This work was published at the 28th Australasian Conference on
Information Security and Privacy (ACISP 2023). This is the full version.

Modular Design of KEM-Based Authenticated Key Exchange*

Colin Boyd Bor de Kock ✉ Lise Millerjord

NTNU – Norwegian University of Science and Technology, Trondheim, Norway.
colin.boyd@ntnu.no, bor.dekock@ntnu.no, lise.millerjord@ntnu.no

Abstract

A key encapsulation mechanism (KEM) is a basic building block for key exchange which must be combined with long-term keys in order to achieve authenticated key exchange (AKE). Although several KEM-based AKE protocols have been proposed, KEM-based modular building blocks are not available. We provide a KEM-based authenticator and a KEM-based protocol in the Authenticated Links model (AM), in the terminology of Canetti and Krawczyk (2001). Using these building blocks we achieve a set of generic AKE protocols. By instantiating these with post-quantum secure primitives we are able to propose several new post-quantum secure AKE protocols.

1 Introduction

Authenticated key exchange (AKE) is a fundamental tool for establishing secure communications. An important component in the design of AKE protocols is Diffie–Hellman (DH) key exchange, due to its versatility and potential for providing security properties such as forward secrecy. Today many real-world AKE protocols are based on DH implementations, typically in elliptic curve groups; examples include TLS, IPSec, WireGuard and the generic Noise Framework.

The looming threat of quantum computers has brought about an increasingly pressing need to find post-quantum secure replacements for DH, which itself is well known to be broken by Shor's quantum algorithm for finding discrete logarithms [BL17]. In the absence of many promising candidates for a post-quantum secure direct DH replacement, designs for post-quantum AKE have tended to make use of key encapsulation mechanisms (KEM). This approach aligns well with the research literature where many post-quantum candidate KEMs have been proposed and also with the prominent NIST post-quantum cryptography competition [AAC+22] which requests primitives of only two types, namely

*Boyd and Millerjord are supported by the Research Council of Norway under Project No. 288545. Author list in alphabetical order; see https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf

a KEM [1] or a digital signature. Although DH can be framed as a KEM, DH has special properties which prevent KEMs from being used as a drop-in replacement for DH. For example, DH has the property that two parties can generate their DH shares completely independently; this cannot be achieved in general with KEMs, but rather one party must wait for the other party's input.

Achieving the *authenticated* part of AKE has traditionally been done by applying a digital signature scheme on the messages of a key exchange protocol, but authentication can also be achieved in different ways, which can be advantageous for several reasons: for instance to achieve a speed-up or use less memory, as other works have demonstrated [SSW20].

Although in 2022 the NIST competition for post-quantum secure cryptography (PQ or PQ-crypto) has led to the standardization of several signature schemes, only one KEM was standardized along with an open call for more schemes to be proposed [AAC+22]. To achieve authentication without depending on this one scheme is a desirable property. One of the main motivations for our work is to be flexible in the use of cryptographic primitives so that as the security of post-quantum KEMs or signatures becomes better understood, and as new primitives are designed, it is easy to swap in and out different ones.

1.1 Modular design of AKE

Over the past decade, several key exchange protocols using post-quantum candidate KEMs have been proposed, both authenticated [DAL+17] and unauthenticated [DXL12, Pei15, BCD+16]. Some of these protocols have been proposed based on specific KEM constructions and the security proofs (where available) relate to specific computational assumptions. These are essential constructions for instantiating protocols using abstract primitives, but when using specific constructions as the basis of security for AKE there is a loss of cryptographic agility. Our goal in this work is to design generic AKE protocols where we can be as flexible as possible with regard to choice of specific KEM instantiations and how they are used.

Our protocol designs are based on the modular approach of Bellare, Canetti and Krawczyk [BCK98, HBN06] (hereafter referred to as BCK98) and Canetti and Krawczyk [CK01] (hereafter referred to as CK01). This approach entails defining protocols which are secure in a world which is ideally authenticated and then compiling these protocols with *authenticators* to achieve protocols secure in a world where adversaries completely control the network. A brief introduction to this modular approach is given in Sec. 2.2.

A significant benefit of the modular approach is the ability to "mix-and-match" different components and to use different concrete instances of the same component within one protocol instantiation. This leads to a plethora of different concrete protocols with varying performance characteristics. For example, in our abstract protocol combining our KEM-based ideal-world protocol with our KEM-based authenticator, choosing from

[1] Note that in some documents of the NIST competition [AASA+20], KEM is used as an abbreviation for *key establishment mechanism* but we stick to the traditional name in cryptographic research.

a collection of, say, five different concrete KEMs for each usage, leads to 5^3 different concrete protocols. Not all of these will be of independent interest, but many will have distinctive properties.

We remark that there already exist several protocol designs which are generic in the sense that they can use any specific (secure) instance of different cryptographic primitives such as non-interactive key exchange (NIKE), signatures and/or KEMs [FSXY12, BJS15, JKRS21]. However, such designs do not allow generic mixing of different generic primitives as can be done with the modular approach. For example, the modular approach can be used to replace a digital signature authentication method which a MAC-based method in the case that a pre-shared key is available in a particular application.

While the motivation for this work comes from the goal of achieving post-quantum secure AKE, we note that there is no requirement to use (only) post-quantum secure components in instantiations. Interim solutions combining post-quantum secure ideal KEMs with conventional established primitives for authentication, will lead to more efficient AKEs secure against active attackers in the short-term and post-quantum adversaries into the future. Furthermore, we can provide *hybrid* secure AKEs which remain secure until both conventional and post-quantum secure components are broken. By using signature combiners or KEM combiners [BHMS17, GHP18, BBF+19] we can plug hybrid-secure primitives into any of our generic constructions.

1.2 Contributions

We regard the following as the main contributions of this paper:

1. We develop a new KEM-based authenticator and prove its security as a valid authenticator in the CK01 definition, relying on the established CCA-security definition for KEMs.

2. We frame the well-known method of using an ephemeral KEM as a DH replacement as a protocol in the authenticated-links model (AM) of CK01 and prove its security in that model, relying on the established CPA-security definition for KEMs.

3. We derive efficient and secure generic AKE protocols which can be instantiated with any appropriately secure KEMs and also matched with other primitives such as signatures. Some of these generic protocols are completely new, allowing new instantiations of concrete protocols.

1.3 Related work

In 2017, De Saint Guilhem, Smart and Warinschi [dSW17] presented a generic transformation to convert any two-round forward-secret, but only passively secure, key agreement protocol into a three-round authenticated key agreement protocol. Recognising the value of avoiding signatures for authentication in the post-quantum setting, their transformation makes use of generic CCA-secure public key encryption and a secure MAC. While the approach of De Saint Guilhem et al. has clear parallels with ours, they rely

on encryption rather than the often more efficient notion of KEMs. Moreover, they do not allow mixing of different authentication methods as we do, nor provide KEM-based concrete passively secure protocols. Furthermore, their proofs require a key derivation function modelled as a random oracle. Interestingly, they dismiss the CK01 modular approach stating that it necessarily results in increased number of rounds; below we will explain why this is not the case.

Several recent works show that KEM-based approaches are suitable for replacing signatures in real-world applications. The KEMTLS protocol of Schwabe, Stebila and Wiggers [SSW20] is for instance a complete reworking of the TLS 1.3 handshake without using signatures, showing that this would in theory require only half the bandwidth compared to a classical approach — with additional improvements to be gained if the public keys are exchanged in advance [SSW21]. Some of these theoretical improvements turned out to be less impactful when looking at a real-world implementation [CFS+21].

Using KEM as a building block for AKE is also done in some other purpose-specific works: examples of this include Post-Quantum Noise [ADH+22], FSXY [FSXY12] and Post-Quantum WireGuard [HNS+20]. These are generic in the sense that any suitable KEMs can be used, but they do not allow the flexibility of different authenticators that we obtain. Specifically, they do not provide re-usable and interchangable components for passive security and for authentication.

There exist many formal security models for AKE, amongst which several are incomparable [Cre11] in the sense that any one model is often neither stronger nor weaker than another. The modular approach that we use [CK01] achieves security in the well-established model known widely as the CK-model. This encompasses fundamental security properties of session key indistinguishability against active attackers who can obtain non-target session keys and adaptively corrupt non-target parties. Forward secrecy is also captured. This model can be adapted [Kra05] if other security properties, such as ephemeral key leakage, are desirable.

1.4 Outline

After presenting necessary definitions, Sec. 2 gives an overview of the CK01 modular design methodology, including definitions such as message transmission (MT) and session key indistinguishability (SK security), which we use extensively throughout this work. This also includes an explanation of how to optimise compiled protocols securely through the application of what we call *compressed* authenticators.

