8.1.2.

$$\Pr(break_0) \leq n_P^2 n_S^2 n_{T_{KEM}}^2 \Pr(break_1)$$

Games 2 and 3: Same as Sub-case KEM, CorruptQK is not issued at stage t before *accept*, section 8.1.3.

$$\Pr(break_1) \leq \Pr(break_2) \leq \Pr(abort)$$

Games 4 to 6: Same as Games 4 to 6, respectively, from the sub-case Σ, CompromiseSK is not issued at stage t' and CompromiseSS is not issued at stages u, $t' \leq u < t$, section 8.1.2 (targets ss_{QKD}, K_0 and K_1).

$$\Pr(abort) \leq \varepsilon_{QKD}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) + \Pr(break_6)$$

Games 7 and 8: Same as Games 6 and 7, respectively, from the sub-case KEM, CorruptQK is not issued at stage t before *accept*, section 8.1.3, but targeting $K_{TS_{A1}}$ and $K_{TS_{B1}}$ with $\widetilde{K_1}$.

$$\Pr(break_6) \leq 2\,\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_8)$$

Games 9 and 10: Same as Games 8 and 9, respectively, from the sub-case KEM, CorruptQK is not issued at stage t before *accept*, section 8.1.3, but targeting $\{B\}_{K_{TS_{B1}}}$ and $\{\mathrm{c}_B\}_{K_{TS_{A1}}}$ with $\widetilde{K_{TS_{A1}}}$ and $\widetilde{K_{TS_{B1}}}$.

$$\Pr(break_8) \leq \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \Pr(break_{10})$$

Games 11 to 18: Same as Games 5 to 12, respectively, from the sub-case KEM, CorruptQK is not issued at stage t before *accept*, section 8.1.3, but targeting K_2, $K_{TS_{A2}}$, $K_{TS_{B2}}$, $\{B\}_{K_{TS_{B1}}}$, $\{\mathrm{c}_A, \tau_B\}_{K_{TS_{B2}}}$, $\{\mathrm{c}_B, \tau_A\}_{K_{TS_{A2}}}$, K_3, K_{MAC_B} and τ_B with the respective uniformly random and independent parameters and security challengers.

$$\begin{aligned} \Pr(break_{10}) \;&\leq\; 5\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\ &+\; \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + \Pr(break_{18}) \end{aligned}$$

Game 19: Same as Game 13 from the sub-case Σ, CompromiseSK is not issued at stage t' and CompromiseSS is not issued at stages u, $t' \leq u < t$, section 8.1.2, but targeting *SecState* with $\widetilde{K_3}$.

$$\Pr(break_{18}) \leq \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_{19})$$

The next game hops involving the secret state *SecState* are repeated $(t - t')$ times in each consecutive stage u where $t' < u \leq t$. Recall that there is a matching session π_j^r with π_i^s for all the stages following stage t'.

Game 20: Same as Game 6 but targeting K_1 with uniformly random and independent parameter $\widetilde{SecState}$, and with a PRF challenger.

$$\Pr(break_{19}) \leq n_T\big\{\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(break_{20})\big\}$$

Games 21 to 33: Same as Games 7 to 19

$$\begin{aligned}\Pr(\mathit{break}_{20}) &\leq 8\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + 2\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + 2\mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\ &+ \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\end{aligned}$$

To include potential attacks where collision of the digests could be exploited, as done in all of the precedent cases, the union bonds of the advantages related to the collision probability are also included. That is:

$$\begin{aligned}&\Pr(\mathit{break}_{18}) \leq 5\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(\mathit{break}_{19})\\ &\Pr(\mathit{break}_{33}) \leq 5\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\end{aligned}$$

Since there are no more primitives or key material that the adversary can target, the final advantage can be estimated.

By combining all the inequalities together, the inequality 4 follows.

$$\begin{aligned}\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A},C_1}_{\Pi,n_P,n_S,n_{T_{KEM}}}(\kappa) &\leq n_P^2 n_S n_{T_{KEM}}\big[\mathrm{Adv}^{\text{IND-CCA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa)\\ &+ 4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\ &+ \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 5\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\big] \qquad (4)\\ &+ n_P^2 n_S^2 n_{T_{KEM}}^2\big[\varepsilon_{QKD}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa)\\ &+ (1 + n_{T_{KEM}})\big\{9\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + 2\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\ &+ 2\mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 5\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\big\}\big]\end{aligned}$$

The sub-case for Π_{MAC} is treated next.

8.1.5 Case 1: Sub-case MAC, CompromiseKP is not issued at stage t before *accept*

Games 0 and 1: Defined immediately before and immediately after subsection 8.1, respectively.

$$\Pr(\mathit{break}_0) \leq n_P^2 n_S n_{T_{MAC}} \Pr(\mathit{break}_1)$$

For this sub-case, the adversary does not have to guess the stage from which the keys K_0 are being taken, since honest parties follow a public convention. For example, parties could use the last derived key session from the K_0 derivation stage, or even extract sufficient keys in the first QKD iteration in order not to rely on asymmetric cryptography anymore in further QKD iterations.

