



Muckle+: End-to-End Hybrid Authenticated Key Exchanges^{*}

Sonja Bruckner^{1**}, Sebastian Ramacher² , and Christoph Striecks² 

¹ University of Applied Sciences Upper Austria, Hagenberg, Austria

`sonja.bruckner@scch.at`

² AIT Austrian Institute of Technology, Vienna, Austria

`{firstname.lastname}@ait.ac.at`

Abstract. End-to-end authenticity in public networks plays a significant role. Namely, without authenticity, the adversary might be able to retrieve even confidential information straight away by impersonating others. Proposed solutions to establish an authenticated channel cover pre-shared key-based, password-based, and certificate-based techniques. To add confidentiality to an authenticated channel, authenticated key exchange (AKE) protocols usually have one of the three solutions built in. As an amplification, hybrid AKE (HAKE) approaches are getting more popular nowadays and were presented in several flavors to incorporate classical, post-quantum, or quantum-key-distribution components. The main benefit is redundancy, i.e., if some of the components fail, the primitive still yields a confidential and authenticated channel. However, current HAKE instantiations either rely on pre-shared keys (which yields inefficient end-to-end authenticity) or only support one or two of the three above components (resulting in reduced redundancy and flexibility).

In this work, we present an extension of a modular HAKE framework due to Dowling, Brandt Hansen, and Paterson (PQCrypto'20) that does not suffer from the above constraints. While their instantiation, dubbed Muckle, requires pre-shared keys (and hence yields inefficient end-to-end authenticity), our extended instantiation called Muckle+ utilizes post-quantum digital signatures. While replacing pre-shared keys with digital signatures is rather straightforward in general, this turned out to be surprisingly non-trivial when applied to HAKE frameworks (resulting in adapted proof techniques).

Keywords: end-to-end security, hybrid authenticated key exchange, post-quantum cryptography

1 Introduction

Authenticated key exchange (AKE) is an essential cryptographic building block [DH76,Mau93,BR95]. From user-to-user to server-to-server communication, data exchanged between any two parties is expected to be confidential even in the event of potentially active attacks on the communication channel. Ensuring confidentiality between two parties first requires that one can distinguish friend from foe. Specifically, if an adversary can impersonate a party in the system, all confidentiality guarantees are void since in that case the communication with the adversary is secured against outsiders, but the adversary itself may gain access to all data. Therefore, authenticity is a necessary requirement for achieving confidentiality on any level in any system and in the specific context of communication we thus require *end-to-end* authenticity. That is, both parties can directly verify the authenticity of the other party regardless of how many untrusted network links are located between them.

For network protocols on public or untrusted networks, well-established protocols such as Transport Layer Security (TLS) [Res18], IPsec [Kau05], QUIC [IT21], WireGuard [Don17] employ various forms of an end-to-end AKE [BR95]; on the one hand to authenticate the other peer and on the other hand to establish an ephemeral session key to secure the communication channel. Depending on the concrete application, AKE protocols offer certificate-based authentication, password-based authentication, pre-shared

^{*} This is the full version of [BRS23] which appears in the Post-Quantum Cryptography, 14th International Workshop, PQCrypto 2023, August 16-18, 2023, College Park, MD, USA, proceedings.

^{**} The work was conducted while the author was with AIT Austrian Institute of Technology.

key-based authentication whereas the secret keys are exchanged often using an ephemeral Diffie-Hellman key exchange or – on a more abstract level – with a key exchange using ephemeral key encapsulation mechanism (KEM) keys. Authentication in those protocols may be unilateral, e.g., only the initiator verifies the authenticity of the responder which is the default deployment mode of TLS on the web as the authentication of users is managed on the application layer, or mutual.

End-to-End Authentication Techniques. We will now discuss different techniques to achieve authenticity for key-exchange protocols: in a key exchange with pre-shared keys (PSK), both peers are required to agree on a secret key off-channel. This key is then part of the key-exchange protocol (e.g., is used as input in the key derivation function to derive the session keys) and only if the key is known, the protocol can be completed successfully. As a folklore consequence, networks with n peers necessitate the initial setup of $\mathcal{O}(n^2)$ PSKs to uniquely identify each peer. Otherwise, i.e., where 3 or more peers share the same PSK, peers would be unable to distinguish one communication partner from the other. Moreover, dynamically changing the network components becomes inefficient, e.g., if a new peer is added to the network, fresh PSKs have to be distributed to all other peers off-channel.

Password-based authenticated key exchanges [BPR00,BMP00] are of interest in a multi-client, single-server scenario where each client is uniquely identified using a (low-entropy) password. Similar to the PSK approach, the password is an intrinsic part of the exchange which cannot be completed without knowledge of the specific password. As the scenario we are considering is not a multi-client single-server scenario and, more importantly, the password-based authentication is related to a PSK authentication scenario, we will omit further discussions of this type of key exchange.

Finally, with certificate-based protocols, peers have long-term public keys (typically of a digital signature scheme) whereas certificate authorities ensure the authenticity of these keys and establish a chain of trust. During a protocol run, peers are then required to sign certain messages to authenticate the exchange. A prominent example of such a protocol is SIGMA [Kra03] which serves as a prototype for the key exchange deployed in IPsec [Kau05], for example. Recently, due to the bandwidth requirements of post-quantum signature schemes, variants with long-term KEM keys such as KEMTLS [SSW20] are also gaining interest as such variants are able to provide implicit server-to-client authentication. In this protocol, instead of signing the handshake transcript, after establishing an ephemeral secret, the client encapsulates another secret using the long-term KEM key embedded in the server’s certificates. The server can only provide a valid key confirmation message if it is able to decapsulate the ciphertext with respect to the long-term key and thereby implicitly proves knowledge of the corresponding long-term secret key. This change in the protocol incurs the cost of an additional message but KEMTLS benefits from reduced runtime and bandwidth requirements.

While PSK key exchanges can be implemented solely from symmetric-key primitives, managing the required keys is a complex task. As no key material is available during system setup, those keys need to be securely exchanged via trusted couriers, installed on devices in the fab, or other methods are required to allow the keys be installed without relying on a yet unsecured communication channel. This task becomes more complex as the network grows and infeasible if parties have no trivial way to securely exchange the PSK.

Authenticated Key Exchanges Resilient Against Quantum Attacks. End-to-end post-quantum AKE protocols have already been studied, e.g., most prominently in the area of Transport Layer Security (TLS) [BCNS15,Lan16,KSL⁺19,PST20,SSW20]. Moreover, standardization efforts towards post-quantum (hybrid) key exchanges are already in progress while NIST is expected to publish the first standards on post-quantum key-exchange mechanisms and digital signatures soon.³ For most practical use-cases that require security against cryptographically relevant quantum computers, the post-quantum cryptography (PQC) paradigm seems to be a strong fit, although some techniques are rather recent and severe attacks are happening [Beu22,CD23].

³ <https://www.ietf.org/archive/id/draft-ietf-tls-hybrid-design-05.html>, <https://csrc.nist.gov/projects/post-quantum-cryptography>

For highly secure use-cases, quantum-key distribution (QKD) [ABB⁺14,MNR⁺20] is gaining quite some attention recently with an expected market growth of 12 billion USD in 10 years.⁴ Moreover, the European initiative for a quantum communication infrastructure named EuroQCI was recently established.⁵ The benefit of a QKD system is that it guarantees information-theoretic security (ITS) compared to conjectured computational security of post-quantum primitives. However, QKD comes with significant limitations such as range and costly hardware.

To achieve ITS, QKD must use ITS authentication mechanisms [PAL⁺16] which can be enforced by relying on PSK-based authentication methods. Noteworthy, the PSKs for the individual QKD links are not enough to establish authenticity for the full path through the network as they only ensure authenticity for one link. Moreover, given the limited range of QKD link transmissions, all nodes in between are turned into so-called *trusted nodes* [MNR⁺20]. With trusted nodes, however, deployment in large-scale networks may become even more complex.⁶ Hence, practical end-to-end authenticity guarantees for the to-be-anticipated QKD networks are still under investigation.

Since both, the PQC and QKD paradigms, have benefits and downsides, and following the approach “Don’t put all your eggs in one basket,” we are interested in how to achieve end-to-end authentication and confidentiality for key exchanges with the best possible security guarantees against future threats. One promising approach is using hybrid⁷ techniques.

Hybrid Authenticated Key Exchanges (with Forward and Post-Compromise Security). Hybrid AKE (HAKE) approaches are getting more popular nowadays and were presented in several flavors to incorporate classical (or, non-quantum-safe), PQC, or QKD components [MSU13,BFG19,BBF⁺19,DHP20]. The main benefit is redundancy, i.e., if some of the components fail, the primitive still yields a confidential and authenticated channel. Moreover, HAKE provides an approach towards the transition of non-quantum-secure networks to quantum-secure ones.

Particularly interesting is the recently proposed HAKE framework with its instantiation dubbed Muckle due to Dowling, Brandt Hansen, and Paterson [DHP20]. Muckle combines secret keys obtained from a QKD network with session keys obtained from a classical and post-quantum secure key encapsulation mechanisms (KEMs). The combination of the keys is performed using a sequence of pseudo-random function evaluations.

Importantly, Muckle inherits desirable advanced security guarantees which are de-facto standard features nowadays for key exchanges, namely, *forward* and *post-compromise security*. Forward security is an important security feature in several domains. Besides being of interest in interactive key-exchange protocols [Gün90,DvOW92,DDG⁺20,RSS23], such a security feature was studied for public-key encryption [CHK03,Gro21], digital signatures [BM99,DGNW20], search on encrypted data [BMO17], 0-RTT key exchange [GHJL17,DJSS18,CRSS20,DGJ⁺21], updatable cryptography [SS21], mobile Cloud backups [DCM20], proxy cryptography [DKL⁺18], Tor [LGM⁺20], and content-delivery networks [DRSS21], among others.

Forward security allows to evolve secret key material over epochs which particularly mitigates “store-now-decrypt-later” attacks (such that access to prior ciphertexts or signing capabilities can be restricted for older epochs). Nowadays, over 99% of Internet sites⁸ support (some form of) forward security.

⁴ <https://www.reuters.com/article/us-toshiba-cyber-idUSKBN2730KW>

⁵ <https://digital-strategy.ec.europa.eu/en/policies/european-quantum-communication-infrastructure-euroqci>

⁶ Interestingly, while some approaches even backed by patents (<https://www.ipo.gov.uk/p-ipsu/Case/PublicationNumber/GB2590064>) claim to provide long-range QKD networks without trusted nodes (i.e., establishing a secure channel between any two nodes), a recent work [HAD⁺22] demonstrates that such claim cannot be met.

⁷ We are sticking to the term “hybrid” here as it was coined in prior work on AKEs [DHP20] in the meaning of combining classical (or, non-quantum-safe), QKD, and post-quantum cryptographic primitives. Other works may use the term “quantum-safe” to combine QKD and PQC primitives, or different terms.

⁸ Due to Qualys SSL Labs, <https://www.ssllabs.com/ssl-pulse/>, accessed in August 2023.