In Sec. 3 we define a new KEM-based MT-authenticator and prove it is a valid authenticator as long as the underlying KEM is CCA-secure, and define a new KEM-based AM protocol, Π which we prove SK-secure.

We then apply compressed authenticators to the authentic messages in Π which results in a 3-message protocol Π' secure in the unauthenticated links model (UM), in Sec. 4. By removing repeated message fields from Π' and combining parallel messages in the same direction, we also obtain efficient generic UM-secure protocols in this section.

Finally, in Sec. 5, we compare the new protocols with existing KEM-based protocols from other works, and examine concrete instantiations of our generic protocols by

plugging in concrete KEMs.

2 Background

The main goal of this section is to present the background necessary to understand the modular approach of (Bellare), Canetti and Krawczyk [BCK98, CK01]. This includes our method to optimise, in a rigorous way, protocols obtained through the approach.

2.1 Basic definitions

We use the following standard definitions for KEM, MAC, signatures etc. throughout the paper. These can be found in, for example, the textbook of Katz and Lindell [KL14].

Definition 1. *A key encapsulation mechanism* (KEM) *is a tuple of PPT algorithms* (Gen, Encap, Decap) *such that:*

1. *The key generation algorithm* Gen *takes the security parameter* 1^n *and outputs a public-/private-key pair (sk, pk):* $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$.
2. *The encapsulation algorithm* Encap *takes as input a public key pk. It outputs a ciphertext c and a key* $k \in \{0,1\}^{l(n)}$*:* $(c, k) \leftarrow \mathsf{Encap}(pk)$.
3. *The deterministic decapsulation algorithm* Decap *takes as input a private key sk and a ciphertext c, and outputs a key k or the special symbol* $\perp$ *denoting failure:* $k \leftarrow \mathsf{Decap}(sk, c)$.

We require correctness *from the* KEM*: If* $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$ *and* $(c, k) \leftarrow \mathsf{Encap}_{pk}(1^n)$*, and* $k' \leftarrow \mathsf{Decap}_{sk}(c)$*, then* $k' = k$ *except with negligible probability.*

Furthermore we show the CPA (resp. CCA) indistinguishability experiment(s) for KEMs.

Definition 2. *The* CPA $\boxed{\textit{resp. CCA}}$ *indistinguishability experiment proceeds as follows:*

1. *The key generation algorithm is run:* $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. *The encapsulation algorithm is run:* $(c, k) \leftarrow \mathsf{Encap}(pk)$*, with* $k \in \{0,1\}^n$.
3. *A uniform bit* $b \in \{0,1\}$ *is chosen. If* $b = 0$*, set* $k' = k$*. Otherwise, if* $b = 1$*, choose a uniform* $k' \in \{0,1\}^n$.
4. *The experiments outputs* (pk, c, k') *to* $\mathcal{A}$.

> $\mathcal{A}$ *is also given access to a decapsulation oracle,* $\mathsf{Decap}_{sk}(\cdot)$*, but cannot query the decapsulation oracle on the ciphertext c.*

5. *$\mathcal{A}$ outputs a bit b'. If $b' = b$, $\mathcal{A}$ wins and the experiment outputs 1. Otherwise, $\mathcal{A}$ loses and the experiment outputs 0.*

The advantage of the adversary $\mathcal{A}$ in the CPA $\boxed{\text{CCA}}$ *experiment is defined to be:*

$$\mathsf{Adv}^{\mathsf{CPA}\boxed{\mathsf{CCA}}}_{\mathsf{KEM}}(\mathcal{A}) = 2 \cdot \left|\Pr\left[b' = b\right] - 1/2\right|.$$

Definition 3. *A* message authentication code *or* MAC *consists of three probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{MacVer})$ *such that:*

1. *The* key-generation algorithm Gen *takes as input the security parameter 1^n and outputs a key k with $|k| \geq n$.*

2. *The* tag-generation algorithm Mac *takes as input a key k and a message $m \in \{0,1\}^*$, and outputs a tag t. Since this algorithm may be randomized, we write this as $t \leftarrow \mathsf{Mac}_k(m)$.*

3. *The deterministic* verification algorithm MacVer *takes as input a key k, a message m and a tag t. It outputs a bit b, with $b = 1$ meaning* valid *and $b = 0$ meaning* invalid. *We write this as $b := \mathsf{MacVer}_k(m, t)$.*

It is required that for every n, every key k and output by $\mathsf{Gen}(1^n)$ and every $m \in \{0,1\}$, it holds that $\mathsf{MacVer}_k(m, \mathsf{Mac}(m)) = 1$.

Definition 4. *The existential unforgeability under chosen message attacks* (EUF-CMA) *experiment for* $\mathsf{MAC}(\mathsf{Gen}, \mathsf{Mac}, \mathsf{MacVer})$ *proceeds as follows:*

1. *A key is generated: $k \leftarrow \mathsf{Gen}(1^n)$.*

2. *The adversary $\mathcal{A}$ gets oracle access to $\mathsf{Mac}_k(\cdot)$. Let Q be the set of all queries $\mathcal{A}$ made to the oracle. The adversary eventually outputs (m, t).*

3. *$\mathcal{A}$ wins if and only if*

   (a) $\mathsf{MacVer}_k(m, t) = 1$, and
   (b) $m \notin Q$.

   In that case the experiment outputs 1. Otherwise, the experiment outputs 0.

The advantage of the adversary $\mathcal{A}$ in the EUF-CMA *experiment is defined to be:*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{A}) = \Pr\left[\mathsf{G}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{A}) = 1\right].$$

Definition 5. *A* (digital) signature scheme *is a tuple of three PTT-algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{SigVer})$ *such that:*

1. *The* key generation algorithm Gen *takes the security parameter 1^n and outputs a public-/private-key pair (sk, pk): $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$.*

2. *The* signing algorithm Sign *takes as input a private key sk and a message m from some message space (that may depend on pk). It outputs the signature* σ *and we write this as* $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$.

3. *The deterministic* verification algorithm SigVer *takes as input a public key pk, a message m and a signature* σ. *It outputs a bit b, with* $b = 1$ *meaning* valid *and* $b = 0$ *meaning* invalid. *We write this as* $b := \mathsf{SigVer}_{pk}(m, \sigma)$.

We require correctness *from the scheme: If* $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$ *then, except with negligible probability,* $\mathsf{SigVer}_{pk}(m, \mathsf{Sign}_{sk}(m)) = 1$.

For brevity we often denote the key generation algorithms from the various primitives as $\mathsf{Gen}()$, omitting the security parameter.

2.2 Canetti–Krawczyk modular design

The *modular approach*, arising originally in a 1998 paper of Bellare, Canetti and Krawczyk [BCK98], is to first define protocols secure against a limited adversary which can then be promoted to protocols secure against a realistic adversary using a generic compiler. In the *authenticated links model* (AM) the adversary is not permitted to fabricate messages, but can otherwise control the network and deliver messages out of order or to different parties from those intended. Compilers, or *authenticators*, can be applied to messages in an AM protocol to obtain protocols in the *unauthenticated links model* (UM) where the adversary can alter or fabricate messages limited only by the available computational power.

In both the UM and AM, adversaries control the execution of protocols by initiating parties and then invoking parties with available queries, including with message inputs. (In Sec. 2.4 we describe the available adversarial queries.) Parties respond to input messages by following the protocol definition and to other queries as defined by the query. Each party computes *local output* which is available to the adversary. The local output includes protocol decisions, such as whether a message is accepted (see Sec. 2.4 for details).

Bellare et al. [BCK98] provide a theorem showing that a secure protocol, Π_{AM}, in the AM maps to a secure protocol in the UM, Π_{UM}, if the mapping is defined by a valid authenticator. An authenticator is valid if an observer, or distinguisher, is unable to distinguish between the world where an adversary $\mathcal{A}$ is interacting with the Π_{AM} and the world where an adversary $\mathcal{U}$ is interacting with the protocol Π_{UM}. This is captured in the notion of protocol emulation in Definition 7.

Definition 6. *The* AM-UM *distinguishing experiment,* $\mathsf{G}^{\mathsf{AM\text{-}UM\text{-}dist}}_{\Pi_{\mathsf{AM}}\text{-}\Pi_{\mathsf{UM}}}(\mathcal{D})$ *proceeds as follows:*

1. *A uniform bit* $b \in \{0, 1\}$ *is chosen. If* $b = 0$, $\mathcal{D}$ *will interact with* $\mathcal{A}$ *and the* AM *protocol* Π_{AM}. *Otherwise, if* $b = 1$, $\mathcal{D}$ *will interact with* $\mathcal{U}$ *and the* UM *protocol* Π_{UM}.