Games 2 and 3: Same as sub-cases Σ (and KEM), CorruptQK is not issued at stage t before *accept*, section 8.1.1 (and 8.1.3), respectively.

$$\Pr(\mathit{break}_1) \leq \Pr(\mathit{break}_2) \leq \Pr(\mathit{abort})$$

Game 4: The shared secret K_0 is replaced with a uniformly random and independent value, $\widetilde{K_0}$, of the same length. The replacement is performed with a challenger, which has a shared secret key pool that stores session keys from past protocols Π run by the challenger themselves, and $\widetilde{K_0}$ is a key from the challenger's shared secret key pool. If the session π_i^s reaches the accept state, then the adversary must have compromised or guessed K_0 correctly. As the queries CompromiseKP(i, s, t) or CompromiseKP(j, r, t) have not been issued, it follows that:

$$\Pr(abort) \leq \mathrm{Adv}_{\Pi,n_P,n_S,(n'_T|K_0 \text{ derivation stage})}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}(\kappa) + \Pr(break_4)$$

Games 5 to 11: Same as Games 6 to 12, respectively, from the sub-case Σ, CompromiseSK is not issued at stage t' and CompromiseSS is not issued at stages u, $t' \leq u < t$, section 8.1.2.

$$\begin{aligned}\Pr(break_4) \quad &\leq \quad \mathrm{Adv}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}^{\mathrm{ind}}(\kappa) + 3\mathrm{Adv}_{\mathrm{PRF},\mathcal{A}}^{\mathrm{ind}}(\kappa) + \mathrm{Adv}_{\mathrm{AEAD},\mathcal{A}}^{\mathrm{IND\text{-}CPA}}(\kappa) \\ &+ \quad \mathrm{Adv}_{\mathrm{AEAD},\mathcal{A}}^{\mathrm{INT\text{-}CTXT}}(\kappa) + \mathrm{Adv}_{\mathrm{MAC},\mathcal{A}}^{\mathrm{EUF\text{-}CMA}}(\kappa) + \Pr(break_{11})\end{aligned}$$

For this long-term HPT-secure MAC authentication protocol, the hash function cannot be used to find vulnerabilities - the attacks where an adversary plays with the digest go back to trying to compromise the primitives that take a hash digest as input, which are already taken into account in the described previous Games.

By combining everything together, this sub-sub case yields:

$$\begin{aligned}\Pr(break_0) \quad &\leq \quad n_P^2 n_S n_{T_{MAC}} \big[\mathrm{Adv}_{\Pi,n_P,n_S,(n'_T|K_0 \text{ derivation stage})}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}(\kappa) \\ &+ \quad \mathrm{Adv}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}^{\mathrm{ind}}(\kappa) + 3\mathrm{Adv}_{\mathrm{PRF},\mathcal{A}}^{\mathrm{ind}}(\kappa) + \mathrm{Adv}_{\mathrm{AEAD},\mathcal{A}}^{\mathrm{IND\text{-}CPA}}(\kappa) \\ &+ \quad \mathrm{Adv}_{\mathrm{AEAD},\mathcal{A}}^{\mathrm{INT\text{-}CTXT}}(\kappa) + \mathrm{Adv}_{\mathrm{MAC},\mathcal{A}}^{\mathrm{EUF\text{-}CMA}}(\kappa)\big]\end{aligned}$$

8.1.6 Case 1: Sub-case MAC, CompromiseKP are not issued at stage t' and CompromiseSS are not issued at stages u, $t' \leq u < t$

Game 0: Defined immediately before subsection 8.1.

Game 1: Same as previous Game 1 (sub-case MAC, CompromiseKP is not issued at stage t before *accept*, section 8.1.5), where the adversary has to additionally guess the session π_j^r in which the test session π_i^s has matched in stages u, and taking into account the guessed stage t' where the adversary $\mathcal{A}$ has not issued CompromiseKP(i, s, t') or CompromiseKP(j, r, t'), and CompromiseSS(i, s, u) or CompromiseSS(j, r, u) have been issued, where the test session π_i^s matches session π_j^r in stages u, where $t' \leq u < t$:

$$\Pr(break_0) \leq n_P^2 n_S^2 n_{T_{MAC}}^2 \Pr(break_1)$$

Games 2 to 11: Same as Games 2 to 11, respectively, of the sub-case MAC, CompromiseKP is not issued at stage t before *accept*, section 8.1.5:

$$\begin{aligned}\Pr(\mathit{break}_0) &\leq n_P^2 n_S^2 n_{T_{MAC}}^2 \big[\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,(n'_T|K_0 \text{ derivation stage})}(\kappa) \\ &+ \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) + 3\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\ &+ \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + \Pr(\mathit{break}_{11})\big]\end{aligned}$$

Game 12: Same as Game 13 from the sub-case Σ, CompromiseSK is not issued at stage t' and CompromiseSS is not issued at stages u, $t' \leq u < t$, section 8.1.2, but targeting *SecState* using $\widetilde{K_1}$).

$$\Pr(\mathit{break}_{11}) \leq \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(\mathit{break}_{12})$$

The next game hops involving the secret state *SecState* are repeated $(t - t')$ times in each consecutive stage u where $t' < u \leq t$. Recall that there is a matching session π_j^r with π_i^s for all the stages following stage t'.