In the concrete hybrid key-exchange setting, forward security guarantees that prior session keys cannot be retrieved even if the current session and long-term keys leak. Moreover, even if *all* classic KEM key material is leaked (e.g., in the event of a cryptographically relevant quantum computer), old session keys stay safe due to the PQC and QKD guarantees. Moreover, if additionally *all* post-quantum KEM keys should leak, an adversary cannot retrieve old sessions keys due to the QKD guarantees. Conversely, if *all* QKD keys leak, the security features of the post-quantum KEM component prevent an adversary from retrieving old session keys.

Moreover, post-compromise security guarantees that future sessions are safe again (even against adversaries that gained access to an old key but do not compromise the system anymore). Thereby, we strictly require that at least one of the classic, PQC, or QKD components stays secure against a then-passive attacker.

The Muckle authentication, however, solely relies on the presence of pre-shared keys. Consequently, Muckle inherits the key management problem of PSKs in large-scale networks discussed above. In this work, we present an extension of the HAKE framework in [DHP20] via an amplification of their Muckle scheme with end-to-end authenticity and better efficiency (given that we can rely on multi-path QKD) while no sacrifices on the security guarantees have to be made.

1.1 Contribution

Our contribution can be summarized as follows:

- We extend Muckle with a certificate-based authentication mechanism via digital signatures and dub it Muckle+. While replacing pre-shared keys with digital signatures is rather straightforward in general, this turned out to be surprisingly non-trivial when applied to HAKE frameworks (resulting in adapted proof techniques). The benefits are that we avoid the usage of PSKs (with its inherent quadratic blow-up to achieve end-to-end authenticity) which results in more efficient end-to-end HAKE instantiation than previously known. While gaining significant efficiency and flexibility with our approach compared to Muckle, to retrieve the same security guarantees, we need that the QKD keys are distributed via multi-path techniques.
- We implement the Muckle+ protocol and validated its functionality using a small QKD network in the field. To the best of our knowledge, such a proof-of-concept experiment for HAKes is the first one with QKD hardware. Thereby, we can demonstrate the added authenticity guarantees that ensure an end-to-end secure connection between the initiator and responder.

More on Muckle+ and the Differences to Muckle. The Muckle protocol uses a hybrid approach combining classical, PQC, and QKD keys through the use of a key derivation function. Muckle requires a classical and post-quantum KEM as well as data from a QKD channel to create the final shared secret. Additionally, the protocol relies on a secure pseudorandom function and a message authentication code (MAC). The latter is used in combination with a QKD pre-shared key to ensure the authenticity of the key exchange. To avoid such pre-shared keys for authentication, we carefully extend Muckle to allow certificate-based authentication. Technically, we use digital signatures as a building block instead of PSKs for authentication.

However, replacing PSKs with digital signatures in HAKE is not straightforward. Using PSKs yield an interesting cryptographic feature, namely, it guarantees that a sender and a receiver share a common secret key for end-to-end authentication (leaving the quadratic blow-up in that case on the side for a moment). Now, if digital signatures are used, we cannot build on such guarantee anymore (as we are in the public-key setting).

The key observation in the HAKE realm is that in the latter case, we either require a post-quantum KEM or we need multi-path approaches for the QKD part to guarantee end-to-end authenticity again. As we want to allow the post-quantum KEM components to fail (as in Muckle), we need that the QKD keys are distributed using a multi-path approach (essentially, by distributing key components via mutually disjoint paths from the initiator to the responder such that no individual trusted node knows all of the

key material depending on some bound of colluding nodes). This is different to Muckle where Muckle only requires a “single path” to distribute the QKD key. However, as Muckle+ shows its full potential in larger-scale quantum-secure networks with many nodes, we assume that multiple paths between initiators and responders are available.

Through this alteration, we achieve the desired security properties, i.e., we are able to endure all security claims from original Muckle (in particular, forward and post-compromise security) while avoiding PSKs, which we show by formally proving our variant Muckle+ secure in the HAKE framework. Moreover, our instantiation allows for an efficient approach to achieve end-to-end security which we justify via an implementation.

Implementing Muckle+. The implementation of Muckle+ to demonstrate its efficacy follows the typical structure of both a QKD security application in the sense of the ETSI QKD GS standard documents (and in particular, ETSI QKD GS 014 [ETS19]) and an authenticated key exchange using application well-understood from their use on the modern web. Thereby, the initiator of the connection obtains a key ID and the corresponding key material from a QKD device and transmits the key ID as part of the initial authenticated key exchange message to the receiver.

By providing an interface the applications that follow the structure of deployed authenticated key exchanges, we expect to reduce the required effort to integrate the use of QKD keys into applications that are already using TLS [Res18], QUIC [LRW⁺17], or similar protocols. Except for configuring the connection to the local QKD end-point, no further configuration will be necessary to establish secure channels with any service deployed on the QKD network.

On Further Directions to Achieve End-to-End HAKes. We expect that end-to-end HAKes can be built using further directions. Notably, Schwabe, Stebila and Wiggers proposed KEMTLS [SSW20], a unilaterally authenticated key exchange protocol where authentication of the responders is performed using a long-term KEM key. The basic idea is, that after establishing an ephemeral key, the initiator encapsulates a secret with respect to the responder’s long-term KEM key. The responder can only produce the authentication tags for session authentication if it can decapsulate using its long term KEM key. Thereby, the responder is implicitly authenticated via its knowledge of the corresponding private key. We chose to build Muckle+ with digital signatures for end-to-end authentication as a natural first step and leave extending Muckle+ with KEMTLS approaches for future work.

1.2 Related Work

Authenticated key exchanges have a long history and are still a very active area of research as they represent the core component of any protocol for secure communication. Notably, Krawczyk’s Sign-and-MAC (SIGMA) protocols [Kra03] serve as a template for many of the protocols used in practice. The basic idea of this template is to combine an ephemeral key exchange using key encapsulation methods (KEMs) to exchange a fresh shared secret, a signature scheme for authentication of the communication parties as well as a MAC to authenticate the shared secret. Keys are derived using a pseudorandom function (PRF). One execution then runs roughly as follows: the initiator produces a new ephemeral KEM key and sends the public key to the responder. The responder then performs the key encapsulation using the received public key, signs the produced ciphertext together with the first message to authenticate itself, and derives a shared secret to authenticate the session using the MAC. Ciphertext, authentication tag and signature are sent to the initiator. The initiator then decapsulates the shared secret key, verifies the received signature as well as the authentication tag. In a mutual authentication setting, the initiator also authenticates itself using the signature scheme, but the session is also always authenticated by the initiator using the MAC. This information is sent to the responder for verification. Afterwards, the two parties share an authenticated and fresh secret key.

While SIGMA was originally proposed using Diffie-Hellman for the ephemeral key exchange, presenting it in terms of KEMs allows us to consider it in a post-quantum setting as we then can instantiate all build

blocks using post-quantum secure schemes. It can also be extended with responder or initiator privacy features [Zha16,SSL20,RSW21], whereas the latter be observed in practice as part of the TLS handshake. With the migration towards post-quantum secure protocols, work on adapting and improving key exchange protocols based on the performance and bandwidth characteristics of post-quantum secure key encapsulation mechanisms and digital signature schemes has commenced [BCNS15,SM16,HKSU20,HNS⁺21], though.

In the area of QKD networks, proposals exist to address the trusted-node problem with secret-sharing-based multipath protocols, e.g. [RK11,RKJ⁺21], to exchange the secret key. In a similar vein, multipath authentication protocols have been proposed too, whereas those are built on the assumption that an adversary is unable to compromise multiple nodes in the network. When considering network topologies with many routes available for connecting any two nodes, it is therefore possible to split sensitive information into parts (e.g., via secret sharing) and to send the shares via multiple paths instead of one.

For example, Rass and Schartner [RS10] introduced a MAC-based multipath authentication protocol specifically for the application in quantum networks. In the scenario they consider, two nodes wanting to communicate in a QKD network may not necessarily establish pre-shared keys. There are however shared QKD secrets between every node and each of its immediate neighbors. The protocol uses those secrets in combination with a multipath approach to share an authenticated message between the nodes and relies on the assumptions that (a) keys created by two adjacent nodes via the QKD channel are secure, and (b) every node shares a secret key with its neighboring nodes. While the protocol is secure against $k < n$ compromised paths when executed with n disjoint paths, it does not fit into the typical notion of an authenticated key exchange and it lacks end-to-end authenticity.

Finally, secure multipath key exchange (SMKEX) [CCG⁺18] utilizes two disjoint paths to facilitate authentication and key exchange. The protocol is based on a typical key exchange, but in addition the second channel is used to send a random nonce that is authenticated using the secret key exchanged via the first channel. SMKEX therefore ensures unilateral authenticity and computational security against an active adversary as long as only one path is compromised.

2 Preliminaries

In this section, we briefly recall notions related to (hybrid) authenticated key exchanges.

2.1 Cryptographic Primitives and Schemes

Definition 1 (Pseudo-Random Function). Let $\mathcal{F} : \mathcal{S} \times \mathcal{D} \rightarrow \mathcal{R}$ be a family of functions and let Γ be the set of all functions $\mathcal{D} \rightarrow \mathcal{R}$. For a PPT distinguisher \mathcal{D} we define the advantage function as

$$\text{Adv}_{\mathcal{D},\mathcal{F}}^{\text{PRF}}(\kappa) = \left| \Pr \left[s \xleftarrow{R} \mathcal{S} : \mathcal{D}^{\mathcal{F}(s,\cdot)}(1^\kappa) = 1 \right] - \Pr \left[f \xleftarrow{R} \Gamma : \mathcal{D}^{f(\cdot)}(1^\kappa) = 1 \right] \right|.$$

\mathcal{F} is a pseudorandom function (family) if it is efficiently computable and for all PPT distinguishers \mathcal{D} there exists a negligible function $\varepsilon(\cdot)$ such that

$$\text{Adv}_{\mathcal{D},\mathcal{F}}^{\text{PRF}}(\kappa) \leq \varepsilon(\kappa).$$

A PRF \mathcal{F} is a dual PRF [BL15], if $\mathcal{G} : \mathcal{D} \times \mathcal{S} \rightarrow \mathcal{R}$ defined as $\mathcal{G}(d, s) = \mathcal{F}(s, d)$ is also a PRF.

We recall the notion of message authentication codes (MACs) as well as digital signature schemes, and the standard unforgeability notions below.

Definition 2 (Message Authentication Codes). A message authentication code MAC is a triple $(\text{KGen}, \text{Sign}, \text{Ver})$ of PPT algorithms, which are defined as:

$\text{KGen}(1^\kappa)$: This algorithm takes a security parameter κ as input and outputs a secret key sk .

$\text{Auth}(\text{sk}, m)$: This algorithm takes a secret key $\text{sk} \in \mathcal{K}$ and a $m \in \mathcal{M}$, and outputs an authentication tag τ .

$\text{Ver}(\text{sk}, m, \tau)$: This algorithm takes a secret key sk , a message $m \in \mathcal{M}$, and an authentication tag τ as input, and outputs a bit $b \in \{0, 1\}$.

A MAC is correct if for all $\kappa \in \mathbb{N}$, for all $\text{sk} \leftarrow \text{KGen}(1^\kappa)$ and for all $m \in \mathcal{M}$, it holds that

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

where the probability is taken over the random coins of KGen and Auth .