2. *To conclude the experiment,* $\mathcal{D}$ *will halt and output* b'.

3. *The experiment will output* 1 *if and only if* $b = b'$.

We define the advantage of the distinguisher $\mathcal{D}$ *to be*

$$\mathsf{Adv}^{\mathsf{AM\text{-}UM\text{-}dist}}_{\Pi_{\mathsf{AM}}\text{-}\Pi_{\mathsf{UM}}}(\mathcal{D}) = 2 \cdot \left| \Pr\left[\mathsf{G}^{\mathsf{AM\text{-}UM\text{-}dist}}_{\Pi_{\mathsf{AM}}\text{-}\Pi_{\mathsf{UM}}}(\mathcal{D}) = 1\right] - \frac{1}{2} \right|.$$

Definition 7. *Let* Π_{UM} *and* Π_{AM} *be protocols in the* UM *and* AM *models respectively. We say that* Π_{UM} ϵ*-emulates* Π_{AM} *in unauthenticated networks if for any* UM*-adversary* $\mathcal{U}$ *interacting with* Π_{UM}*, there exists an* AM*-adversary* $\mathcal{A}$ *interacting with* Π_{AM} *such that for any distinguisher* $\mathcal{D}$ *playing the* AM-UM *distinguishing game,*

$$\mathsf{Adv}^{\mathsf{AM\text{-}UM\text{-}dist}}_{\Pi_{\mathsf{AM}}\text{-}\Pi_{\mathsf{UM}}}(\mathcal{D}) \leq \epsilon.$$

An authenticator is a specific type of *protocol compiler* transforming one protocol into another. The modularity of the approach relies on the observation that an authenticator will actually preserve protocol security as we will see in Sec. 2.4.

Definition 8 ([BCK98])**.** *An authenticator is a compiler,* C*, that takes an* AM *protocol* Π_{AM} *as input and outputs a* UM *protocol* Π_{UM}*, such that* Π_{UM} *emulates* Π_{AM}*.*

2.3 MT-authenticators

Defining an authenticator for any protocol, regardless of the number of messages, seems at first a difficult problem. To deal with this, BCK98 [BCK98] define a simpler notion of an MT-authenticator, designed to authenticate a single arbitrary message. They also showed that repeated use of a valid MT-authenticator is a valid authenticator, so that protocol messages can be treated separately.

A bit more formally we define MT as a *message transmission* protocol in authenticated networks that works as follows: when P_i is activated with (P_j, m), party P_i sends the message $(\mathsf{P}_i, \mathsf{P}_j, m)$ to party P_j and outputs "P_i sent m to P_j". Upon receiving $(\mathsf{P}_i, \mathsf{P}_j, m)$, P_j outputs "P_j received m from P_i". Note that the quoted outputs are local outputs of the parties and are critical in proving proper emulation; however, when we later show compiled protocols we omit mention of these local outputs.

An MT-authenticator, λ, is a protocol that emulates MT in unauthenticated networks. Given a sequence of MT-authenticators, $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_t)$, the derived protocol compiler, C_Λ, uses the next MT-authenticator to authenticate the next message. More precisely, given a protocol Π in the AM with t messages, $m_1, m_2, \ldots, m_t$ the protocol $\Pi' = C_\Lambda(\Pi)$ in the UM is defined as follows. For each message, m_k, sent in Π, λ_k is run to send the same message from the same initiator to the same recipient. Whenever a party, P_j, outputs "P_j received m from P_i" in λ_k, then Π is activated at P_j with message m_k from P_i. If Λ is a sequence of t MT-authenticators then C_Λ is an authenticator. We restate this in Thm. 1 and give a sketch of the proof, which is given in full in [BCK98].

Theorem 1 ([BCK98])**.** *Let $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_t)$ be a sequence of t MT-authenticators so that each λ_k ϵ-emulates MT. Then the compiler, C_Λ, will be an authenticator such that for any protocol Π in the AM, $C_\Lambda(\Pi)$ $(t \cdot \epsilon)$-emulates Π in the UM.*

Proof. Let Π be an AM protocol. Let $\mathcal{U}$ be a UM-adversary interacting with $C_\Lambda(\Pi)$. $\mathcal{A}$ runs $\mathcal{U}$ on a simulated interaction. Action requests from $\mathcal{U}$ to parties in the UM can be mimiced by $\mathcal{A}$ in the AM and $\mathcal{A}$ relays its results back to $\mathcal{U}$. The only problem with the simulation could occur in the case that $\mathcal{U}$ specifies that a message is received by some party P_j from some party P_i in the UM, but that message is not in the set of messages waiting for delivery in the AM. But this can happen with probability bounded by ϵ. Such an event could occur for any of the t messages and so the probability that the simulation is correct is at least $(1-\epsilon)^t \geq 1 - t \cdot \epsilon$. Finally, any observer will be able to distinguish between the run of Π in the AM and $C_\Lambda(\Pi)$ in the UM with advantage at most $\epsilon' = t \cdot \epsilon$. □

Note that although we have assumed that each MT-authenticator has the same security level ϵ, the theorem is still true if the MT-authenticators have different security levels $\epsilon_1, \epsilon_2, \ldots, \epsilon_t$ and we take $\epsilon = \max_k(\epsilon_k)$. In the cases we are interested in, we will always have $t = 2$.

An example of an MT-authenticator is the encryption-based authenticator [BCK98] shown in Fig. 1, where N_B denotes a nonce and (e_A, d_A) an encryption-decryption keypair. This is a valid MT-authenticator as long as the public key encryption used is CCA-secure and the MAC is secure. The protocol can be optimized in various ways, as we will show later.

Encryption-based MT-authenticator [BCK98]

Alice		**Bob**
"Alice sent m to Bob"	$\xrightarrow{\quad m \quad}$	$\mathsf{N_B} \xleftarrow{\$} \{0,1\}^n$
$\mathsf{N'_B} \leftarrow \mathsf{Decrypt}_{d_\mathsf{A}}(c)$	$\xleftarrow{\quad m, c = \mathsf{Encrypt}_{e_\mathsf{A}}(\mathsf{N_B}) \quad}$	
	$\xrightarrow{\quad m, \tau = \mathsf{Mac}_{\mathsf{N'_B}}(m, \mathsf{B}) \quad}$	If $\mathsf{MacVer}_{\mathsf{N_B}}(\tau, m, \mathsf{B}) = 1$
		"Bob received m from Alice"

Figure 1: Bob authenticates message m from Alice.

There exist several other MT-authenticators defined in the literature including signature-based [BCK98], MAC-based [CK01] and password-based [HBN06]. We show, in compressed form, the signature-based MT-authenticator later (Fig. 9). The modular approach allows combination of any MT-authenticator together with any AM-protocol, resulting in automatically secure UM protocols. Therefore adding any new building

block, either an MT-authenticator or an AM-protocol, results in several new protocols of potential interest.

2.4 SK-security

SK-security is the AKE security notion of CK01 [CK01], capturing session key indistinguishability and correctness of the protocol. To define this notion we need to state the capabilities of the adversary and the indistinguishability experiment. Each protocol run at a party A is associated with a session identifier s. In the AM the value s is an input at the start of a run to the initiator party. Later we will see that session identifiers can be replaced by protocol messages as long as parties can verify that no incomplete sessions between the same parties have the same session identifier. The state of a session consists of the following information:

- status – whether or not the session is complete, aborted, or still in progress;
- any ephemeral key material needed to complete the protocol;
- the session key, sk, if the protocol is completed and has not expired.

The global state of a party may include long-term authentication keys $pk_{\mathsf{A}}, sk_{\mathsf{A}}$. As in most AKE models, we do not explicitly model distribution of long-term keys. Furthermore, we assume that long-term public keys are immediately available to any party that needs them. This may be too strong an assumption in some real-world protocols, such as TLS, and we remark further on this issue when we examine concrete protocols in Sec. 5.

Definition 9 (Matching sessions [CK01])**.** *The two sessions* $(\mathsf{A}, \mathsf{B}, s, role)$ *and* $(\mathsf{A}', \mathsf{B}', s', role')$ *are* matching *if* $\mathsf{A} = \mathsf{B}'$*,* $\mathsf{B} = \mathsf{A}'$ *and* $s = s'$*.*

The adversary may issue the following queries, subject to certain restrictions we will see later.