Game 13: Same as Game 5 from sub-case *MAC*, CompromiseKP is not issued at stage t before *accept*, section 8.1.5, but targeting K_1 using a uniformly random and independent parameter $\widetilde{SecState}$ and with a PRF challenger.

$$\Pr(\mathit{break}_{12}) \leq n_{T_{MAC}}(\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \Pr(\mathit{break}_{13}))$$

Games 14 to 20: Same as Games 6 to 12 but targeting K_{TS_A}, K_{TS_B}, K_{MAC_B}, $\{\tau_A\}_{K_{TS_A}}$, $\{\tau_B\}_{K_{TS_B}}$, τ_B and *SecState*) with the corresponding uniformly random material:

$$\begin{aligned}\Pr(\mathit{break}_{13}) &\leq 4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\ &+ \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + \Pr(\mathit{break}_{20})\end{aligned}$$

For this long-term HPT-secure MAC authentication protocol, the hash function cannot be used to find vulnerabilities - the attacks where an adversary plays with the digest go back to trying to compromise the primitives that take a hash digest as input, which are already taken into account.

Since there are no more primitives or key material that the adversary can target, the final advantage can be estimated.

By combining all the inequalities together, inequality 5 follows.

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A},C_1}_{\Pi,n_P,n_S,n_{T_{MAC}}}(\kappa) &\leq n_P^2 n_S n_{T_{MAC}}\big[\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,(n'_T|K_0 \text{ derivation stage})}(\kappa)\\
&+ \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) + 3\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\
&+ \mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\big]\\
&+ n_P^2 n_S^2 n_{T_{MAC}}^2\big[\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,(n'_T|K_0 \text{ derivation stage})}(\kappa)\\
&+ \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) + 4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\
&+ \mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\\
&+ n_{T_{MAC}}\big\{5\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\
&+ \mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\big\}\big] \qquad (5)
\end{aligned}$$

8.2 Case 2: Test *resp* session without origin session

Case 2 follows analogously to Case 1 and is the same regarding the signature-based and MAC-based protocols, whereas a small change is introduced for the KEM-based protocol.

8.2.1 Case 2: Sub-case Σ

Analogously, the advantage of Case 2 is exactly the same as the advantage of Case 1 for the signature-based protocol, giving the inequality 6.

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A},C_2}_{\Pi,n_P,n_S,n_{T_\Sigma}}(\kappa) &\leq n_P^2 n_S n_{T_\Sigma}\big[\mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\Sigma,\mathcal{A}}(\kappa) + 6\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa))\\
&+ n_P^2 n_S^2 n_{T_\Sigma}^2(\varepsilon_{QKD}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) \qquad (6)\\
&+ (1+n_{T_\Sigma})\big\{5\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\
&+ \mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 6\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\big\}\big]
\end{aligned}$$

8.2.2 Case 2: Sub-case KEM

Analogously, the advantage of Case 2 is exactly the same as the advantage of Case 1 for the KEM-based protocol, giving the inequality 7.

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A},C_2}_{\Pi,n_P,n_S,n_{T_{KEM}}}(\kappa) &\leq n_P^2 n_S n_{T_{KEM}}\big[\mathrm{Adv}^{\mathrm{IND\text{-}CCA}}_{\mathrm{KEM},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa)\\
&+ 4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\
&+ \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 5\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\big] \qquad (7)\\
&+ n_P^2 n_S^2 n_{T_{KEM}}^2\big[\varepsilon_{QKD}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa)\\
&+ (1+n_{T_{KEM}})\big\{9\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + 2\mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa)\\
&+ 2\mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) + 5\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\big\}\big]
\end{aligned}$$

8.2.3 Case 2: Sub-case MAC

Analogously, the advantage of Case 2 is exactly the same as the advantage of Case 1 for the MAC-based protocol, giving the inequality 8.

$$
\begin{aligned}
\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A},C_2}_{\Pi,n_P,n_S,n_{T_{MAC}}}(\kappa) \;\leq\;& n_P^2 n_S n_{T_{MAC}} \big[\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,(n'_T|K_0 \text{ derivation stage})}(\kappa) \\
+\;& \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) + 3\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
+\;& \mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\big] \\
+\;& n_P^2 n_S^2 n_{T_{MAC}}^2 \big[\mathrm{Adv}^{\mathrm{HAKE,clean}_{HPT},\mathcal{A}}_{\Pi,n_P,n_S,(n'_T|K_0 \text{ derivation stage})}(\kappa) \\
+\;& \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) + 4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
+\;& \mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa) \\
+\;& n_{T_{MAC}}\big\{5\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
+\;& \mathrm{Adv}^{\mathrm{INT\text{-}CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\big\}\big] \qquad (8)
\end{aligned}
$$

8.3 Case 3: Test session with matching session

All sub-cases of Case 3 have almost the same upper bound, given by the nature of QKD and the lack of derivation of a proper session key. If the test session π_i^s has a matching session π_j^r, then the public information that both sessions intended to send to each other arrived at the intended destination without modification by the adversary.