Definition 3 (EUF-CMA security of MAC). For a PPT adversary \mathcal{A} , we define the advantage function in the sense of existential unforgeability under chosen message attacks (EUF-CMA) as

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

where the corresponding experiment is depicted in Experiment 1. If for all PPT adversaries \mathcal{A} there is a negligible function $\varepsilon(\cdot)$ such that $\text{Adv}_{\mathcal{A}, \text{MAC}}^{\text{euf-cma}}(1^\kappa) \leq \varepsilon(\kappa)$, we say that MAC is EUF-CMA secure.

$\text{Exp}_{\mathcal{A}, \text{MAC}}^{\text{euf-cma}}(1^\kappa)$:

$\text{sk} \leftarrow \text{KGen}(1^\kappa), \mathcal{Q} \leftarrow \emptyset$

$(m^*, \tau^*) \leftarrow \mathcal{A}^{\text{Auth}', \text{Ver}'}(1^\kappa)$

where oracle $\text{Auth}'(m)$:

$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$

return $\text{Auth}(\text{sk}, m)$

where oracle $\text{Ver}'(m, \tau)$:

return $\text{Ver}(\text{sk}, m, \tau)$

return 1, if $\text{Ver}(\text{sk}, m^*, \tau^*) = 1 \wedge m^* \notin \mathcal{Q}$, return 0, otherwise

Experiment 1: EUF-CMA security experiment for a MAC MAC.

Definition 4 (Signature Scheme). A signature scheme Σ is a triple $(\text{KGen}, \text{Sign}, \text{Ver})$ of PPT algorithms, which are defined as follows:

$\text{KGen}(1^\kappa)$: This algorithm takes a security parameter κ as input and outputs a secret (signing) key sk and a public (verification) key pk with associated message space \mathcal{M} (we may omit to make the message space \mathcal{M} explicit).

$\text{Sign}(\text{sk}, m)$: This algorithm takes a secret key sk and a message $m \in \mathcal{M}$ as input, and outputs a signature σ .

$\text{Ver}(\text{pk}, m, \sigma)$: This algorithm takes a public key pk , a message $m \in \mathcal{M}$ and a signature σ as input, and outputs a bit $b \in \{0, 1\}$.

For correctness, we require that for all $\kappa \in \mathbb{N}$, for all $(\text{sk}, \text{pk}) \leftarrow \text{KGen}(1^\kappa)$ and for all $m \in \mathcal{M}$ it holds that

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

where the probability is taken over the random coins of KGen and Sign .

Definition 5 (EUF-CMA of Σ). For a PPT adversary \mathcal{A} , we define the advantage function in the sense of existential unforgeability under chosen message attacks (EUF-CMA) as

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

where the corresponding experiment is depicted in Experiment 2. If for all PPT adversaries \mathcal{A} there is a negligible function $\varepsilon(\cdot)$ such that $\text{Adv}_{\mathcal{A}, \Sigma}^{\text{euf-cma}}(1^\kappa) \leq \varepsilon(\kappa)$, we say that Σ is EUF-CMA secure.

$\text{Exp}_{\mathcal{A}, \Sigma}^{\text{euf-cma}}(1^\kappa)$:
 $(\text{sk}, \text{pk}) \leftarrow \text{KGen}(1^\kappa)$, $\mathcal{Q} \leftarrow \emptyset$
 $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}}(\text{pk})$
 where oracle $\text{Sign}'(m)$:
 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
 return $\text{Sign}(\text{sk}, m)$
 return 1, if $\text{Ver}(\text{pk}, m^*, \sigma^*) = 1 \wedge m^* \notin \mathcal{Q}$, return 0, otherwise

Experiment 2: EUF-CMA security experiment for a digital signature scheme Σ .

We recall the notion of key-encapsulations mechanisms (KEMs), and the standard chosen-plaintext and chosen-ciphertext notions below.

Definition 6 (Key-Encapsulation Mechanism). A key-encapsulation mechanism scheme KEM with key space \mathcal{K} consists of the three PPT algorithms (KGen, Enc, Dec):

$\text{KGen}(1^\kappa)$: This algorithm takes a security parameter κ as input, and outputs public and secret keys (pk, sk) .

$\text{Enc}(\text{pk})$: This algorithm takes a public key pk as input, and outputs a ciphertext c and key K .

$\text{Dec}(\text{sk}, c)$: This algorithm takes a secret key sk and a ciphertext c as input, and outputs K or $\{\perp\}$.

We call a KEM correct if for all $\kappa \in \mathbb{N}$, for all $(\text{pk}, \text{sk}) \leftarrow \text{KGen}(\kappa)$, for all $(c, K) \leftarrow \text{Enc}(\text{pk})$, we have that

$$\Pr[\text{Dec}(\text{sk}, c) = K] = 1,$$

where the probability is taken over the random coins of KGen and Enc.

Definition 7 (IND-CPA and IND-CCA security of KEM). For a PPT adversary \mathcal{A} , we define the advantage function in the sense of indistinguishability under chosen-plaintext attacks (IND-CPA) and indistinguishability under chosen-ciphertexts attacks (IND-CCA) as

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

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

where the corresponding experiments are depicted in Experiment 3. If for all PPT adversaries \mathcal{A} there is a negligible function $\varepsilon(\cdot)$ such that

$$\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(1^\kappa) \leq \varepsilon(\kappa) \text{ or } \text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cca}}(1^\kappa) \leq \varepsilon(\kappa),$$

then we say that KEM is IND-CPA or IND-CCA secure, respectively.

2.2 Hybrid Authenticated Key Exchange

We recall the hybrid authenticated key exchange (HAKE) security model due to Dowling et al. [DHP20] which already foresees the use of long-term post-quantum digital signature keys. For a general treatment of authenticated key exchanges (AKE), we refer the reader to [DvOW92, KL14]. The HAKE security experiment $\text{Exp}_{\mathcal{A}, \Pi, n_P, n_S, n_T}^{\text{hake, clean}}$ is described as follows:

$\text{Exp}_{\mathcal{A}, \text{KEM}}^{\text{ind-}T}(\kappa)$:
 $(\text{pk}, \text{sk}) \leftarrow \text{KGen}(1^\kappa)$
 $(c^*, K_0) \leftarrow \text{Enc}(\text{pk}), K_1 \xleftarrow{R} \mathcal{K}$
 $\mathcal{Q} \leftarrow \emptyset, b \xleftarrow{R} \{0, 1\}^\kappa$
 $b^* \leftarrow \mathcal{A}^\mathcal{O}(\text{pk}, c^*, K_b)$
 where $\mathcal{O} = \{\text{Dec}'\}$ if $T = \text{cca}$ with oracle $\text{Dec}'(c)$:
 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{c\}$
 return $\text{Dec}(\text{sk}, c)$
 return 1, if $b = b^* \wedge c^* \notin \mathcal{Q}$, return otherwise 0

Experiment 3: IND-CPA and IND-CCA security experiments for KEM with $T \in \{\text{cpa}, \text{cca}\}$.

Execution environment. We consider a set of n_P parties P_1, \dots, P_{n_P} which are able to run up to n_S sessions of a key-exchange protocol between them, where each session may consist of n_T different stages of the protocol. Each party P_i has access to its long-term key pair $(\text{pk}_i, \text{sk}_i)$ and to the public keys of all other parties. Each session is described by a set of session parameters:

- $\rho \in \{\text{init}, \text{resp}\}$: The role (initiator or responder) of the party during the current session.
- $\text{pid} \in n_P$: The communication partner of the current session.
- $\text{stid} \in n_T$: The current stage of the session.
- $\alpha \in \{\text{active}, \text{accept}, \text{reject}, \perp\}$: The status of the session. Initialized with \perp .
- $m_i[\text{stid}], i \in \{s, r\}$: All messages sent ($i = s$) or received ($i = r$) by a session up to the stage stid . Initialized with \perp .
- $k[\text{stid}]$: All session keys created up to stage stid . Initialized with \perp .
- $\text{exk}[\text{stid}], x \in \{q, c, s\}$: All ephemeral post-quantum (q), classical (c) or symmetric (s) secret keys created up to stage stid . Initialized with \perp .
- $\text{pss}[\text{stid}]$: The per-session secret state (SecState) that is created during the stage stid for the use in the next stage.
- $\text{st}[\text{stid}]$: Storage for other states used by the session in each stage.

We describe the protocol as a set of algorithms $(f, \text{KGen}XY, \text{KGen}ZS)$:

- $f(\kappa, \text{pk}_i, \text{sk}_i, \text{pskid}_i, \text{psk}_i, \pi, m) \rightarrow (m', \pi')$: A probabilistic algorithm that represents an honest execution of the protocol. It takes a security parameter κ , the long-term keys $(\text{pk}_i, \text{sk}_i)$, the session parameters π representing the current state of the session, and a message m , and outputs a response m' and the updated session state π' .
- $\text{KGen}XY(\kappa) \rightarrow (\text{pk}, \text{sk})$: A probabilistic asymmetric key-generation algorithm that takes a security parameter κ and creates a public-secret-key pair (pk, sk) . $X \in \{E, L\}$ determines whether the created key is an ephemeral (E) or a long-term (L) secret key. $Y \in \{Q, C\}$ determines whether the key is classical (C) or post-quantum (Q).
- $\text{KGen}ZS(\kappa) \rightarrow (\text{psk}, \text{pskid})$: A probabilistic symmetric key-generation algorithm that takes a security parameter κ and outputs symmetric keying material (psk) . $Z \in \{E, L\}$ determines whether the created key is an ephemeral (E) or a long-term (L) secret key.

For each party P_1, \dots, P_{n_P} , classical as well as post-quantum long-term keys are created using the corresponding $\text{KGen}XY$ algorithms. The challenger then queries a uniformly random bit $b \leftarrow \{0, 1\}$ that will determine the key returned by the Test query. From this point on, the adversary may interact with the challenger using the queries defined in the next section. At some point during the execution of the protocol, the adversary \mathcal{A} may issue the Test query and present a guess for the value of b . If \mathcal{A} guesses correctly and the session satisfies the cleanness predicate, the adversary wins the key-indistinguishability experiment.

Adversarial Interaction. The HAKE framework defines a range of queries that allow the attacker to interact with the communication:

- $\text{Create}(i, j, \text{role}) \rightarrow \{(s), \perp\}$: Initializes a new session between party P_i with role role and the partner P_j . If the session already exists, then the query returns \perp , otherwise the session (s) is returned.
- $\text{Send}(i, s, m) \rightarrow \{m', \perp\}$: Enables \mathcal{A} to send messages to sessions and receive the response m' by running f for the session π_i^s . Returns \perp if the session is not active.
- $\text{Reveal}(i, s, t)$: Provides \mathcal{A} with the session keys corresponding to a session π_i^s if the session is in the accepted state. Otherwise, \perp is returned.
- $\text{Test}(i, s, t) \rightarrow \{k_b, \perp\}$: Provides \mathcal{A} with the real (if $b = 1$) or random ($b = 0$) session key for the key-indistinguishably experiment.
- $\text{CorruptXY}(i) \rightarrow \{\text{key}, \perp\}$: Provides \mathcal{A} with the long-term $XY \in \{\text{SK}, \text{QK}, \text{CK}\}$ keys for P_i . If the key has been corrupted previously, then \perp is returned. Specifically:
 - CorruptSK : Reveals the long-term symmetric secret (if available).
 - CorruptQK : Reveals the post-quantum long-term key (if available).
 - CorruptCK : Reveals the classical long-term key (if available).
- $\text{CompromiseXY}(i, s, t) \rightarrow \{\text{key}, \perp\}$: Provides \mathcal{A} with the ephemeral $XY \in \{\text{QK}, \text{CK}, \text{SK}, \text{SS}\}$ 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 : Reveals the ephemeral post-quantum key.
 - CompromiseCK : Reveals the ephemeral classical key.
 - CompromiseSK : Reveals the ephemeral quantum key.
 - CompromiseSS : Reveals the ephemeral per session state (SecState).