- NewSession$(\mathsf{A}, \mathsf{B}, s, r)$: the adversary issues the NewSession query to party A, specifying its intended partner B, the session identifier[2] s, and the role r (initiator or responder) of A in the session. A will follow the protocol definition and may return an output message intended for B.
- Send$(\mathsf{A}, \mathsf{B}, m)$: represents activation of A by an incoming message m (possibly including a session identifier) from party B. A will follow the protocol and may reject, accept, or return an output message intended for B.
- Corrupt(A): the adversary learns the whole state of A including any long-term keys. The corruption event is recorded in the local output of A. Subsequently A can never be activated but the adversary can take the role of A in the protocol.

[2]We remark that instantiation of session identifiers differs between the models. In UM, s can be blank as the session identifier need not be determined by the adversary.

- $\mathsf{RevealKey}(\mathsf{A}, \mathsf{B}, s)$: the adversary learns the session key accepted in the session s by A with partner B, if it exists. The reveal event is recorded in the local output of A.
- $\mathsf{RevealState}(\mathsf{A}, \mathsf{B}, s)$: the adversary learns the state information associated with session s at A, such as ephemeral keys. The reveal state event is recorded in the local output of A.
- $\mathsf{Expire}(\mathsf{A}, \mathsf{B}, s)$: if there is a completed session s at A with B then any session key associated with that session is deleted from the memory of A. The Expire event is recorded in the local output of A.
- $\mathsf{Test}(\mathsf{A}, \mathsf{B}, s)$: this query can be asked only once and can only be made to a completed session s at A with partner B. Furthermore there cannot have been any of the following queries made: $\mathsf{RevealKey}(\mathsf{A}, \mathsf{B}, s)$ or $\mathsf{RevealState}(\mathsf{A}, \mathsf{B}, s)$ or $\mathsf{Corrupt}(\mathsf{A})$ or $\mathsf{Corrupt}(\mathsf{B})$. If the bit b specified by the challenger is $b = 1$ then the session key is returned. Otherwise $b = 0$ and a random key from the keyspace are returned.

Now we are in a position to define the SK-security experiment.

Definition 10. *The key indistinguishability experiment,* $\mathsf{G}_{\Pi}^{\mathsf{Key\text{-}Ind}}(\mathcal{A})$ *is defined as follows:*

1. *The challenger chooses a random bit* b *needed to define the* Test *query response.*
2. *The challenger initialises* n *parties and any long-term keys.*
3. $\mathcal{A}$ *may issue queries as defined above.*
4. *Eventually* $\mathcal{A}$ *halts and outputs a bit* b' *to indicate its guess for* b*, based on the response to the* Test *query. The experiment outputs 1 if and only if* $b' = b$*.*

Definition 11. *A key exchange protocol* Π *is* $\epsilon - \mathsf{SK}$*-secure if the following holds for any adversary* $\mathcal{A}$*:*

1. *two* honest *parties (i.e. uncorrupted parties who faithfully execute the protocol instructions) completing matching sessions of the protocol* Π *will output the same key, except with negligible probability, and*
2. *the advantage of the adversary* $\mathcal{U}$ *in the key indistinguishability experiment is:*

$$\mathsf{Adv}_{\Pi}^{\mathsf{Key\text{-}Ind}}(\mathcal{A}) = 2 \cdot \left| \Pr\left[b' = b\right] - 1/2 \right| \leq \epsilon.$$

The final step needed to bring the modular approach together is to show that emulation preserves SK-security. This was proven in CK01 and we re-state and re-prove it as Thm. 2 including concrete bounds. Note that using emulation of an ideal key exchange process as a definition of security, the original idea of BCK98, results in too strong a definition to allow some well-known protocols to be proven secure [CK01, Appendix A].

Theorem 2 ([CK01])**.** *Let* Π *be an* $\epsilon-\mathsf{SK}$*-secure protocol in the* AM *with* t *messages. Let* C_Λ *be the compiler based on* MT*-authenticators* $\lambda_1, \lambda_2, \ldots, \lambda_t$ *such that for any protocol* Π *in the* AM*,* $C_\Lambda(\Pi)$ α*-emulates* Π *in the* UM*. Then protocol* $\Pi' = C_\Lambda(\Pi)$ *is an* $\epsilon' - \mathsf{SK}$*-secure protocol in the* UM *with* $\epsilon' = \epsilon + \alpha$*.*

Proof. Assume to the contrary that there exists a UM adversary $\mathcal{U}$ that has advantage ϵ' in the UM. Using $\mathcal{U}$, we build an AM adversary, $\mathcal{A}$, playing the game of Defn. 10. When $\mathcal{A}$ receives its setup information consisting of system parameters and public keys from its challenger, $\mathcal{A}$ sends the same information to $\mathcal{U}$. Then $\mathcal{A}$ invokes $\mathcal{U}$ and mimics its behaviour in the AM, using its challenger to respond to the action requests when any party is exposed.

When $\mathcal{U}$ sends a message to a party P_j from a party P_i in the UM, $\mathcal{A}$ sends the same message between the same parties in the AM. The emulation will be perfect unless $\mathcal{U}$ successfully sends a message m to some party P_j from P_i but m was never sent by P_i. In this case we will say that $\mathcal{U}$ made a forgery and we let forge be the event that a forgery happens at any time during the run of $\mathcal{U}$.

If forge occurs then $\mathcal{A}$ will abort the simulation and return a random bit to its challenger. Note that this also defines a distinguisher $\mathcal{D}$ which will always win in the case that forge occurs. If forge does not occur then at some point $\mathcal{U}$ will ask its Test query for a session s. $\mathcal{A}$ then announces session s for its own Test query in the AM, receives a real or random key, and returns it to $\mathcal{U}$. Eventually $\mathcal{U}$ will halt and output its bit which $\mathcal{A}$ copies as its response. In this case, $\mathcal{A}$ wins whenever $\mathcal{U}$ wins.

$$\begin{aligned}
\Pr[\mathcal{A}\text{ wins}] &= \Pr[\mathcal{A}\text{ wins}|\mathsf{forge}] \cdot \Pr[\mathsf{forge}] + \Pr[\mathcal{A}\text{ wins}|\neg\mathsf{forge}] \cdot \Pr[\neg\mathsf{forge}] \\
&= 1/2 \cdot \Pr[\mathsf{forge}] + \Pr[\mathcal{B}\text{ wins}] \cdot (1 - \Pr[\mathsf{forge}]) \\
&\geq \Pr[\mathcal{B}\text{ wins}] - 1/2 \cdot \Pr[\mathsf{forge}]
\end{aligned}$$

We also implicitly defined a distinguisher, $\mathcal{D}$, which wins when forge occurs or wins with probability at least $1/2$ when forge does not occur: $\Pr[\mathcal{D}\text{ wins}] \geq \Pr[\mathsf{forge}]/2 + 1/2$. Putting this together we get:

$$\mathsf{Adv}^{\mathsf{Key\text{-}Ind}}_{\Pi'}(\mathcal{U}) \leq \mathsf{Adv}^{\mathsf{Key\text{-}Ind}}_{\Pi}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{AM\text{-}UM\text{-}dist}}_{\Pi-\Pi'}(\mathcal{D}).$$

□

2.5 Optimising the UM protocol

Simple application of an MT-authenticator to each message of an SK-secure AM protocol results in an SK-secure UM protocol as proven in Thm. 2. However, such a protocol is far from optimal. The most obvious drawback is that a two-message protocol, such as Diffie–Hellman, compiles to a six-message protocol. The obvious way to optimise such a protocol is to "piggyback" messages going in the same direction. The resulting protocol may be secure, but formally this process may break the security proof because it may alter the order of the local output of the parties, allowing trivial distinguishability outputs of the AM protocol from the outputs of the compiled UM protocol [HBN06].

Because of such issues, the modular approach of CK01 has been criticised [dSW17] for not achieving efficient protocols. There is some truth in such criticisms — for example, when using signature- or encryption-based authenticators it is not possible to achieve secure 2-message AKE protocols which are often seen in the literature. Fortunately, rigorous optimisations are not difficult to achieve, typically resulting in 3-message protocols as efficient as real-world protocols. Indeed, 3-message AKE protocols are necessary in any case to achieve desirable security properties such as mutual entity authentication or key confirmation.

Hitchcock et al. [HBN06] designed a general technique for altering message ordering in a security-preserving way. This involved defining an intermediate model between the AM and UM, which they call the *hybrid model*. Rather than use this more comprehensive approach, here we apply simple techniques to allow optimisation of the number of messages and re-use message components as session identifiers. Consequently, the drawbacks of practical application of authenticators are removed resulting in generic protocols as efficient as standalone protocols.