The adversary can try to perform unauthenticated QKD with the test session π_i^s and the matching session π_j^r, such that the public traffic to authenticate in both cases is the same, or the digests given by different traffics collide. In this way, the adversary would be able to share session keys with honest parties without having to deal with the authentication step. Due to the no-cloning theorem and the rejection given a high enough QBER, the adversary has a negligible probability to realize such a person-in-the-middle attack without being detected, which is taken into account within ε_{QKD}: the negligible chance of success for related attacks that target QKD itself rather than the authentication process are all included within the security of QKD, inside the parameter $\varepsilon_{QKD}(\kappa)$.

Game 1: The indices (i, s, t) and (j, r, t), corresponding to the targeted matching sessions π_i^s and π_j^r, are guessed. If the query Test(i', s', t') is issued to a session $\pi_{i'}^{s'}$, where $\pi_{j'}^{r'}$ matches $\pi_{i'}^{s'}$ in stage t', and if $(i, s, j, r, t) \neq (i', s', j', r', t')$, then the game aborts, yielding for the worst-case scenario:

$$\Pr(break_0) \leq n_P^2 n_S^2 n_T \Pr(break_1)$$

The following games target the confidentiality security of the exchanged AEAD ciphers. Since the adversary has no flexibility to forge and only targets learning what is inside the AEAD ciphers, only the IND-CPA security for the AEAD algorithm is taken into account.

Games 2 to 5 for Π_Σ and Π_{MAC}, Games 2 to 5 and $2_{\mathbf{bis}}$ to $5_{\mathbf{bis}}$ for Π_{KEM}: These indistinguishability games target the traffic keys (K_{TS_A}, K_{TS_B}) or ($K_{TS_{A1}}$, $K_{TS_{B1}}$, $K_{TS_{A2}}$, $K_{TS_{B2}}$) and the AEAD ciphers ($\{B, \sigma_B, \tau_B\}_{K_{TS_B}}$, $\{A, \sigma_A, \tau_A\}_{K_{TS_A}}$), ($\{B\}_{K_{TS_{B1}}}$, $\{A\}_{K_{TS_{A1}}}$, $\{\mathrm{c}_A, \tau_B\}_{K_{TS_{B2}}}$, $\{\mathrm{c}_B, \tau_A\}_{K_{TS_{A2}}}$) or ($\{\tau_A\}_{K_{TS_A}}$, $\{\tau_B\}_{K_{TS_B}}$) for Π_Σ, Π_{KEM} and Π_{MAC}, respectively, in same way as undertaken in previous Games (for example Games 7, 8, 10 and 11 from the sub-case Σ, CompromiseSK is not issued at stage t' and CompromiseSS is not issued at stages u, $t' \leq u < t$, section 8.1.2).

For Π_Σ and Π_{MAC}:

$$\Pr(break_1) \leq 2\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) + \Pr(break_5)$$

For Π_{KEM}:

$$\Pr(break_1) \leq 4\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) + \Pr(break_5)$$

Game 6: The session key ss_{rest} is replaced with a uniformly random and independent value, $\widetilde{ss_{rest}}$, of the same length. The replacement is performed with an unauthenticated QKD challenger that replaces the public traffic m_{QKD} with m'_{QKD}, which is used to distill $\widetilde{ss_{rest}}$. Since π_i^s matches session π_j^r in stage t', the replaced traffic m'_{QKD} is received by both sessions without modification. The detection of this replacement by $\mathcal{A}$ implies having an efficient distinguishing algorithm against the indistinguishability security of the unauthenticated QKD, where the algorithm does not target authentication. This relates to the RevealITS query, in which the conditioned unbounded adversary does not compromise the QKD keys with a small probability $\varepsilon_{QKD}(\kappa)$, for a security parameter κ:

$$\Pr(break_5) \leq \varepsilon_{QKD}(\kappa) + \Pr(break_6)$$

Since the session keys are uniformly random and independent, $\Pr(break_6)$ is equal to zero. Hence, the security analysis of Case 3 yields the inequality 9.

$$\begin{aligned}
\mathrm{Adv}^{\text{HAKE},\text{clean}_{HPT},\mathcal{A},C_3}_{\Pi,n_P,n_S,n_T}(\kappa) \;\leq\;& n_P^2 n_S^2 n_{T_\Sigma}\left[2\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) + \varepsilon_{QKD}(\kappa)\right] \\
+\;& n_P^2 n_S^2 n_{T_{KEM}}\left[4\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) + \varepsilon_{QKD}(\kappa)\right] \\
+\;& n_P^2 n_S^2 n_{T_{MAC}}\left[2\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) + \varepsilon_{QKD}(\kappa)\right] \\
&= n_P^2 n_S^2 n_T\left[2\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa) + \varepsilon_{QKD}(\kappa)\right] \\
&+ n_P^2 n_S^2 n_{T_{KEM}} 2\mathrm{Adv}^{\text{IND-CPA}}_{\text{AEAD},\mathcal{A}}(\kappa)
\end{aligned} \tag{9}$$

9 Discussion

The presented protocols in this work are comparable to those presented within the Muckle works. Specifically, the presented signature-based, KEM-based and MAC-based protocols are comparable with the Muckle+ [17], Muckle# [18] and Muckle [15] works, respectively.