Matching sessions. Furthermore, we recall the definitions of matching sessions [LKZC07] and origin sessions [CF12] which covers that the two parties involved in a session have the same view of their conversation.

Definition 8 (Matching sessions). We consider two sessions π_i^s and π_j^r in stage t to be matching if all messages sent by the former session $\pi_i^s.m_s[t]$ match those received by the later $\pi_j^r.m_r[t]$ and all messages sent by the later session $\pi_j^r.m_s[t]$ are received by the former $\pi_i^s.m_r[t]$.

π_i^s is considered to be prefix-matching with π_j^r if $\pi_i^s.m_s[t] = \pi_j^r.m_r[t']$ where $\pi_j^r.m_r[t]$ is truncated to the length of $\pi_i^s.m_s[t]$ resulting in $\pi_j^r.m_r[t']$.

Definition 9 (Origin sessions). We consider a session π_i^s to have an origin session with π_j^r if π_i^s matches π_j^r or if π_i^s prefix-matches π_j^r .

HAKE security. Dowling et al. [DHP20] define key indistinguishability (i.e., what we dub HAKE security) with respect to a predicate clean . However, their predicate is specific to Muckle and, hence, we therefore only give the formal security definition next and postpone the discussion of the predicate to Section 3.3.

Definition 10 (HAKE security). Let Π be a key-exchange protocol and $n_P, n_S, n_T \in \mathbb{N}$. For a predicate clean and an adversary \mathcal{A} , we define the advantage of \mathcal{A} in the HAKE key-indistinguishability game as

$$\text{Adv}_{\mathcal{A}, \Pi, n_P, n_S, n_T}^{\text{hake, clean}}(\kappa) = \left| \Pr \left[\text{Exp}_{\mathcal{A}, \Pi, n_P, n_S, n_T}^{\text{hake, clean}}(\kappa) = 1 \right] \right|.$$

We say that Π is HAKE-secure if $\text{Adv}_{\mathcal{A}, \Pi, n_P, n_S, n_T}^{\text{hake, clean}}(\kappa)$ is negligible in the security parameter κ for all \mathcal{A} .

3 Extending Muckle With Signature-Based Authentication

In this section, we recap Muckle [DHP20] and present our novel variant Muckle+.

3.1 Muckle

The Muckle protocol combines classical, PQC, and QKD keys through the use of a key derivation function. More concretely, Muckle requires classical and post-quantum key encapsulation mechanisms (KEMs) as well as data from a QKD channel (i.e., a symmetric key k_q) to create the final shared secret between communication partners.

Muckle is a multi-stage protocol. While a Muckle instance is active between two parties, a single stage is run repeatedly, creating a pair of session keys during each execution. The communication that occurs during one stage of the protocol is detailed in Figure 1.

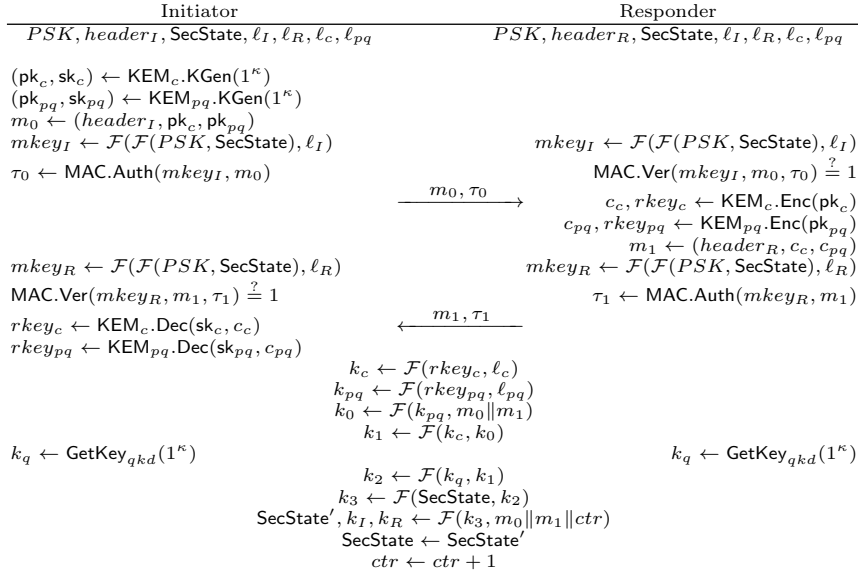


Fig. 1. One stage of the Muckle protocol [DHP20] with a classical KEM KEM_c , a post-quantum KEM KEM_{pq} , a MAC MAC , and a PRF \mathcal{F} whereas k_q represents the symmetric key from the QKD component (provided out-of-band via function $GetKey_{qkd}$).

The Muckle key exchange requires a symmetric pre-shared key PSK and unique party identifiers (implicit in ℓ_I and ℓ_R) to be distributed to the communication partners before the key exchange. The parties also have to set an initial value for the session secret state $SecState$. To begin a new session, the initiator uses the classical KEM KEM_c and post-quantum KEM KEM_{pq} to create a classical key pair (pk_c, sk_c) and a post-quantum key pair (pk_{pq}, sk_{pq}) , respectively. Both public keys are then combined with a header containing meta-data into the message m_0 . The PRF \mathcal{F} is applied over PSK and $SecState$ to create a unique value for the current session, which is then used as an input in another round of the PRF with the value ℓ_I resulting in the message key $mkey_I$. The key $mkey_I$ is used as the MAC key to create a tag τ_0 for the message m_0 . The message m_0 and the tag τ_0 are then sent to the responder.

Receiving the transmission, the responder will check the authenticity of the message m_0 by verifying the tag τ_0 with its $mkey_I$ (where $mkey_I$ is derived via PSK , shared $SecState$, and ℓ_I). If the verification succeeds, the responder can now use the encapsulation functions of the KEMs to create the keys $rkey_c$ and $rkey_{pq}$ as well as the ciphertexts c_c and c_{pq} , respectively. The responder proceeds to create a message m_1 and a tag τ_1 analogously to the initiator's MAC procedure, but using the ciphertexts instead of the public keys and the responder value $header_R$.

Table 1. Comparison of the protocols in terms of provided security guarantees: KC (key confirmation), PFS (perfect forward secrecy; synonymously to forward security), and PCS (post-compromise security).

	protocol	authentication	KC	PFS	PCS
multipath	MAC-based [RS10]	initiator			
	SMKEX [CCG ⁺ 18]	responder		*	*
post-quantum	SIGMA [Kra03]	explicit mutual	✓	✓	
	KEMTLS [SSW20]	explicit ¹ responder/mutual	✓ ²	✓	? ³
	Muckle [DHP20]	explicit mutual	✓	✓	✓ ⁴

* not applicable (no long-term secret)

¹ implicit for client during mutual authentication

² only for responder authentication

³ PCS is not explicitly shown

⁴ under the conditions discussed in Section 3.1

m_1 and τ_1 are then transmitted to the initiator, who can use them in the KEM decapsulation function to get the keys $rkey_c$ and $rkey_{pq}$ (after successful verification of m_1). From this point on, the initiator and responder share the same information and proceed with the same steps.

First, the both keys are entered into the PRF \mathcal{F} together with labels ℓ_c and ℓ_{pq} to create the further keys. Then, the key schedule starts combining all the keys into a final shared secret k_I, k_R and setting a new session state as well as incrementing the session counter.

Muckle offers mutual authentication, forward security, and post-compromise security. Post-compromise security is guaranteed under the condition that at least one previous stage has been completed without the attacker compromising all the ephemeral (classical, QKD, post-quantum and session secret) secrets, and that the attacker has been only acting passively since then.

3.2 Extending Muckle With Signature-Based Authentication

In Table 1, we compare the security properties of the protocols we have discussed in the introduction and Muckle. From this comparison, we can conclude that Muckle offers the most features and is therefore a suitable candidate for realizing end-to-end secure hybrid authenticated key exchanges. However, the protocol relies on PSKs for end-to-end authentication. As the other components including the QKD-layer do not provide end-to-end authentication (cf. Section 3.1), we extend Muckle to also offer mutual signature-based authentication. Through this alteration, we preserve the desirable security whilst avoiding the issues associated with PSKs (given that we can rely on multi-path QKD). We will from now on refer to this new protocol as Muckle+.

Like Muckle, Muckle+ is a multi-stage protocol. One such stage is detailed in Figure 2. The basic structure of Muckle+ is very similar to the original Muckle protocol. Up to the computation of the final chaining key, the PSK-based authentication is replaced with signature-based authentication and the addition of two random nonces n_I and n_R to avoid issues with the reuse of signatures. We note that the modifications essentially correspond to changing to a SIGMA-style key exchange with multiple KEMs and an additional ephemeral secret that is provided by the QKD link. We note that the correctness of the protocol follows directly from the correctness of the employed primitives.

3.3 Security of Muckle+

Similar to Muckle, Muckle+ achieves the same security properties including forward security, or, mostly called perfect forward secrecy (PFS) in the AKE regime, and post-compromise security (PCS). In this section, we formally proof this claim. The presented security analysis of the Muckle+ protocol is based on the HAKE framework as introduced by Dowling et al. [DHP20]. We will use the definitions and notations use in the HAKE framework in this analysis unless stated otherwise.



Fig. 2. One stage of the Muckle+ protocol with a classical KEM KEM_c , a post-quantum KEM KEM_{pq} , a MAC Σ , and a PRF \mathcal{F} whereas k_q represents the symmetric key from the QKD component (provided out-of-band via function GetKey_{qkd}). $cert_I$ and $cert_R$ are certificates (provided out-of-band) for the public keys pk_I and pk_R , respectively. $CATS$ and $SATS$ are the client and server application traffic secrets, respectively. Messages $m_i: \{m_{i,1}, \dots\}_k$ denote that $m_{i,1}, \dots$ is encrypted with an authenticated encryption scheme using the secret key k . The various contexts and labels are given in Tables 2 and 3.

Table 2. Values for the contexts used in the Muckle+ key schedule. The context inputs follow the choices in the TLS 1.3 handshake [DFGS21].