Compressed authenticators. The first step is to compress the authenticator to remove redundant elements. Notice that use of the authenticator in Fig. 1 expands each message m from the AM into three messages in the UM. However, sending m in all three messages is not actually needed (to achieve security), so we can simplify the encryption-based authenticator into a compressed version shown in Fig. 2. It is not hard to see [BCK98, HBN06] that removal of the repeated m fields does not affect the security of the MT-authenticator. Depending on the application scenario, the version in Fig. 1 may remain appropriate. The version in Fig. 2 is useful in a situation where Bob knows that some message, as yet unknown, will be authentically received from Alice; this case typically occurs in AKE protocols. Later we will see that to apply optimisation it is important that the first message in Fig. 2 is independent of the message to be authenticated, so that it can be generated and sent early in the protocol.

Compressed Encryption-based MT-authenticator

Alice		**Bob**
$\mathsf{N}'_\mathsf{B} \leftarrow \mathsf{Decrypt}_{d_\mathsf{A}}(c)$	$\xleftarrow{c = \mathsf{Encrypt}_{e_\mathsf{A}}(\mathsf{N}_\mathsf{B})}$	$\mathsf{N}_\mathsf{B} \xleftarrow{\$} \{0,1\}^n$
"Alice sent m to Bob"	$\xrightarrow{m, \tau = \mathsf{Mac}_{\mathsf{N}'_\mathsf{B}}(m, \mathsf{B})}$	
		If $\mathsf{MacVer}_{\mathsf{N}_\mathsf{B}}(\tau, m, \mathsf{B}) = 1$ "Bob received m from Alice"

Figure 2: Compressed version of MT-authenticator in Fig. 1.

Session identifiers. In the original formulation of CK01, session identifiers are sent in each protocol run in the AM. These must be unique for each active protocol run between the same parties, but it is not defined how they should be obtained in practice. Although the only property required of session identifiers is uniqueness, a natural way of obtaining them is to use random values chosen by each party; in that case the probability that session identifiers are not unique is negligible. In practice it may not be a burden for each party to ensure that there are no other incomplete sessions with the same identifier so that uniqueness is unconditionally guaranteed.

We assume that higher communication layers will provide a mechanism to ensure that messages get delivered to the correct session. They can also be explicitly added to the protocol messages if desired.

3 KEM-based building blocks

This section defines and proves security for the basic KEM-based MT-authenticator and AM protocol, which will be brought together in Sec. 4 as components in defining generic efficient KEM-based AKE.

3.1 KEM-based MT-authenticator

Fig. 3 illustrates our KEM-based MT-authenticator. The construction is closely related to the encryption-based authenticator of BCK98.

KEM-based MT-authenticator

Alice		**Bob**
"Alice sent m to Bob"	$\xrightarrow{\quad m \quad}$	$(c, k) \leftarrow \mathsf{Encap}(pk_{\mathsf{A}})$
	$\xleftarrow{\quad m, c \quad}$	
$k \leftarrow \mathsf{Decap}(sk_{\mathsf{A}}, c)$	$\xrightarrow{\quad m, \tau = \mathsf{MAC}_k(m, \mathsf{B}) \quad}$	If $\mathsf{MacVer}_k(\tau, m, \mathsf{B}) = 1$
		"Bob received m from Alice"

Figure 3: KEM-based MT-authenticator, λ_{KEM}: Bob authenticates m from Alice.

Next we give a theorem that λ_{KEM} is secure, meaning that it emulates MT in unauthenticated networks, as long as the KEM used achieves CCA security.

Theorem 3. *The* KEM*-based* MT*-authenticator,* λ_{KEM}*, in Fig. 3, when instantiated with a* CCA*-secure* KEM *and a secure* MAC *scheme,* ϵ*-emulates protocol* MT *in unauthenticated networks such that* $\epsilon \leq l \cdot (\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{D}) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{F}))$ *where* $l = n_P^2 \times n_M$*,* n_P *is the*

number of parties that run the protocol and n_M is the maximum number of challenge messages that can be sent by any party.

The proof of Thm. 3 follows closely the proof strategy of Bellare et al. [BCK98, Proposition 5]. We have included some extra details into our version of their proof which we hope may be useful to the reader.

Proof. Let $\mathcal{U}$ be a UM-adversary that interacts with λ_{KEM}. We construct an AM-adversary $\mathcal{A}$ such that for any distinguisher $\mathcal{D}$ observing, $\mathsf{View}^{\mathsf{MT}}_{\mathcal{A}}(\mathcal{D}) \stackrel{s}{=} \mathsf{View}^{\lambda_{\mathsf{KEM}}}_{\mathcal{U}}(\mathcal{D})$.

First note that $\mathcal{A}$ can simply copy all of the outputs of $\mathcal{U}$ unless the following event bad happens.

> bad: $\mathcal{U}$ outputs "Q received m from P" for some parties P and Q, but there was no previous output "P sent m to Q".

If we assume that bad happens with probability ϵ then for any distinguisher $\mathcal{D}$ we must have $\mathsf{Adv}(\mathcal{D}) \leq \epsilon$. Thus from now on we focus on bounding ϵ. We construct an experiment involving multiple adversaries as follows and illustrated in Fig. 4. We start by describing the purpose and connections between the different entities. Afterwards we give details of the security game.

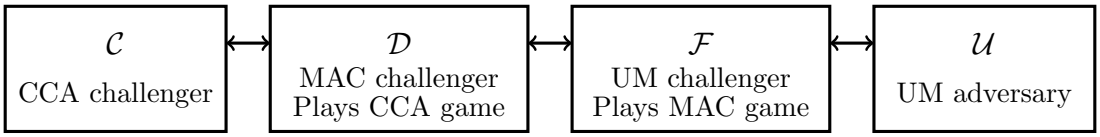

Figure 4: Overview of the experiment in the proof of Thm. 3.

- $\mathcal{U}$ is the UM adversary running λ_{KEM} in such a way that event bad will occur with probability ϵ. When bad happens $\mathcal{U}$ will produce a valid MAC for some session without knowing the decapsulation key of the party P which should be producing the MAC.
- $\mathcal{F}$ is going to interact with $\mathcal{U}$ in order to simulate all of the parties in the run of $\mathcal{U}$. We will see that $\mathcal{F}$ can make the simulation perfect in case bad happens only in a chosen target session. $\mathcal{F}$ also is playing a forging game against the MAC scheme. Since $\mathcal{F}$ also receives a ciphertext from its challenger it may not be playing the usual EUF-CMA game, but in cases where that ciphertext is independent of the MAC key the game $\mathcal{F}$ plays is equivalent to the EUF-CMA game.
- $\mathcal{D}$ is acting as the challenger for $\mathcal{F}$ in the EUF-CMA game, providing a MAC oracle. $\mathcal{D}$ is also playing the IND-CCA game from Def. 2 against the KEM. $\mathcal{D}$ will provide its challenge encapsulation to $\mathcal{F}$.

- $\mathcal{C}$ is the IND-CCA challenger, providing the challenge encapsulation and the decapsulation oracle.

The experiment proceeds as follows.

1. $\mathcal{D}$ starts the CCA indistinguishability experiment with $\mathcal{C}$. The first 4 steps in Def. 2 are run between $\mathcal{D}$ and $\mathcal{C}$ so that $\mathcal{D}$ gets (pk, c^*, k') where k' is either encapsulated in c^* $(b = 0)$ or is a key chosen randomly in the range of Encap $(b = 1)$.

2. $\mathcal{D}$ starts $\mathcal{F}$ by passing $\mathcal{F}$ the encapsulation c^*. From now on $\mathcal{D}$ will answer any MAC oracle queries by using the key k' to compute the correct MAC tag and returning it to $\mathcal{F}$. The goal of $\mathcal{F}$ is to produce a forgery (m^*, t) which is a valid MAC (for key k') which was never returned by $\mathcal{D}$ following a MAC oracle query from $\mathcal{F}$.

   $\mathcal{F}$ is also provided with a decapsulation oracle for the key encapsulated in c^*. $\mathcal{F}$ will need this oracle in order to properly simulate the parties for $\mathcal{U}$. This oracle is available to $\mathcal{D}$ through the game that $\mathcal{D}$ is playing with $\mathcal{C}$ so $\mathcal{D}$ can also simulate the oracle for $\mathcal{F}$.

   Note that there are two cases.

   (a) The encapsulation c contains key k', the key for the MAC that $\mathcal{F}$ is attempting to forge. In this case $\mathcal{F}$ has received some additional help in finding a forgery by knowing c and having the decapsulation oracle for k'. Thus $\mathcal{F}$ is an *aided oracle* [BCK98] and even though a correct forgery is produced and accepted by $\mathcal{D}$, this does not win the MAC security game in the $\mathsf{EUF\text{-}CMA}$ sense.