9.1 Comparison with the Muckle protocols

The difference of the advantages given by the HAKE key-indistinguishability games for the protocols presented here and those in the Muckle works can be explained through the different cleanness predicates used for this work and the Muckle approaches. Furthermore, this work takes into account the confidentiality aspect, which can be ignored in order to compare this work with the Muckle series.

There are two main differences between the cleanness predicates presented in this work compared to those defined in the Muckle works.

First, the cleanness predicates defined in this work regarding the existence, or not, of an origin session π_j^r of π_i^s (in stage t) have more queries that are not allowed to be issued before sessions get to an *accept* state. This set of prohibited queries covers every possibility regarding authentication. Specifically, the adversary is allowed to obtain all but one of the key materials used to authenticate and the authentication still holds. This difference in the cleanness predicate is the cause of the additional MAC EUF-CMA advantages that this work presents for Cases 1 and 2 - the final key confirmation step is used as a probe to detect fraudulent attempts at authentication. The ephemeral key advantages that the protocols from the Muckle series present are replaced in this work with the advantage given by the ephemeral QKD keys, namely ε_{QKD}. In respect to the MAC-based protocol, the advantage of the shared material is taken into account, which is not the case for the protocol presented within Muckle, where the shared key is assumed to be perfectly secure. And finally, the different number of PRF and dual PRF advantages is natural, since the number of uses of those functions depends on how the protocols are constructed.

The second main difference is the cleanness predicate regarding the matching session scenario, because the session keys do not depend on the intermediate keys derived in the protocol, nor the authentication material. The QKD keys are shown to be ITS and ε_{QKD} secure, regardless of the authentication method. This makes sense, since if the adversary is only interacting passively with the QKD traffic (no impersonation nor substitution attacks), QKD is secure within its security parameter.

The idea of using PQC to authenticate QKD is to only rely on PQC for the very first QKD iteration. Afterwards, authentication can rely on non-ITS MAC, or ITS MAC. The following part provides an example of how to manipulate the advantage inequality formula given in theorem 23, in the case scenario where KEM-based PQC authentication is performed for stage 1 (the first iteration of QKD) and the non-ITS MAC is used for the subsequent stages (2nd and beyond iterations of QKD). Since the security of the QKD keys decreases as the number of iterations increases, the first distilled QKD keys in iteration 1 may be preferred to perform the MAC-based protocol. Nevertheless, QKD keys distilled in other iterations can be used, as long as the security parameter given by the MAC-based protocol does not surpass an arbitrary threshold.

9.2 Example: KEM-based in stage 1, MAC-based for rest

Let two parties perform one session to derive session keys during n stages, $n_P = 2$ and $n_S = 1$. They perform the KEM-based protocol in stage 1, afterwards performing the MAC-based protocol in stages 2 to n. Since the signature-based protocol is not applied, $n_{T_\Sigma} = 0$ at all times.

The security level at stage $n_T = 1$ is:

$$\begin{aligned}
\mathrm{Adv}_{\Pi,n_P=2,n_S=1,n_T=1}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}(\kappa) &= \mathrm{Adv}_{\Pi,n_P=2,n_S=1,n_{T_{KEM}}=1}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}(\kappa) \\
&\leq 8\big[\mathrm{Adv}_{\mathrm{KEM},\mathcal{A}}^{\mathrm{IND\text{-}CCA}}(\kappa) + \varepsilon_{QKD}(\kappa) + 2\mathrm{Adv}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}^{\mathrm{ind}}(\kappa) \\
&\quad +22\mathrm{Adv}_{\mathrm{PRF},\mathcal{A}}^{\mathrm{ind}}(\kappa) + 5\mathrm{Adv}_{\mathrm{AEAD},\mathcal{A}}^{\mathrm{IND\text{-}CPA}}(\kappa) \\
&\quad +5\mathrm{Adv}_{\mathrm{AEAD},\mathcal{A}}^{\mathrm{INT\text{-}CTXT}}(\kappa) + 3\mathrm{Adv}_{\mathrm{MAC},\mathcal{A}}^{\mathrm{EUF\text{-}CMA}}(\kappa) + 15\mathrm{Adv}_{\mathrm{H},\mathcal{A}}(\kappa)\big] \\
&\quad +4\big[4\mathrm{Adv}_{\mathrm{AEAD},\mathcal{A}}^{\mathrm{IND\text{-}CPA}}(\kappa) + \varepsilon_{QKD}(\kappa)\big]
\end{aligned}$$

For the following stages, it is assumed that parties have stored sufficient keys in their shared secret key pool from stage 1 to run the rest of the stages. The MAC-based advantage, for stage 2 and the subsequent stages, takes as input the advantage where the keys were derived, i.e. in stage $n_T = 1$ in this case.