Label	Context Input	Label	Context Input
H_ε	“”	H_0	$H(\text{“”})$
H_1	$H(\mathbf{m}_1 \parallel \mathbf{m}_2)$	H_2	$H(\mathbf{m}_1 \parallel \dots \parallel \mathbf{m}_3)$
H_3	$H(\mathbf{m}_1 \parallel \dots \parallel \mathbf{m}_4)$	H_4	$H(\mathbf{m}_1 \parallel \dots \parallel \mathbf{m}_5)$
H_5	$H(\mathbf{m}_1 \parallel \dots \parallel \mathbf{m}_6)$	H_6	$H(\mathbf{m}_1 \parallel \dots \parallel \mathbf{m}_7)$

Table 3. Values for the labels used in the Muckle+ key schedule for domain separation. Some of these labels are directly based on the corresponding labels in the TLS 1.3 handshake [DFGS21]. The concrete value of these labels is unimportant as long as they are unique.

Label	Label Input	Label	Label Input
ℓ_0	“derive k c”	ℓ_1	“derive k pq”
ℓ_2	“first ck”	ℓ_3	“second ck”
ℓ_4	“third ck”	ℓ_5	“fourth ck”
ℓ_6	“derived”	ℓ_7	“c hs traffic”
ℓ_8	“s hs traffic”	ℓ_9	“finished”
ℓ_{10}	“c ap traffic”	ℓ_{11}	“s ap traffic”
ℓ_{12}	“secstate”	ℓ_{13}	“TLS 1.3, server CertificateVerify”
ℓ_{14}	“TLS 1.3, client CertificateVerify”		

An adversary \mathcal{A} has access to all queries defined in the HAKE framework. As no pre-shared key exists in the Muckle+ protocol, the query `CorruptSK` will return \perp if called. As multiple sessions keys are created in the new protocol, we specify that the key to be guessed during the `Test` query is the master secret MS.

We define a new cleanness predicate $\text{clean}_{\text{Muckle+}}$ for our protocol that captures the same goals – post-compromise security and perfect forward secrecy – but adapt it to match our setting. As our protocol does not require a long term PSK, we can omit handling compromise of the PSK in our predicate. We however have to take care of long-term signature keys instead. Hence, we consider their compromise in $\text{clean}_{\text{Muckle+}}$ as well. Overall, the goal of the cleanness predicate is to handle the compromise of as many combinations as possible as long as one set of keys – the post-quantum secure keys or the keys obtained from the QKD link – stay secure.

More formally, we define the cleanness of a session as follows. A session π_i^s in stage t is considered clean under the predicate $\text{clean}_{\text{Muckle+}}$ if:

- `Reveal`(i, s, t) has not been issued for session π_i^s .
- `Reveal`(j, r, t) has not been issued for all sessions π_j^r matching π_i^s at stage t .
- If π_i^s has a matching session π_j^r , at least one of the following conditions has been met:
 - No `CompromiseQK`(i, s, t) or `CompromiseQK`(j, r, t) have been issued.
 - No `CompromiseSK`(i, s, t) or `CompromiseSK`(j, r, t) have been issued.
 - No `CompromiseQK`(i, s, t') or `CompromiseQK`(j, r, t) have been issued with π_i^s matching π_j^r in stages u where $t' \leq u \leq t$. No `CompromiseSS`(i, s, u) or `CompromiseSS`(j, r, u) have been issued.
 - No `CompromiseSK`(i, s, t') or `CompromiseSK`(j, r, t') have been issued with π_i^s matching π_j^r in stages u where $t' \leq u \leq t$. No `CompromiseSS`(i, s, u) or `CompromiseSS`(j, r, u) have been issued.
- 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 either `CompromiseSK`(i, j, t) and `CompromiseSK`(j, i, t) or `CorruptQK`(i) and `CorruptQK`(j) have not been issues before $\pi_i^s.\alpha[t] \leftarrow \text{accept}$. If there exists $(j, r, t) \in [n_P] \times [n_S] \times [n_T]$ such that π_j^r is an origin session of π_i^s in stage t , then either `CompromiseSK`(i, j, t) and `CompromiseSK`(j, i, t) or `CorruptQK`(i) and `CorruptQK`(j) have not been issued before $\pi_i^s.\alpha[t] \leftarrow \text{accept}$.

The first condition ensures, that the session key of the session used in the `Test` query has not been revealed to the adversary through the use of the `Reveal` query. Similarly, the second condition specifies that no

session matching the test session may have been targeted by a `Reveal` query either, as any matching session will own the same session key as the test session. The third condition ensures that at least one ephemeral secret is not compromised by \mathcal{A} or that the secret session state in the multi-stage setting. Finally, the fourth case restricts access to one of the long-term secrets for the first round without origin session to exclude otherwise trivial impersonation attacks.

We note that similar to $\text{clean}_{\text{Muckle}}$, we can define classical and quantum variants of the predicate to also reflect compromise of the classical keys. In that case, $\text{clean}_{\text{cMuckle+}}$ is extended to include the following two conditions for matching sessions π_i^s and π_j^r :

- No $\text{CompromiseCK}(i, s, t)$ or $\text{CompromiseCK}(j, r, t)$ have been issued.
- No $\text{CompromiseCK}(i, s, t')$ or $\text{CompromiseCK}(j, r, t')$ have been issued with π_i^s matching π_j^r in stages u where $t' \leq u \leq t$. No $\text{CompromiseSS}(i, s, u)$ or $\text{CompromiseSS}(j, r, u)$ have been issued.

We will now prove that the proposed protocol is secure with the cleanness predicate $\text{clean}_{\text{Muckle+}}$. In order to do so, we analyze the five cases corresponding to the conditions that are necessary to fulfil the $\text{clean}_{\text{Muckle+}}$ predicate.

Theorem 1. *The Muckle+ key exchange protocol is HAKE-secure with the cleanness predicate $\text{clean}_{\text{Muckle+}}$ assuming that the PRF \mathcal{F} is a dual PRF, the MAC MAC is EUF-CMA secure, the KEMs KEM_c and KEM_{pq} are IND-CPA secure and the signature scheme Σ is EUF-CMA secure. If the security of \mathcal{F} , MAC , KEM_{pq} and Σ or of QKD hold against a quantum adversary, then so does the security of Muckle+.*

Proof. We divide the proof into different cases where the query $\text{Test}(i, s, t)$ has been issued and prove them separately:

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

Similar to the proof of Muckle, we show the first and the third case. The second case follows analogously to the first case.

Case 1: Test init session without origin session. In case 1, we show that \mathcal{A} has negligible chance of getting a session to reach the `accept` state if a `CorruptQK` or a `CompromiseSK` query has been issued. If the session does not reach the `accept` stage, the `Test` query will always return \perp , preventing \mathcal{A} from winning the indistinguishability game. First we consider the case that no `CorruptQK` query has been issued.

Game 0: Standard HAKE-experiment with advantage

$$\text{Adv}_{\mathcal{A}, \text{Muckle+}, n_P, n_S, n_T}^{\text{HAKE}, \text{clean}_{\text{Muckle+}}, C_1}(\kappa) = \Pr[S_0].$$

Game 1: In Game 1, the parameters (i, s, t) for a session and its matching session (j, r, t) are guessed. If a $\text{Test}(i', s', t)$ query is issued for any session $\pi_{i'}^{s'}$ that is not the test session π_i^s , the game aborts. The advantage is

$$\Pr[S_0] \leq n_P^2 n_S n_T \cdot \Pr[S_1].$$

Game 2: Game 2 aborts, if the test session π_i^s ever reaches the status `reject`. As the `Test` query will always return \perp if the session reaches this status, the advantage gained by \mathcal{A} is 0. The advantage is

$$\Pr[S_0] \leq n_P^2 n_S n_T \cdot \Pr[S_2].$$

Game 3: Game 3 aborts, if the session reaches the status `accept`. The advantage is

$$\Pr[S_0] \leq n_P^2 n_S n_T \cdot \Pr[S_3].$$

We now bound the probability of \mathcal{A} reaching the abort event. Assuming that the session reaches the status `accept`, we construct an EUF-CMA adversary against Σ . The challenge `pk` is used as the party's public key. For all other sessions, the signing oracle is used to produce the corresponding signatures. Now, if the test session reaches `accept` stage, we output the signature σ_I as forgery on the message $\ell_{14}||H_5$. The signature verifies since `accept` stage was reached and has not been queried to the signing oracle (except for collisions of the hash function H). Hence, we obtain:

$$\Pr[S_0] \leq n_P^2 n_S n_T \cdot \left(\text{Adv}_{\mathcal{A}, \Sigma}^{\text{euf-cma}}(\kappa) \right).$$

The case that no `CompromiseSK` query has been issued before reaching the `accept` stage, follows analogously to [DHP20, Theorem 1, Case 1] and is not repeated here.

Case 3: Test session with matching session. We will show that any adversary \mathcal{A} has a negligible chance of winning the key-indistinguishability game using a sequence of games for each of the four cases. We denote with S_i the event of the adversary winning game i . Note that the proofs are the same regardless of whether $\text{KEM} \in \{\text{KEM}_c, \text{KEM}_{pq}\}$, whereas security against a quantum adversary can only be achieved for $\text{KEM} = \text{KEM}_{pq}$. We split the proof into several subcases.

Subcase 1: No `CompromiseQK(i, s, t)` or `CompromiseQK(j, r, t)` have been issued. Subcase 1 shows, that if the attacker issues a `Test` query to a session that is clean due to the secrecy of the ephemeral post-quantum key, the attacker has a negligible advantage in guessing the test bit. In this scenario, all ephemeral secrets except the post-quantum key as well as the long-term classical and post-quantum secrets are known to the attacker.

Game 0: Standard HAKE-experiment with advantage

$$\text{Adv}_{\mathcal{A}, \text{Muckle}^+, n_P, n_S, n_T}^{\text{HAKE}, \text{cleanMuckle}^+, C_2}(\kappa) = \Pr[S_0].$$

Games 1-7: Games 1 to 7 for `Muckle` are equivalent to the Games 1 to 7 of the proof of case 3.1 as described in [DHP20], resulting in the following advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \left(\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(\kappa) + 2 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\lambda) + 3 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

Game 8: In Game 8, the computation of the derived handshake secret `dHS` is replaced by a uniformly random value. To achieve this, ℓ_6 is queried together with the context input H_0 and a PRF challenger is initialized for the computation. The output of the challenger is used to replace the `dHS` secret. As k_3 is uniformly random by Game 7, this is a valid replacement. To distinguish between the case where $dHS \leftarrow \mathcal{F}(k_3, \ell_6, H_0)$ or $dHS \xleftarrow{R} \{0, 1\}^\kappa$ the attacker would have to break the **prf** security of PRF and thus has the following advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \left(\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(\kappa) + 3 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\lambda) + 3 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

Game 9: In Game 9, the derivation of the master secret `MS` is replaced by a uniformly random value. A PRF challenger is initialised and its output used to replace `MS`. Since `dHS` is already random by Game 8, this is a valid substitution. To distinguish between the case, where $MS \leftarrow \mathcal{F}(dHS, 0)$ or $M \xleftarrow{R} \{0, 1\}^\kappa$, \mathcal{A} would have to break the **prf** security of PRF which leaves the attacker with the following advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \left(\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(\kappa) + 4 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\lambda) + 3 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

Game 10: In Game 10, the application traffic secrets (`CATS`, `SATS`) and the session state `SecState` are replaced by a uniformly random value. This is done by initializing a PRF challenger for each

computation and querying the labels 10,11 and 12 as well as the context input H_4 and replacing the corresponding value with the output from the challenger. Since the master secret MS is already random by Game 9, this is a valid substitution. For \mathcal{A} to distinguish between the case where $CATS, SATS, \text{SecState} \leftarrow \mathcal{F}(MS, \ell_{\{10,11,12\}}, H_4)$ or $CATS, SATS, \text{SecState} \xleftarrow{R} \{0,1\}^{\mathcal{F}}$, it would have to break the **prf** security of PRF.