   (b) The encapsulation c contains a random key, independent of k'. In this case $\mathcal{F}$ does not receive any useful help in finding a forgery for k'. (Note that $\mathcal{F}$ could easily simulate the "help" itself in this case.) Thus a forgery wins the $\mathsf{EUF\text{-}CMA}$ MAC game in Def. 4.

3. $\mathcal{F}$ starts $\mathcal{U}$ and chooses two random parties P^* and P and a randomly selected session s at P. This is the session where $\mathcal{F}$ guesses that event bad will happen. Note that the probability that bad happens in session s is at least ϵ/l where $l = n_P^2 \times n_M$ where n_P is the number of parties and n_M is the maximum number of challenge messages that can be sent by party P.

   $\mathcal{F}$ chooses public key pairs for all parties except P^* and assigns pk to party P^*. $\mathcal{F}$ can now simulate all parties in the run of $\mathcal{U}$ as long as $\mathcal{U}$ does not choose to corrupt P^* or asks for a MAC for the message used in the target session s with partner P; however if either of these two things happen then $\mathcal{F}$ will abort and $\mathcal{D}$ will return a random bit as its guess for b'.

   When simulating a session involving party P^*, but different from s, $\mathcal{F}$ has to make use of the decapsulation oracle from $\mathcal{D}$ since it does not have the private decapsulation key of P^*. Then $\mathcal{F}$ can obtain the correct MAC key and provide the

correct third message for P. Also if $\mathcal{F}$ is asked by $\mathcal{U}$ to simulate a session where P^* receives m, c^* from party Q with $m \neq m^*$ or $Q \neq P$, then $\mathcal{F}$ asks $\mathcal{D}$ for the MAC tag in order to continue the simulation.

4. If bad does not happen in session s then $\mathcal{F}$ halts with no output and then $\mathcal{D}$ returns a random bit at its guess for b'. With probability at least ϵ/l, event bad does happen in session s. Since party P^* was never activated in this session, $\mathcal{U}$ will directly activate party P with message m^* to send the second (challenge) message to party P^*. Then $\mathcal{F}$ responds with m^*, c^*. When bad happens $\mathcal{U}$ will send a MAC value t for message m^* to party P which is then passed to $\mathcal{F}$ (who is simulating P). $\mathcal{F}$ then sends (m^*, t) to $\mathcal{D}$ as its forgery attempt. Note that $\mathcal{U}$ cannot ask for the MAC tag from P^* directly since P^* is not involved in the run when bad happens, so the forgery must originate from $\mathcal{U}$.

5. $\mathcal{U}$ receives the putative forgery (m^*, t) from $\mathcal{F}$ and checks its correctness using key k'. If $\mathsf{MacVer}_{k'}(m^*, t) = 1$ then $\mathcal{U}$ decides that encapsulation c contains key k' and returns its guess $b' = 0$ to $\mathcal{C}$. Otherwise $\mathcal{U}$ returns $b' = 1$.

Analysis We consider two cases, each of which occurs with probability 1/2.

- If $b = 0$ then $\mathcal{F}$ will always return a correct MAC tag for key k' when bad happens. In this case $\mathcal{D}$ will return $b' = 0$ and so will win the CCA game with $\mathcal{C}$.
- If $b = 1$ then we have two possibilities.
  1. If $\mathcal{F}$ returns an incorrect MAC tag for key k' when bad happens then $\mathcal{D}$ will return $b' = 1$ and so will again win its CCA game with $\mathcal{C}$.
  2. If $\mathcal{F}$ returns a correct MAC tag for key k' then $\mathcal{D}$ will return $b' = 0$ and so will lose the CCA game with $\mathcal{C}$. However, $\mathcal{F}$ got no help in its forgery game and so can win the $\mathsf{EUF\text{-}CMA}$ game against the MAC.

First note that

$$\Pr[\mathcal{F} \text{ wins}] \quad = \quad \Pr[\mathcal{F} \text{ wins}|\mathsf{bad}] \cdot \Pr[\mathsf{bad}] = \Pr[\mathcal{F} \text{ wins}|\mathsf{bad}] \cdot \frac{\epsilon}{l}$$

since $\Pr[\mathcal{F} \text{ wins}|\overline{\mathsf{bad}}] = 0$. Also

$$\Pr[\mathcal{D} \text{ wins}|\mathsf{bad}] \quad = \quad \frac{1}{2} + \frac{1}{2} \cdot (1 - \Pr[\mathcal{F} \text{ wins}|\mathsf{bad}]) = 1 - \Pr[\mathcal{F} \text{ wins}] \cdot \frac{l}{2\epsilon}.$$

Putting these together we get:

$$\Pr[\mathcal{D} \text{ wins}] \quad \geq \quad \left(1 - \frac{l}{2\epsilon} \Pr[\mathcal{F} \text{ wins}]\right) \cdot \frac{\epsilon}{l} + \frac{1}{2}\left(1 - \frac{\epsilon}{l}\right) = \frac{1}{2} + \frac{\epsilon}{2l} - \frac{1}{2} \Pr[\mathcal{F} \text{ wins}].$$

Finally we re-arrange to obtain: $2 \cdot \left(\Pr[\mathcal{D} \text{ wins}] - \frac{1}{2}\right) + \Pr[\mathcal{F} \text{ wins}] \geq \frac{\epsilon}{l}$, and then by Def. 2 and 4, $\epsilon \leq l \cdot \left(\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{D}) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{F})\right)$.

□

Now that λ_{KEM} is proven to be an MT-authenticator we can invoke Thm. 2 to conclude that λ_{KEM} can be used to authenticate messages in an SK-secure AM protocol and results in a SK-secure UM protocol. In order to optimise the resulting protocol we will want to use a compressed version of the authenticator (see Sec. 2.5) as shown in Fig. 5.

Compressed KEM-based MT-authenticator

Alice		**Bob**
	$\xleftarrow{\quad c \quad}$	$(c, k) \leftarrow \mathsf{Encap}(pk_{\mathsf{A}})$
$k \leftarrow \mathsf{Decap}(sk_{\mathsf{A}}, c)$	$\xrightarrow{\quad m, \tau = \mathsf{MAC}_k(m, \mathsf{B}) \quad}$	If $\mathsf{MacVer}_k(\tau, m, B) = 1$
"Alice sent m to Bob"		"Bob received m from Alice"

Figure 5: λ_{KEM}, the compressed KEM-based MT-authenticator.

The security proof for the compressed authenticator is identical to the proof for the full authenticator since the only difference is the deletion of plaintext messages in the UM which are ignored in the security proof.

Corollary 1. *Theorem 3 still holds if the authenticator in Fig. 3 is replaced by the compressed* KEM*-based* MT*-authenticator,* λ_{KEM}*, in Fig. 5.*

3.2 KEM-based AM protocol

In Fig. 6 we present a KEM-based protocol Π that is SK-secure in the AM. The protocol is a generalisation of the basic Diffie–Hellman AM protocol of CK01 [CK01]. We assume that a setup with parameters for the KEM is known already to all parties. The initiator A will be invoked by the $\mathsf{NewSession}(\mathsf{A}, \mathsf{B}, s, r)$ query and responds with a new ephemeral KEM public key pk_e. Upon receipt of (pk_e, s) the responder encapsulates a new session key sk in c, and returns it to party A.

KEM-based AM-protocol

Alice		**Bob**
$(pk_e, sk_e) \leftarrow \mathsf{Gen}()$	$\xrightarrow{\quad pk_e, s \quad}$	$(c, sk) \leftarrow \mathsf{Encap}(pk_e)$
$sk' \leftarrow \mathsf{Decap}(sk_e, c)$	$\xleftarrow{\quad c, s \quad}$	

Figure 6: KEM-based protocol with any CPA-secure KEM (see Def. 2).

Theorem 4. *Let* $\mathcal{A}$ *be an adversary against the* SK*-security of protocol* Π *shown in Fig. 6. Let* $\mathcal{A}$ *interact with at most* q *sessions of* Π *for each pair of parties. Let* n *be the*

maximum number of parties involved in the protocol run. Then the advantage of $\mathcal{A}$ can be bounded by: $\mathsf{Adv}^{\mathsf{SK}}_{\Pi}(\mathcal{A}) \leq n^2 q \cdot \mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{KEM}}(\mathcal{B})$.

Proof. The definition of SK-security has two requirements. The first requirement is achieved by the correctness of the KEM. For the second requirement we use adversary $\mathcal{A}$ against the *sk*-security of the protocol Π to construct an adversary $\mathcal{B}$ against the CPA-security of the KEM. Adversary $\mathcal{B}$ interacts with its challenger $\mathcal{C}$ in the $\mathsf{G}^{\mathsf{CPA}}_{\mathsf{KEM}}(\cdot)$ experiment.