The security level at stage t, for $2 \leq t$, is:

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_T=t}(\kappa) &= \mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_{T_{KEM}}=1}(\kappa) + \mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_{T_{MAC}}=t-1}(\kappa) \\
&\leq \mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_T=1}(\kappa) \\
&\quad +4(t-1)\big[\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_T=1}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) \\
&\quad +3\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
&\quad +\mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\big] \\
&\quad +4(t-1)^2\big[\mathrm{Adv}^{\mathrm{HAKE},\mathrm{clean}_{HPT},\mathcal{A}}_{\Pi,n_P=2,n_S=1,n_T=1}(\kappa) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF}^{\mathrm{dual}},\mathcal{A}}(\kappa) \\
&\quad +4\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
&\quad +\mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\big] \\
&\quad +4(t-1)^3\big[5\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{PRF},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) \\
&\quad +\mathrm{Adv}^{\text{INT-CTXT}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathrm{MAC},\mathcal{A}}(\kappa)\big] \\
&\quad +4(t\left[2\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa) + \varepsilon_{QKD}(\kappa)\right] + 2\mathrm{Adv}^{\text{IND-CPA}}_{\mathrm{AEAD},\mathcal{A}}(\kappa))
\end{aligned}$$

Once the corresponding advantage terms are known, the security level at stage t, which takes into account confidentiality, integrity and secrecy, is upper bounded by a known term.

10 Conclusion

The motivating scenario for this work is authentication of QKD, when parties have no initial shared secret material to leverage. A solution for this scenario enables new QKD users to securely join a new (to them) already-secure network remotely. This is clearly a very relevant and important scenario in the modern mobile world.

To address this scenario, two protocols of PQC-based authentication (signature and KEM-based) for QKD and a non-ITS long-term secure MAC-based authentication have been analyzed and proven to be secure, according to defined security parameters and bounded adversaries. The T_{HPT}-term security has been defined, along with the HPT and conditioned unbounded adversaries. Given the security framework, honest parties can distill the ITS QKD keys with the underlying security provided by the security analysis.

After comparison between the protocols presented in this work and those given in the Muckle series [15, 17, 18], a practical example of the advantages to take into account, when the first QKD iteration is authenticated using PQC KEM algorithms and authentication for posterior QKD iterations relays only in non-ITS instead of PQC, is presented.

Neither the form of QKD, nor the PQC algorithms, are specified in this work, so it is widely applicable across the QKD and PQC spectra. Any PQC algorithms that are secure following the definitions from section 3 and 4 are valid PQC algorithms for use. However, using standardized PQC algorithms, such as those from NIST, is identified as best practice [8]. The security proofs presented here also take into account the confidentiality aspect. Additionally, honest parties are free to switch between the three presented protocols. However, it is always possible to instead adopt an ITS authentication protocol starting from stage 2, provided that the parties have derived sufficient key material in the first stage 1 iteration.

PQC can offer remote and secure authentication to QKD for the very first iteration between new correspondents and can also be used to provide redundant security once a QKD link is established, as shown in the Muckle series. An open question is to determine if there are other methods to combine PQC with QKD to complement each other, or combine securities.

Acknowledgements JAVG thanks Christopher Battarbee, Ludovic Perret, Delaram Kahrobaei and Panagiotis Papanastasiou for valuable discussions. JAVG has conducted this work with the support of EPSRC PhD studentship Grant EP/W524657/1. JAVG and TS have conducted this work partially with the support of ONR Grant 62909-24-1-2002.

References

[1] Kelsey Horan and Delaram Kahrobaei. "The Hidden Subgroup Problem and Post-quantum Group-Based Cryptography". In: *Mathematical Software – ICMS 2018*. Ed. by James H. Davenport et al. Cham: Springer International Publishing, 2018, pp. 218–226. ISBN: 978-3-319-96418-8.

[2] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[3] Nicolas Gisin et al. "Quantum cryptography". In: *Rev. Mod. Phys.* 74 (1 Mar. 2002), pp. 145–195. DOI: 10.1103/RevModPhys.74.145. URL: https://link.aps.org/doi/10.1103/RevModPhys.74.145.

[4] S. Pirandola et al. "Advances in quantum cryptography". In: *Advances in Optics and Photonics* 12.4 (Dec. 2020), p. 1012. ISSN: 1943-8206. DOI: 10.1364/aop.361502. URL: http://dx.doi.org/10.1364/AOP.361502.

[5] Feihu Xu et al. "Secure quantum key distribution with realistic devices". In: *Reviews of Modern Physics* 92.2 (May 2020). DOI: 10.1103/revmodphys.92.025002. URL: https://doi.org/10.1103%2Frevmodphys.92.025002.