At this point, the application traffic secrets and the session state are shown to be uniformly random under the condition of case 2 and \mathcal{A} has an advantage of

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \left(\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(\kappa) + 5 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\lambda) + 3 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

Subcase 2: No CompromiseSK(i, s, t) or CompromiseSK(j, r, t) have been issued. This case shows, that if the attacker issues a **Test** query to a session that is clean due to the secrecy of the ephemeral quantum key, the attacker has a negligible advantage in guessing the test bit. In this scenario, all ephemeral secrets except the quantum key as well as the long-term classical and post-quantum secrets are known to the attacker.

Game 0: Standard HAKE-experiment with advantage

$$\text{Adv}_{\mathcal{A}, \text{Muckle}+, n_P, n_S, n_T}^{\text{HAKE}, \text{cleanMuckle}+, C_3}(\kappa) = \Pr[S_0].$$

Games 1-3: Games 1 to 3 for Muckle are equivalent to Games 1 to 3 of the proof of case 3.2 as described in [DHP20], resulting in the following advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \left(\text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\kappa) + \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

Games 4-6: Games 4 to 6 are equivalent to Games 8 to 10 in subcase 1, resulting in the final advantage of

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \left(4 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\kappa) + \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

Subcase 3: No CompromiseQK(i, s, t') or CompromiseQK(j, r, t') have been issued with π_i^s matching π_j^r in stages u where $t' \leq u \leq t$. No CompromiseSS(i, s, u) or CompromiseSS(j, r, u) have been issued. This case shows, that if a previous session has been completed cleanly under the predicate $\text{cleanMuckle}+$ and \mathcal{A} has not compromised the session state SecState since then, the attacker has a negligible advantage in guessing the test bit of the current session.

Game 0: Standard HAKE-experiment with advantage

$$\text{Adv}_{\mathcal{A}, \text{Muckle}+, n_P, n_S, n_T}^{\text{HAKE}, \text{cleanMuckle}+, C_4}(\kappa) = \Pr[S_0].$$

Game 1: In Game 1, the parameters (i, s, t) for a session and its matching session (j, r, t) , as well as the stage t' are guessed. If \mathcal{A} issues a **Test**(i', s', t) query for any session $\pi_{i'}^s$ that is not the test session π_i^s the game aborts.

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2$$

Games 2-10: Games 2 to 10 are equivalent to Games 2 to 10 in subcase 1. The advantage is

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \left(\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(\kappa) + 5 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\lambda) + 3 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

After Game 10, the session π_i^s has been completed cleanly in stage t' . The following Games take place in each stage u and are therefore executed not once, but u -times. To represent the worst case scenario where \mathcal{A} has compromised every stage after the first one, we replace the factor u by n_T .

Game 11: In Game 11, the computation of k_3 is replaced by a uniformly random value. This is done by initializing a post-quantum PRF challenger with the value k_2 and replacing k_3 with the output. As SecState is uniformly random by Game 10, this is a valid substitution. To distinguish between the case of $k_3 \leftarrow \mathcal{F}(\text{SecState}, k_2)$ or $k_3 \xleftarrow{R} \{0, 1\}^{\mathcal{F}}$, the attacker would have to break the **prf** security of the PRF resulting in the advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \left(\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(\kappa) + (5 + n_T) \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\lambda) + 3 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

Games 12-14: Games 12 to 14 are equivalent to Games 8 to 10 in case 2:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \left(\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(\kappa) + (5 + 4n_T) \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\lambda) + 3 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

Subcase 4: No $\text{CompromiseSK}(i, s, t')$ or $\text{CompromiseSK}(j, r, t')$ have been issued with π_i^s matching π_j^r in stages u where $t' \leq u \leq t$. No $\text{CompromiseSS}(i, s, u)$ or $\text{CompromiseSS}(j, r, u)$ have been issued. This case shows, that if a previous session has been completed cleanly under the predicate $\text{clean}_{\text{Muckle+}}$ and \mathcal{A} has not compromised the session state SecState since then, the attacker has a negligible advantage in guessing the test bit of the current session.

Game 0: Standard HAKE-experiment with advantage

$$\text{Adv}_{\mathcal{A}, \text{Muckle+}, n_P, n_S, n_T}^{\text{HAKE}, \text{clean}_{\text{Muckle+}}, C_5}(\kappa) = \Pr[S_0].$$

Game 1: In Game 1, the parameters (i, s, t) for a session and its matching session (j, r, t) , as well as the stage t' are guessed. If \mathcal{A} issues a **Test** query for any session that is not the test session π_i^s the game aborts. The advantage is

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \Pr[S_1].$$

Games 2-6: Games 2 to 6 are equivalent to Games 2 to 6 in subcase 2. The advantage is

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \left(4 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\kappa) + \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

After Game 6 the session π_i^s has been completed cleanly in stage t' . The following Games take place in each stage u and are therefore executed not once, but u -times. To represent the worst case scenario where \mathcal{A} has compromised every stage after the first one, we replace the factor u by n_T .

Games 7-10: Games 7 to 10 are equivalent to Games 11 to 14 in subcase 3. The advantage is

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \left((4 + 4n_T) \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\kappa) + \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right).$$

Finally, we obtain the following advantage:

$$\begin{aligned} \text{Adv}_{\mathcal{A}, \text{Muckle+}, n_P, n_S, n_T}^{\text{HAKE}, \text{clean}_{\text{Muckle+}}}(\kappa) &\leq \\ &n_P^2 n_S^2 n_T^2 \cdot \text{Adv}_{\mathcal{A}, \Sigma}^{\text{euf-cma}}(\kappa) + \\ &n_P^2 n_S^2 n_T^2 \cdot \left(\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(\kappa) + 9 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\lambda) + 4 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right) + \\ &n_P^2 n_S^2 n_T^2 \cdot \left(\text{Adv}_{\mathcal{A}, \text{KEM}}^{\text{ind-cpa}}(\kappa) + (9 + 8n_T) \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{prf}}(\kappa) + 4 \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{dual-prf}}(\lambda) \right). \end{aligned}$$

3.4 Instantiating Muckle+

Finally, we discuss some possible choices when instantiating the primitives used in Muckle+ . Especially in QKD networks providing high bandwidth communication, the sizes of ciphertexts and signatures might not be a limiting factor. For the choice of signature schemes, we can thus consider candidates that are built from hash functions such as XMSS [BDH11, HBG⁺18] and SPHINCS+ [BHK⁺19] or block ciphers

such as Picnic [CDG⁺17,KKW18] and its variant built from AES [BDK⁺21]. Considering that in high bandwidth networks, the use of these symmetric primitives is perfectly valid to reduce the consumption of QKD keys, the use of these signature schemes does not require the addition of any new hardness assumptions to the overall system. We however want to note this chance comes with an increased runtime cost compared to KEM-based authentication as in KEMTLS.

We however want to note, that with the introduction of a signature-based authentication mechanism, the question arises on how to authenticate the other peer’s public key. Note though, that even if all public keys are shared a priori, the complexity is reduced to n keys instead of $\mathcal{O}(n^2)$ pre-shared keys. With the introduction of a Public Key Infrastructure (PKI) such as PKIX [CSF⁺08], the amount of pre-installed public keys that then serve as certificate authority (CA) can be drastically reduced. In a setting with only one provider, this can be a single CA. With more providers, various different scenarios can be considered with one external CA or multiple CAs where, for example, each provider handles the certification of the public keys used by their network components.

For QKD networks with trusted nodes, we note that all trusted nodes have access to the QKD key. In the HAKE security model, we thus need to assume that `CompromiseSK` has been queried and therefore the security of Muckle+ solely relies on the security of the KEM and signature scheme. To achieve fault tolerance in such a setting, we can consider multipath QKD systems that apply a typical secret-sharing-based approaches, e.g., [KGSR02,FFGV07]. Thereby, the QKD key is shared on the initiator side and the shares are transported via mutually disjoint paths in the QKD network to the receiver. Such an approach has been considered to some extent in the literature specifically for QKD networks, e.g., to boost throughput [YLL⁺21] and with semi-trusted and fully-trusted paths [CCCZ23] to increase the security of the network. The latter focuses specifically on the routing algorithms without going into details on the method to share the keys. By applying the techniques, e.g., from [KGSR02] to the QKD keys, and under the assumption that at least one path is non-compromised or more specifically – similar to the non-collusion in multiparty computation systems – that none of the nodes on disjunct paths collude, the risk stemming from trusted nodes can be mitigated.

4 Implementation and Evaluation

In order to evaluate the performance of Muckle+ in practical application, we implemented a prototype of the protocol. This prototype was implemented in Python using bindings⁹ of `liboqs` [SM16] for the support of post-quantum primitives and the `cryptography`¹⁰ module for all classically-secure schemes. As displayed Figure 3, the Muckle+ protocol operates on the application layer. The quantum key material is fetched by all endpoints by their respective key managements services (KMS) that provide key material to applications via the interfaces from ETSI GS QKD 014 [ETS19].

While the Muckle+ protocol allows for server-only, as well as mutual authentication, we benchmarked the implementation with mutual authentication. Authentication of both parties is achieved through the use of hybrid certificates containing both post-quantum and classical long term public keys. Certificates were signed with classical (EdDSA [BDL⁺11]) and post-quantum signatures. We assumed a 2-tier certificate hierarchy to simulate a PKI hierarchy for Muckle+ reflecting current practice, e.g., similar to Let’s Encrypt [ABC⁺19].

Our Muckle+ implementation was set up using a small network with three QKD links offering two mutually disjoint paths between endpoints. Initiator and responder of the protocol were executed on a notebook running Windows 10 with an Intel i5 2.60GHz CPU and 8 GB of RAM. Several instantiations of the protocol using different post-quantum KEMs and signature schemes were tested, resulting in the execution times displayed in Figure 4 for directly linked nodes. For all executions of the protocol, the remaining primitives have been instantiated with X25519 [Ber06] as KEM_c , HKDF-SHA2 as PRF and HMAC-SHA2 as MAC.

⁹ <https://github.com/open-quantum-safe/liboqs-python>

¹⁰ <https://pypi.org/project/cryptography/>

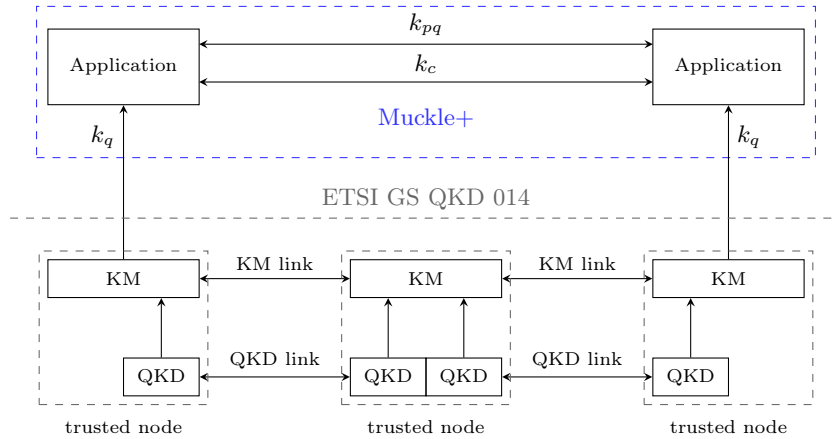


Fig. 3. Architecture of a Muckle+ implementation with a single intermediate node.