First $\mathcal{B}$ is given (pk^*, c^*, k^*) by $\mathcal{C}$ and should output $b \in \{0,1\}$ guessing if k^* is the real key ($b = 1$) or a random key ($b = 0$). Next $\mathcal{A}$ chooses random parties A^* and B^* from the set of all (n possible) parties and for these chooses a *target session* identifier s^* by choosing a unique value in $[1..q]$, where q is the maximum number of sessions at any party. Then $\mathcal{B}$ invokes $\mathcal{A}$ and simulates all responses as follows.

- $\mathsf{NewSession}(\mathsf{A}, \mathsf{B}, r, s)$: if r is initiator then $\mathcal{B}$ checks whether $\mathsf{A} = \mathsf{A}^*$, $\mathsf{B} = \mathsf{B}^*$, and $s = s^*$ and if so $\mathcal{B}$ returns (pk^*, s^*) to $\mathcal{A}$. Otherwise $\mathcal{B}$ runs $(pk_e, sk_e) \leftarrow \mathsf{Gen}()$, returns (pk_e, s) to $\mathcal{A}$ and stores $(s, \mathsf{A}, \mathsf{B}, sk_e)$ for answering later queries about session s.

- $\mathsf{Send}(\mathsf{A}, \mathsf{B}, m \parallel s)$: if A is the responder in the run with session identifier s then m is a public encapsulation key pk_e which can be used by $\mathcal{B}$ to compute the correct response. If A is the initiator in the run with session identifier s then $\mathcal{B}$ will accept. Note that if $\mathsf{A} = \mathsf{A}^*$, $\mathsf{B} = \mathsf{B}^*$ and $s = s^*$ then $\mathcal{B}$ does not have the decapsulation key and cannot compute the session key.

- $\mathsf{Corrupt}(\mathsf{A})$: if $\mathsf{A} \in \{\mathsf{A}^*, \mathsf{B}^*\}$ then $\mathcal{B}$ aborts and returns a random bit to $\mathcal{C}$. Otherwise $\mathcal{B}$ returns any session key and sk_e value allocated to A. Note that for this protocol there are no long-term keys.

- $\mathsf{RevealKey}(\mathsf{A}, \mathsf{B}, s)$: if $\mathsf{A} \in \{\mathsf{A}^*, \mathsf{B}^*\}$ and $s = s^*$ then $\mathcal{B}$ aborts and returns a random bit to $\mathcal{C}$. Otherwise $\mathcal{B}$ returns the session key value allocated to A with session identifier s (or $\emptyset$ if it is undefined).

- $\mathsf{RevealState}(\mathsf{A}, \mathsf{B}, s)$: if $\mathsf{A} = \mathsf{A}^*$, $\mathsf{B} = \mathsf{B}^*$ and $s = s^*$ then $\mathcal{B}$ aborts and returns a random bit to $\mathcal{C}$. Otherwise, if A is the initiator in the run with session identifier s then $\mathcal{B}$ returns the private ephemeral key, sk_e, or $\emptyset$ if it is undefined.

- $\mathsf{Expire}(\mathsf{A}, \mathsf{B}, s)$: if there is a completed session s at A with B then $\mathcal{B}$ will return a success flag to $\mathcal{A}$ or otherwise return a failure flag to $\mathcal{A}$.

- $\mathsf{Test}(\mathsf{A}, \mathsf{B}, s)$: if $\mathsf{A} = \mathsf{A}^*$, $\mathsf{B} = \mathsf{B}^*$ and $s = s^*$ then $\mathcal{B}$ returns k^* to $\mathcal{A}$. Otherwise $\mathcal{B}$ aborts and returns a random guess for b.

As long as $\mathcal{B}$ does not abort then $\mathcal{A}$ will eventually halt and output its bit b'. Then $\mathcal{B}$ sets $b \leftarrow b'$ and returns b to its challenger $\mathcal{C}$.

If $\mathcal{A}$ chooses the target session as its test session then $\mathcal{B}$ wins whenever $\mathcal{A}$ wins. This happens with probability at least $1/n^2q$. Note that this necessarily means that $\mathcal{B}$ does not abort on any query. Then we have

$$\begin{aligned} \Pr[\mathcal{B} \text{ wins}] &\geq \frac{1}{2} \cdot \left(1 - \frac{1}{n^2 q}\right) + \Pr[\mathcal{A} \text{ wins}] \cdot \frac{1}{n^2 q} \\ \text{or } \Pr[\mathcal{B} \text{ wins}] - \frac{1}{2} &\geq \frac{1}{n^2 q} \left(\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}\right). \end{aligned}$$

Thus the theorem statement follows.

□

4 Generic KEM-based AKE protocols

With the building blocks from Sec. 3 we now apply MT-authenticators to AM protocols and optimise them to obtain protocols which are both SK-secure in the realistic UM security model and efficient in comparison with other protocols in the literature. There is no restriction to apply the new MT-authenticator in Fig. 5 only to the new AM protocol in Fig. 6; the authenticator can be applied to any SK-secure AM protocol and any authenticator can be applied to the KEM-based AM protocol. Furthermore, we may apply different MT-authenticators to each of the messages in an AM protocol [HBN06, Thm. 6] resulting in yet more ways to construct different secure protocols.

Due to our field's focus on post-quantum security in recent years, we emphasise KEM-based and signature-based components in this section, allowing us to apply any of the primitives from the NIST competition library. We illustrate this usage with several different examples in this section, applying both our new KEM-based authenticator and the existing signature-based authenticator to achieve a variety of protocols. Another example, also with potential for post-quantum security, is to apply the MAC-based authenticator of CK01 to our KEM-based AM protocol. This results in a protocol suitable for pre-shared key environments which is a common scenario, for example in TLS and IPSec. Details of a MAC-based generic protocol construction are available in Appendix 4.3.

4.1 Compiled KEM-based protocol and optimization

We start with the AM-secure protocol from Fig. 6 and then apply the compiler consisting of application of the compressed MT-authenticator to each of its two messages. This leads to the 4-message protocol of Fig. 7.

Messages 1 and 2 are the result of applying the compressed MT-authenticator to authenticate the ephemeral public key pk_e generated by Alice. Messages 3 and 4 are the result of applying the compressed MT-authenticator to authenticate the encapsulated shared key c^* generated by Bob. The difference between m_1 and m_1' (resp. m_2 and m_2') in Fig. 7 is that both players have their own version of s — the MAC verifies the integrity of both the message and the session.

Generic authenticated KEM-based protocol (unoptimised)

Alice		**Bob**
$(pk_e, sk_e) \leftarrow \mathsf{Gen}()$	$\xleftarrow{c_1}$	$(c_1, k_1) \leftarrow \mathsf{Encap}(pk_\mathsf{A})$
$k_1' \leftarrow \mathsf{Decap}(sk_\mathsf{A}, c_1)$ $m_1 \leftarrow (pk_e \parallel s)$		
$\tau_1 \leftarrow \mathsf{Mac}_{k_1'}(m_1 \parallel \mathsf{B})$	$\xrightarrow{m_1, \tau_1}$	$m_1' \leftarrow (pk_e \parallel s)$ If $\mathsf{MacVer}_{k_1}(\tau_1, m_1' \parallel \mathsf{B}) = 0$ **Abort**
$(c_2, k_2) \leftarrow \mathsf{Encap}(pk_\mathsf{B})$	$\xrightarrow{c_2}$	$k_2' \leftarrow \mathsf{Decap}(sk_\mathsf{B}, c_2)$ $(c^*, k^*) \leftarrow \mathsf{Encap}(pk_e)$ $m_2 \leftarrow (c^* \parallel s)$
$m_2' \leftarrow (c^* \parallel s)$	$\xleftarrow{m_2, \tau_2}$	$\tau_2 \leftarrow \mathsf{Mac}_{k_2'}(m_2 \parallel \mathsf{A})$
If $\mathsf{MacVer}_{k_2}(\tau_2, m_2' \parallel \mathsf{A}) = 0$ **Abort** $k^* \leftarrow \mathsf{Decap}(sk_e, c^*)$		

Figure 7: Generic 4-message protocol obtained by compiling the KEM-based AM protocol with the compressed KEM-based MT-authenticator.