[6] Mark N. Wegman and J.Lawrence Carter. "New hash functions and their use in authentication and set equality". In: *Journal of Computer and System Sciences* 22.3 (1981), pp. 265–279. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(81)90033-7. URL: https://www.sciencedirect.com/science/article/pii/0022000081900337.

[7] Evgeniy O. Kiktenko et al. "Lightweight Authentication for Quantum Key Distribution". In: *IEEE Transactions on Information Theory* 66.10 (2020), pp. 6354–6368. DOI: 10.1109/TIT.2020.2989459.

[8] National Institute of Standards and Technology (NIST). *Post-Quantum Cryptography*. 2024. URL: https://csrc.nist.gov/projects/post-quantum-cryptography (visited on 07/03/2025).

[9] Farshad Rahimi Ghashghaei et al. "Enhancing the Security of Classical Communication with Post-Quantum Authenticated-Encryption Schemes for the Quantum Key Distribution". In: *Computers* 13.7 (2024). ISSN: 2073-431X. DOI: 10.3390/computers13070163. URL: https://www.mdpi.com/2073-431X/13/7/163.

[10] Hassan Termos. "Quantum Authentication Evolution: Novel Approaches for Securing Quantum Key Distribution". In: *Entropy* 26.6 (2024). ISSN: 1099-4300. DOI: 10.3390/e26060447. URL: https://www.mdpi.com/1099-4300/26/6/447.

[11] Liu-Jun Wang et al. "Experimental authentication of quantum key distribution with post-quantum cryptography". In: *npj Quantum Information* 7.1 (May 2021), p. 67. DOI: `10.1038/s41534-021-00400-7`. URL: `https://doi.org/10.1038/s41534-021-00400-7`.

[12] Michele Mosca, Douglas Stebila, and Berkant Ustaoğlu. "Quantum Key Distribution in the Classical Authenticated Key Exchange Framework". In: *Post-Quantum Cryptography.* Ed. by Philippe Gaborit. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 136–154. ISBN: 978-3-642-38616-9. URL: `https://doi.org/10.1007/978-3-642-38616-9_9`.

[13] Hugo Krawczyk. "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols". In: *Advances in Cryptology - CRYPTO 2003.* Ed. by Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 400–425. ISBN: 978-3-540-45146-4.

[14] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3.* RFC 8446. 2018. URL: `https://www.rfc-editor.org/rfc/rfc8446` (visited on 07/03/2025).

[15] Benjamin Dowling, Torben Brandt Hansen, and Kenneth G. Paterson. "Many a Mickle Makes a Muckle: A Framework for Provably Quantum-Secure Hybrid Key Exchange". In: *Post-Quantum Cryptography.* Ed. by Jintai Ding and Jean-Pierre Tillich. Cham: Springer International Publishing, 2020, pp. 483–502. ISBN: 978-3-030-44223-1.

[16] Lydia Garms et al. "Experimental Integration of Quantum Key Distribution and Post-Quantum Cryptography in a Hybrid Quantum-Safe Cryptosystem". In: *Advanced Quantum Technologies* 7.4 (2024), p. 2300304. DOI: `https://doi.org/10.1002/qute.202300304`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/qute.202300304`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.202300304`.

[17] Sonja Bruckner, Sebastian Ramacher, and Christoph Striecks. "Muckle+: End-to-End Hybrid Authenticated Key Exchanges". In: *Post-Quantum Cryptography.* Ed. by Thomas Johansson and Daniel Smith-Tone. Cham: Springer Nature Switzerland, 2023, pp. 601–633. ISBN: 978-3-031-40003-2.

[18] Christopher Battarbee et al. *Quantum-Safe Hybrid Key Exchanges with KEM-Based Authentication.* Preprint at https://arxiv.org/abs/2411.04030. 2024. arXiv: `2411.04030 [cs.CR]`. URL: `https://arxiv.org/abs/2411.04030`.

[19] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Post-Quantum TLS Without Handshake Signatures". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security.* CCS '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1461–1480. ISBN: 9781450370899. DOI: `10.1145/3372297.3423350`. URL: `https://doi.org/10.1145/3372297.3423350`.

[20] William K. Wootters, William K. Wootters, and Wojciech H. Zurek. "A single quantum cannot be cloned". In: *Nature* 299 (1982), pp. 802–803. URL: `https://api.semanticscholar.org/CorpusID:4339227`.

[21] Masanao Ozawa. "Heisenberg's original derivation of the uncertainty principle and its universally valid reformulations". In: *Current Science* 109.11 (2015), pp. 2006–2016. ISSN: 00113891. URL: `http://www.jstor.org/stable/24906690` (visited on 02/06/2025).

[22] Artur K. Ekert. "Quantum cryptography based on Bell's theorem". In: *Phys. Rev. Lett.* 67 (6 Aug. 1991), pp. 661–663. DOI: `10.1103/PhysRevLett.67.661`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.67.661`.