Figure 4 depicts the runtime of the initiator for various choices of signature schemes and KEMs for the initiator.¹¹ For the majority of the evaluated schemes, the runtime for a single Muckle+ stage ranged from 0.4 to 1.6 seconds. An average of approximately 0.3 seconds of this runtime can be attributed to the retrieval of the QKD key. Hence, we could demonstrate that the determining factor in the performance of the Muckle+ protocol is the key rate of the QKD link as in the original Muckle protocol. While the use of signature-based authentication incurs an overhead compared to Muckle, it is comparatively small in relation to the costs of accessing QKD keys.

An exception to the observation above is the small and robust SHA256 and SHAKE variants of SPHINCS+ where the slower runtime is attributed to the lack of support for the AVX2 instruction set in `liboqs` for SPHINCS+ on Windows.¹² Hence, we expect the performance of SPHINCS+ to be less of an issue with the availability of optimized implementation on Windows.

In Figure 5, the results of the same experiment with a multi-path setup are depicted. The additional delay is caused by the intermediate nodes fetching additional key material in a serial manner. The overall execution time of the protocol is thus influenced by the slowest path which in our setup corresponds to the longest path. Overall, we can thus conclude that the overhead of our end-to-end secure protocol for hybrid networks is mainly influenced by the performance of the key rate provided by the QKD network.

5 Conclusion and Outlook

With Muckle+, we extend the hybrid authenticated key exchange protocol Muckle with signature-based authentication. Thereby, we are able to provide both certificate-based mutual or unilateral authentication depending on the intended use-case. Our implementation and evaluation of the protocol within a small QKD network demonstrates its practical feasibility. With our intended message flow of the protocol, Muckle+ may be integrated in typical scenarios where especially the authenticity of the responder is essential. We however note that there might be other trade-offs in the order and structure the messages if different privacy properties are required for an application, e.g., SIGMA with responder privacy or a protocol with forward privacy [SSL20,RSW21].

Note also that the Muckle+ protocol offers features that are interesting for a wide range of applications in a similar setting as found in many of today’s applications. Indeed, if one considers classical client-server uses on the web, it is expected that one can connect to almost any server on the network without additional configuration. Hence, handling shared state such as the pre-shared keys is not desired due to scalability issues as well as the out-of-band communication. Considering more high-level use-cases that are envisioned

¹¹ We observed no significant differences for the initiator and responder.

¹² See <https://github.com/open-quantum-safe/liboqs/issues/1476> for some background on this issue.

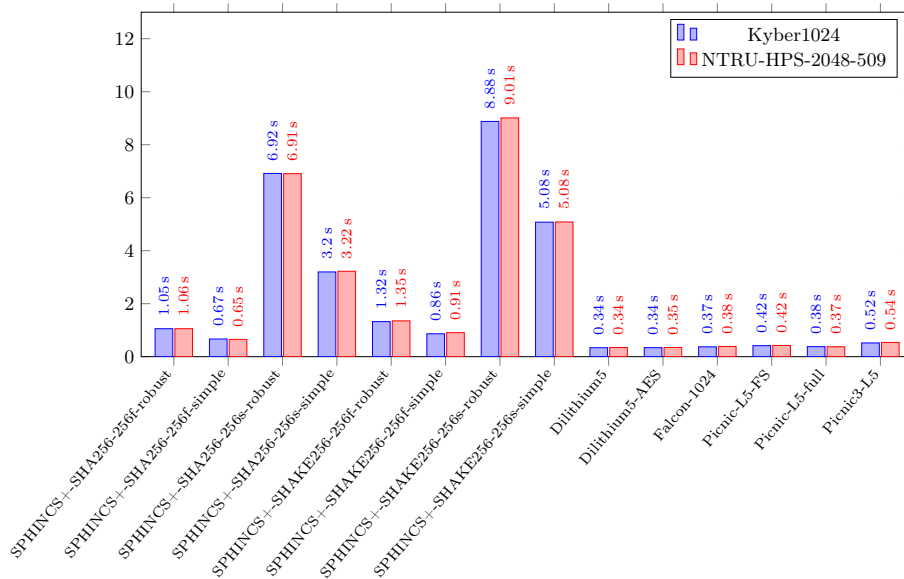


Fig. 4. Execution time of a single Muckle+ stage with mutual authentication. Times are in seconds.

in EuroQCI where network-wide key-management systems will provide QKD keys to security applications, ensuring authenticity with certificate-based mechanisms will provide better scalability especially considering that nowadays process for certificate management can be fully automated [ABC⁺19].

Acknowledgements. The authors want to thank Christian Rechberger and Felix Wissel for insightful discussions, and Florian Kutschera for helping with the setup of the QKD devices. This work received funding from the Austrian Research Promotion Agency (FFG) under grant agreement number FO999886370 (“QKD4GOV”), from the European Defence Industrial Development Programme (EDIDP) under grant agreement number SI2858093 (“DISCRETION”), and from the Digital Europe Program under grant agreement number 101091642 (“QCI-CAT”).

References

- ABB⁺14. Romain Alléaume, Cyril Branciard, Jan Bouda, Thierry Debuisschert, Mehrdad Dianati, Nicolas Gisin, Mark Godfrey, Philippe Grangier, Thomas Länger, Norbert Lütkenhaus, Christian Monyk, Philippe Painchault, Momtchil Peev, Andreas Poppe, Thomas Pornin, John G. Rarity, Renato Renner, Gregoire Ribordy, Michel Riguidel, Louis Salvail, Andrew Shields, Harald Weinfurter, and Anton Zeilinger. Using quantum key distribution for cryptographic purposes: A survey. *Theor. Comput. Sci.*, 560:62–81, 2014. doi:10.1016/j.tcs.2014.09.018.
- ABC⁺19. Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J. Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, Seth D. Schoen, and Brad Warren. Let’s encrypt: An automated certificate authority to encrypt the entire web. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2473–2487. ACM Press, November 2019. doi:10.1145/3319535.3363192.
- BBF⁺19. Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 206–226. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-25510-7_12.
- BCNS15. Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE Computer Society Press, May 2015. doi:10.1109/SP.2015.40.

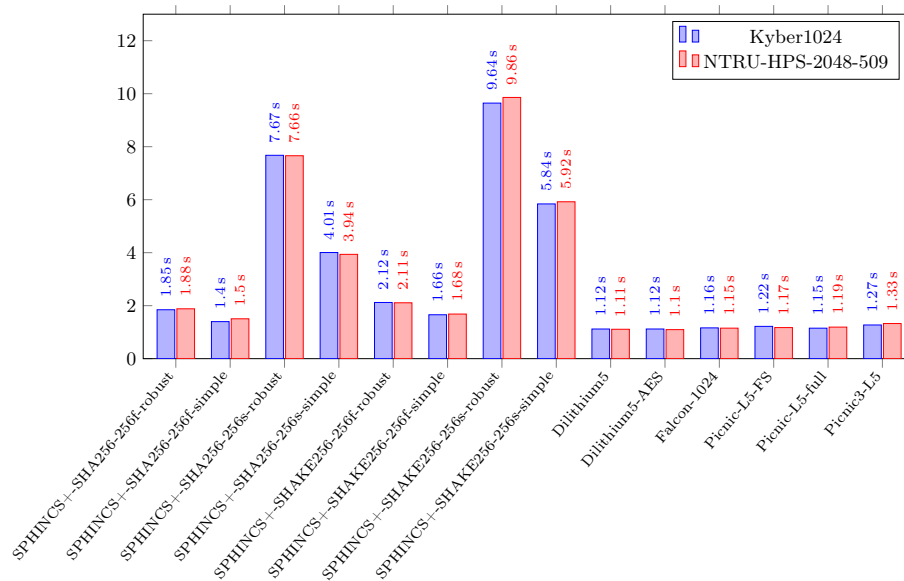


Fig. 5. Execution time of a single Muckle+ stage with mutual authentication in a multi-path setting. Times are in seconds.

- BDH11. Johannes A. Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 117–129. Springer, Heidelberg, November / December 2011. doi:10.1007/978-3-642-25405-5_8.
- BDK⁺21. Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 266–297. Springer, Heidelberg, May 2021. doi:10.1007/978-3-030-75245-3_11.
- BDL⁺11. Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 124–142. Springer, Heidelberg, September / October 2011. doi:10.1007/978-3-642-23951-9_9.
- Ber06. Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer, Heidelberg, April 2006. doi:10.1007/11745853_14.
- Beu22. Ward Beullens. Breaking rainbow takes a weekend on a laptop. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 464–479. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-15979-4_16.
- BFG19. Jacqueline Brendel, Marc Fischlin, and Felix Günther. Breakdown resilience of key exchange protocols: NewHope, TLS 1.3, and hybrids. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part II*, volume 11736 of *LNCS*, pages 521–541. Springer, Heidelberg, September 2019. doi:10.1007/978-3-030-29962-0_25.
- BHK⁺19. Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS⁺ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2129–2146. ACM Press, November 2019. doi:10.1145/3319535.3363229.
- BL15. Mihir Bellare and Anna Lysyanskaya. Symmetric and dual PRFs from standard assumptions: A generic validation of an HMAC assumption. *Cryptology ePrint Archive*, Report 2015/1198, 2015. <https://eprint.iacr.org/2015/1198>.
- BM99. Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_28.

- BMO17. Raphaël Bost, Brice Minaud, and Olga Ohrimenko. Forward and backward private searchable encryption from constrained cryptographic primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1465–1482. ACM Press, October / November 2017. doi:10.1145/3133956.3133980.
- BMP00. Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 156–171. Springer, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_12.
- BPR00. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_11.
- BR95. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: The three party case. In *27th ACM STOC*, pages 57–66. ACM Press, May / June 1995. doi:10.1145/225058.225084.
- BRS23. Sonja Bruckner, Sebastian Ramacher, and Christoph Striecks. Muckle+: End-to-end hybrid authenticated key exchanges. In *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023 (to appear)*, 2023. doi:10.1007/978-3-031-40003-2_22.
- CCCZ23. Liquan Chen, Jing-Qi Chen, Qian-Ye Chen, and Yong-Li Zhao. A quantum key distribution routing scheme for hybrid-trusted QKD network system. *Quantum Inf. Process.*, 22(1):75, 2023. doi:10.1007/s11128-022-03825-x.
- CCG⁺18. Sergiu Costea, Marios O. Choudary, Doru Gucea, Björn Tackmann, and Costin Raiciu. Secure opportunistic multipath key exchange. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 2077–2094. ACM Press, October 2018. doi:10.1145/3243734.3243791.
- CD23. Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 423–447. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30589-4_15.
- CDG⁺17. Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017. doi:10.1145/3133956.3133997.
- CF12. Cas J. F. Cremers and Michele Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS 2012*, volume 7459 of *LNCS*, pages 734–751. Springer, Heidelberg, September 2012. doi:10.1007/978-3-642-33167-1_42.
- CHK03. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, Heidelberg, May 2003. doi:10.1007/3-540-39200-9_16.
- CRSS20. Valerio Cini, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. CCA-secure (puncturable) KEMs from encryption with non-negligible decryption errors. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 159–190. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64837-4_6.
- CSF⁺08. David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and W. Timothy Polk. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. *RFC*, 5280:1–151, 2008. doi:10.17487/RFC5280.
- DCM20. Emma Dauterman, Henry Corrigan-Gibbs, and David Mazières. Safetypin: Encrypted backups with human-memorable secrets. In *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020*, pages 1121–1138. USENIX Association, 2020.
- DDG⁺20. Fynn Dallmeier, Jan P. Drees, Kai Gellert, Tobias Handirk, Tibor Jager, Jonas Klauke, Simon Nachtigall, Timo Renzelmann, and Rudi Wolf. Forward-secure 0-RTT goes live: Implementation and performance analysis in QUIC. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 211–231. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-65411-5_11.
- DFGS21. Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol. *Journal of Cryptology*, 34(4):37, October 2021. doi:10.1007/s00145-021-09384-1.