[23] Christopher Portmann and Renato Renner. "Cryptographic security of quantum key distribution". In: *ArXiv* abs/1409.3525 (2014). URL: `https://api.semanticscholar.org/CorpusID:14053576`.

[24] Stefano Pirandola and Panagiotis Papanastasiou. “Improved composable key rates for CV-QKD”. In: *Phys. Rev. Res.* 6 (2 June 2024), p. 023321. DOI: 10.1103/PhysRevResearch.6.023321. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.6.023321.

[25] Christopher Portmann. “Key Recycling in Authentication”. In: *IEEE Transactions on Information Theory* 60.7 (2014), pp. 4383–4396. DOI: 10.1109/TIT.2014.2317312.

[26] Aysajan Abidin and Jan-Åke Larsson. “Direct proof of security of Wegman–Carter authentication with partially known key”. In: *Quantum Information Processing* 13.10 (Oct. 2014), pp. 2155–2170. ISSN: 1573-1332. DOI: 10.1007/s11128-013-0641-6. URL: https://doi.org/10.1007/s11128-013-0641-6.

[27] Michael Ben-Or et al. “The Universal Composable Security of Quantum Key Distribution”. In: *Theory of Cryptography*. Ed. by Joe Kilian. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 386–406. ISBN: 978-3-540-30576-7.

[28] Douglas Stinson. “Universal Hash Families and the Leftover Hash Lemma, and Applications to Cryptography and Computing”. In: *Journal of Combinatorial Mathematics and Combinatorial Computing* 42 (Mar. 2001).

[29] Mihir Bellare and Chanathip Namprempre. “Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm”. In: *Advances in Cryptology — ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 531–545. ISBN: 978-3-540-44448-0.

[30] Lov K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC ’96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866. URL: https://doi.org/10.1145/237814.237866.

[31] Dan Boneh and Mark Zhandry. “Quantum-Secure Message Authentication Codes”. In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 592–608. ISBN: 978-3-642-38348-9.

[32] D. R. Stinson. “Universal hashing and authentication codes”. In: *Designs, Codes and Cryptography* 4.3 (July 1994), pp. 369–380. ISSN: 1573-7586. DOI: 10.1007/BF01388651. URL: https://doi.org/10.1007/BF01388651.

[33] Akinori Hosoyamada et al. “A Modular Approach to the Incompressibility of Block-Cipher-Based AEADs”. In: *Advances in Cryptology – ASIACRYPT 2022*. Ed. by Shweta Agrawal and Dongdai Lin. Cham: Springer Nature Switzerland, 2022, pp. 585–619. ISBN: 978-3-031-22966-4.

[34] Lily Chen. *Recommendation for Key Derivation Using Pseudorandom Functions*. Tech. rep. SP 800-108r1-upd1. Final version, August 2022. National Institute of Standards and Technology, 2022. URL: https://csrc.nist.gov/pubs/sp/800/108/r1/upd1/final.

[35] Tony Hansen and Donald E. Eastlake 3rd. *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)*. RFC 6234. May 2011. DOI: 10.17487/RFC6234. URL: https://www.rfc-editor.org/info/rfc6234.

[36] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication*. Tech. rep. SP 800-38B. National Institute of Standards and Technology, 2016. URL: https://csrc.nist.gov/pubs/sp/800/38/b/upd1/final.

[37] Tetsu Iwata et al. *The AES-CMAC Algorithm*. RFC 4493. June 2006. DOI: 10.17487/RFC4493. URL: https://www.rfc-editor.org/info/rfc4493.

[38] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. Tech. rep. SP 800-38D. National Institute of Standards and Technology, 2007. URL: `https://csrc.nist.gov/pubs/sp/800/38/d/final`.

[39] John Viega and David McGrew. *The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH*. RFC 4543. May 2006. DOI: `10.17487/RFC4543`. URL: `https://www.rfc-editor.org/info/rfc4543`.

[40] Jin Li et al. "Aggregate Proxy Signature and Verifiably Encrypted Proxy Signature". In: *Provable Security*. Ed. by Willy Susilo, Joseph K. Liu, and Yi Mu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 208–217. ISBN: 978-3-540-75670-5.

[41] Cas Cremers and Michèle Feltz. "Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal". In: *Computer Security – ESORICS 2012*. Ed. by Sara Foresti, Moti Yung, and Fabio Martinelli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 734–751. ISBN: 978-3-642-33167-1.

[42] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. "On Post-compromise Security". In: *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. 2016, pp. 164–178. DOI: `10.1109/CSF.2016.19`.

[43] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Cryptology ePrint Archive, Paper 2004/332. 2004. URL: `https://eprint.iacr.org/2004/332`.

[44] Mihir Bellare and Phillip Rogaway. "Code-Based Game-Playing Proofs and the Security of Triple Encryption". In: *IACR Cryptol. ePrint Arch.* 2004 (2004), p. 331. URL: `https://api.semanticscholar.org/CorpusID:12916996`.