- DGJ⁺21. David Derler, Kai Gellert, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-rtt key exchange. *J. Cryptol.*, 34(2):13, 2021.
- DGNW20. Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2093–2110. USENIX Association, August 2020.
- DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.
- DHP20. 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 Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 483–502. Springer, Heidelberg, 2020. doi:10.1007/978-3-030-44223-1_26.
- DJSS18. David Derler, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 425–455. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78372-7_14.
- DKL⁺18. David Derler, Stephan Krenn, Thomas Lorünser, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Revisiting proxy re-encryption: Forward secrecy, improved security, and applications. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 219–250. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5_8.
- Don17. Jason A. Donenfeld. WireGuard: Next generation kernel network tunnel. In *NDSS 2017*. The Internet Society, February / March 2017.
- DRSS21. David Derler, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Fine-grained forward secrecy: Allow-list/deny-list encryption and applications. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part II*, volume 12675 of *LNCS*, pages 499–519. Springer, Heidelberg, March 2021. doi:10.1007/978-3-662-64331-0_26.
- DvOW92. Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptogr.*, 2(2):107–125, 1992. doi:10.1007/BF00124891.
- ETS19. ETSI. Quantum key distribution (qkd): Protocol and data format of rest-based key delivery api, 2019. URL: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf.
- FFGV07. Matthias Fitz, Matthew K. Franklin, Juan A. Garay, and S. Harsha Vardhan. Towards optimal and efficient perfectly secure message transmission. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 311–322. Springer, Heidelberg, February 2007. doi:10.1007/978-3-540-70936-7_17.
- GHJL17. Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT key exchange with full forward secrecy. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 519–548. Springer, Heidelberg, April / May 2017. doi:10.1007/978-3-319-56617-7_18.
- Gro21. Jens Groth. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Report 2021/339, 2021. <https://eprint.iacr.org/2021/339>.
- Gün90. Christoph G. Günther. An identity-based key-exchange protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT’89*, volume 434 of *LNCS*, pages 29–37. Springer, Heidelberg, April 1990. doi:10.1007/3-540-46885-4_5.
- HAD⁺22. Bruno Huttner, Romain Alléaume, Eleni Diamanti, Florian Fröwis, Philippe Grangier, Hannes Hübel, Vicente Martin, Andreas Poppe, Joshua A. Slater, Tim Spiller, Wolfgang Tittel, Benoit Tranier, Adrian Wonfor, and Hugo Zbinden. Long-range qkd without trusted nodes is not possible with current technology. *npj Quantum Information*, 8(1):1–5, 2022.
- HBG⁺18. Andreas Hülsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: extended merkle signature scheme. *RFC*, 8391:1–74, 2018. doi:10.17487/RFC8391.
- HKSU20. Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. Generic authenticated key exchange in the quantum random oracle model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 389–422. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45388-6_14.
- HNS⁺21. Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R. Zimmermann. Post-quantum WireGuard. In *2021 IEEE Symposium on Security and Privacy*, pages 304–321. IEEE Computer Society Press, May 2021. doi:10.1109/SP40001.2021.00030.
- IT21. Jana Iyengar and Martin Thomson. QUIC: A udp-based multiplexed and secure transport. *RFC*, 9000:1–151, 2021. doi:10.17487/RFC9000.

- Kau05. Charlie Kaufman. Internet key exchange (ikev2) protocol. *RFC*, 4306:1–99, 2005. doi:10.17487/RFC4306.
- KGSR02. M. V. N. Ashwin Kumar, Pranava R. Goundan, K. Srinathan, and C. Pandu Rangan. On perfectly secure communication over arbitrary networks. In Aleta Ricciardi, editor, *21st ACM PODC*, pages 193–202. ACM, July 2002. doi:10.1145/571825.571858.
- KKW18. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018. doi:10.1145/3243734.3243805.
- KL14. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. URL: <https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269>.
- Kra03. Hugo Krawczyk. SIGMA: The “SIGn-and-MAC” approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 400–425. Springer, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_24.
- KSL⁺19. Krzysztof Kwiatkowski, Nick Sullivan, Adam Langley, Dave Levin, and Alan Mislove. Measuring tls key exchange with post-quantum kem. Workshop Record of the Second PQC Standardization Conference, 2019. <https://csrc.nist.gov/CSRC/media/Events/Second-PQC-Standardization-Conference/documents/accepted-papers/kwiatkowski-measuring-tls.pdf>.
- Lan16. Adam Langley. Cccpq1 results. Blog post, 2016. <https://www.imperialviolet.org/2016/11/28/cccpq1.html>.
- LGM⁺20. Sebastian Lauer, Kai Gellert, Robert Merget, Tobias Handirk, and Jörg Schwenk. TORTT: Non-interactive immediate forward-secret single-pass circuit construction. *PoPETs*, 2020(2):336–357, April 2020. doi:10.2478/popets-2020-0030.
- LKZC07. Jin Li, Kwangjo Kim, Fangguo Zhang, and Xiaofeng Chen. Aggregate proxy signature and verifiably encrypted proxy signature. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 208–217. Springer, Heidelberg, November 2007.
- LRW⁺17. Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan R. Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. The QUIC transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*, pages 183–196. ACM, 2017. doi:10.1145/3098822.3098842.
- Mau93. Ueli M. Maurer. Protocols for secret key agreement by public discussion based on common information. In Ernest F. Brickell, editor, *CRYPTO’92*, volume 740 of *LNCS*, pages 461–470. Springer, Heidelberg, August 1993. doi:10.1007/3-540-48071-4_32.
- MNR⁺20. Miralem Mehic, Marcin Niemiec, Stefan Rass, Jiajun Ma, Momtchil Peev, Alejandro Aguado, Vicente Martín, Stefan Schauer, Andreas Poppe, Christoph Pacher, and Miroslav Voznák. Quantum key distribution: A networking perspective. *ACM Comput. Surv.*, 53(5):96:1–96:41, 2020. doi:10.1145/3402192.
- MSU13. Michele Mosca, Douglas Stebila, and Berkant Ustaoglu. Quantum key distribution in the classical authenticated key exchange framework. In Philippe Gaborit, editor, *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013*, pages 136–154. Springer, Heidelberg, June 2013. doi:10.1007/978-3-642-38616-9_9.
- PAL⁺16. Christoph Pacher, Aysajan Abidin, Thomas Lorünser, Momtchil Peev, Rupert Ursin, Anton Zeilinger, and Jan-Åke Larsson. Attacks on quantum key distribution protocols that employ non-its authentication. *Quantum Inf. Process.*, 15(1):327–362, 2016. doi:10.1007/s11128-015-1160-4.
- PST20. Christian Paquin, Douglas Stebila, and Goutam Tamvada. Benchmarking post-quantum cryptography in TLS. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 72–91. Springer, Heidelberg, 2020. doi:10.1007/978-3-030-44223-1_5.
- Res18. Eric Rescorla. The transport layer security (TLS) protocol version 1.3. *RFC*, 8446:1–160, 2018. doi:10.17487/RFC8446.
- RK11. Stefan Rass and Sandra König. Indirect eavesdropping in quantum networks. In *ICQNM 2011: The Fifth International Conference on Quantum, Nano and Micro Technologies*, 10 2011.
- RKJ⁺21. Leila Rashidi, Daniel Kostecki, Alexander James, Anthony Peterson, Majid Ghaderi, Samuel Jero, Cristina Nita-Rotaru, Hamed Okhravi, and Reihaneh Safavi-Naini. More than a fair share: Network

- data remanence attacks against secret sharing-based schemes. In *NDSS 2021*. The Internet Society, February 2021.
- RS10. Stefan Rass and Peter Schartner. Multipath authentication without shared secrets and with applications in quantum networks. In Hamid R. Arabnia, Kevin Daimi, Michael R. Grimaila, George Markowsky, Selim Aissi, Victor A. Clincy, Leonidas Deligiannidis, Donara Gabrielyan, Gevorg Margarov, Ashu M. G. Solo, Craig Valli, and Patricia A. H. Williams, editors, *Proceedings of the 2010 International Conference on Security & Management, SAM 2010, July 12-15, 2010, Las Vegas Nevada, USA, 2 Volumes*, pages 111–115. CSREA Press, 2010.
- RSS23. Paul Rösler, Daniel Slamanig, and Christoph Striecks. Unique-path identity based encryption with applications to strongly secure messaging. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 3–34. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30589-4_1.
- RSW21. Sebastian Ramacher, Daniel Slamanig, and Andreas Weninger. Privacy-preserving authenticated key exchange: Stronger privacy and generic constructions. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *ESORICS 2021, Part II*, volume 12973 of *LNCS*, pages 676–696. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-88428-4_33.
- SM16. Douglas Stebila and Michele Mosca. Post-quantum key exchange for the internet and the open quantum safe project. In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016*, volume 10532 of *LNCS*, pages 14–37. Springer, Heidelberg, August 2016. doi:10.1007/978-3-319-69453-5_2.
- SS21. Daniel Slamanig and Christoph Striecks. Puncture ’em all: Updatable encryption with no-directional key updates and expiring ciphertexts. Cryptology ePrint Archive, Report 2021/268, 2021. <https://eprint.iacr.org/2021/268>.
- SSL20. Sven Schäge, Jörg Schwenk, and Sebastian Lauer. Privacy-preserving authenticated key exchange and the case of IKEv2. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 567–596. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45388-6_20.
- SSW20. Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1461–1480. ACM Press, November 2020. doi:10.1145/3372297.3423350.
- YLL⁺21. Xiaosong Yu, Xiang Liu, Yuhang Liu, Avishek Nag, Xingyu Zou, Yongli Zhao, and Jie Zhang. Multipath-based quasi-real-time key provisioning in quantum-key-distribution enabled optical networks (qkd-on). *Opt. Express*, 29(14):21225–21239, Jul 2021. doi:10.1364/OE.425562.
- Zha16. Yunlei Zhao. Identity-concealed authenticated encryption and key exchange. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1464–1479. ACM Press, October 2016. doi:10.1145/2976749.2978350.