To optimise the 4-message protocol in Fig. 7 we take four simple steps:

1. The messages that were numbered 2 and 3 will be sent in parallel as a new message with number 2. This does not change the order or contents of any messages.
2. The session identifier, s, will be constructed by the parties as part of the protocol, instead of taking it as an external input to the protocol. Recall that the only requirements on s are to be unique between the parties amongst any incomplete protocol session between the two parties. We choose $s = c_1 \parallel c_2$ where c_1 and c_2 are the (randomised) encapsulations (ciphertexts) generated by each party.
3. Repeated message fields and fields previously generated by message receivers are removed from messages.
4. The protocol parties are re-labelled so that Alice becomes the protocol initiator.

Optimised KEM-based UM protocol

Alice		**Bob**
$(c_1, k_1) \leftarrow \mathsf{Encap}(pk_\mathsf{B})$	$\xrightarrow{c_1}$	$(pk_e, sk_e) \leftarrow \mathsf{Gen}()$
		$(c_2, k_2) \leftarrow \mathsf{Encap}(pk_\mathsf{A})$
		$s \leftarrow c_1 \parallel c_2$
		$k'_1 \leftarrow \mathsf{Decap}(sk_\mathsf{B}, c_1)$
$s \leftarrow c_1 \parallel c_2$	$\xleftarrow{pk_e, \tau_1, c_2}$	$\tau_1 \leftarrow \mathsf{Mac}_{k'_1}(pk_e \parallel s \parallel \mathsf{A})$
If $\mathsf{MacVer}_{k_1}(\tau_1, pk_e \parallel s \parallel \mathsf{A}) = 0$		
Abort		
$k'_2 \leftarrow \mathsf{Decap}(sk_\mathsf{A}, c_2)$		
$(c^*, k^*) \leftarrow \mathsf{Encap}(pk_e)$		
$\tau_2 \leftarrow \mathsf{Mac}_{k'_2}(c^* \parallel s \parallel \mathsf{B})$	$\xrightarrow{c^*, \tau_2}$	If $\mathsf{MacVer}_{k_2}(\tau_2, c^* \parallel s \parallel \mathsf{B}) = 0$
		Abort
		$k^* \leftarrow \mathsf{Decap}(sk_e, c^*)$

Figure 8: Optimised UM protocol from the KEM-based AM protocol and the KEM-based MT-authenticator.

Combining all of these steps we obtain the optimised protocol shown in Fig. 8.

As far as we are aware, the precise protocol of Fig. 8 is new in the literature. There are several existing protocols also aimed at achieving AKE based only on KEMs [FSXY12, SSW20, HNS+20] or encryption [dSW17]. Several of these are motivated by the desire to avoid signatures, which tend to suffer efficiency disadvantages compared with KEMs in the post-quantum examples from the NIST competition. The security varies between of each these protocols. For example, the FSXY protocol [FSXY12] provides security against ephemeral key leakage whereas KEMTLS [SSW20], like the protocol of Fig. 8, lacks this property. On the other hand, our protocol does allow state reveals from non-target sessions. KEMTLS is also designed to provide only one-way (server) authentication. Making a judgement on which of these protocols is "better" is therefore difficult since it depends on the security requirements and implementation details. In Sec. 5 we compare efficiency using concrete KEMs and signature schemes to get a better feel for the relative efficiencies.

4.2 Generic protocols using signatures

We now look at two further generic protocols which we can obtain by using signatures in combination with our KEM-based AM protocol. We will need to apply the compressed

signature-based authenticator shown in Fig. 9.

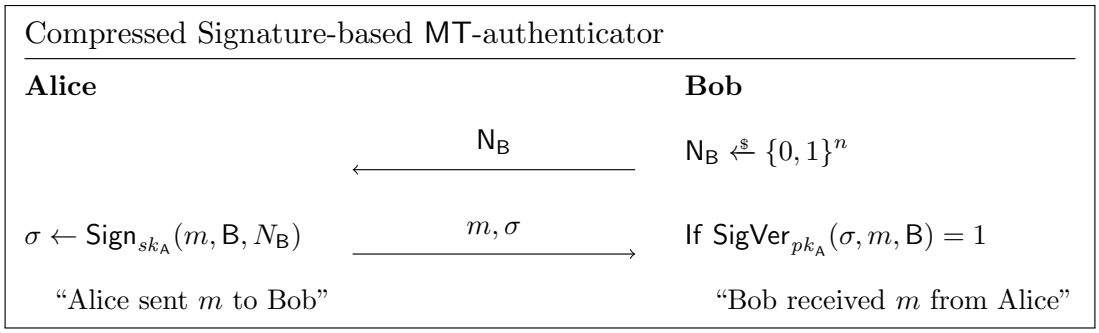

Figure 9: λ_{Sign}, a compressed signature-based MT-authenticator.

The authenticator λ_{Sign} is derived from the authenticator of BCK98 by removing the unnecessary message components in exactly the same way as for the encryption- and KEM-based authenticators. As before, the existing proof that the full authenticator is a valid MT-authenticator [BCK98] still holds.

The optimised protocol for the KEM-based AM protocol compiled with the signature-based authenticator is shown in Fig. 10. The optimisation follows the same process as described in Sec. 4.1. Although more general, the resulting protocol has much in common with the signed Diffie–Hellman protocol which has been widely known and deployed for many years and is today the usual AKE in the latest version of TLS (though with only one-sided authentication).

We have another way to authenticate the KEM-based AM protocol, namely to authenticate its two messages with different MT-authenticators. As far as we are aware there are no examples of such a protocol in the existing literature. There can be practical usages, for example when signatures are very expensive to generate but very cheap to verify. In such a case, when a powerful server authenticates its AM message it can shift computation away from a lightweight client by using the signature-based authenticator, while the client can avoid generating signatures by using a different KEM-based authenticator. In Fig. 14 we show the optimised protocol using the KEM-based authenticator for the first message and the signature-based authenticator for the second message. A mirror protocol results from using the two authenticators the other way around. For completeness this optimised protocol is given as Fig. 11.

4.3 MAC-based MT-authenticator

Canetti and Krawczyk [CK01] present also a MAC-based MT-authenticator (interestingly described only in compressed form) as shown in Fig. 12. This authenticator can be useful in scenarios where pre-shared keys exist such as in many use-cases of TLS with lightweight entities and also in session resumption in the latest TLS 1.3 version.

Since MACs are expected to remain secure in the post-quantum setting it makes

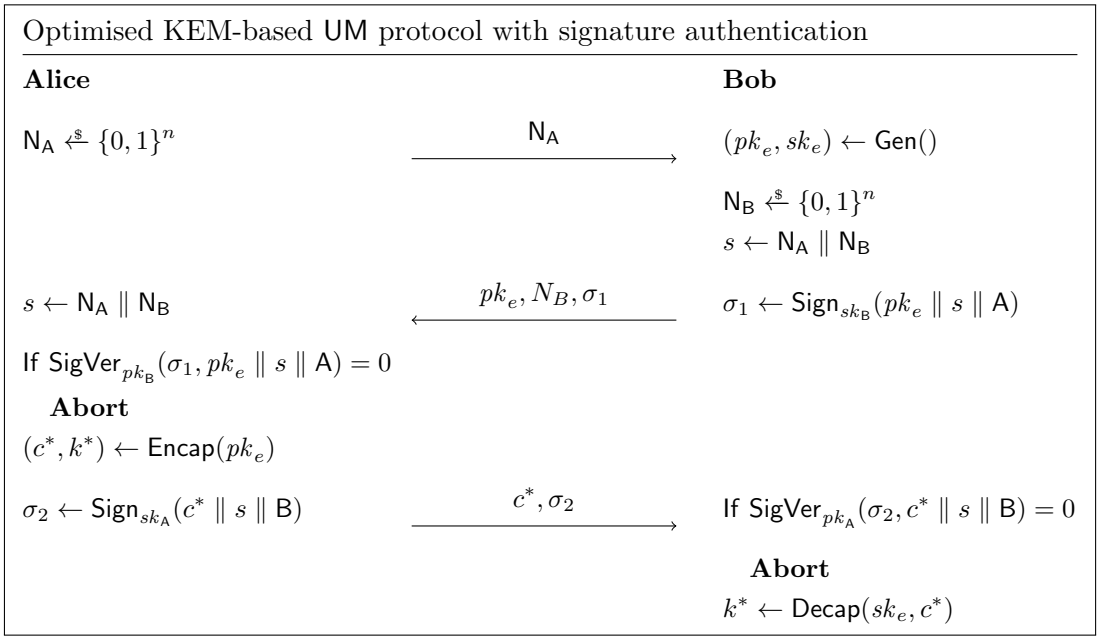

Figure 10: Optimised UM protocol from the KEM-based AM protocol and the signature-based MT-authenticator.

sense to combine this authenticator with our KEM-based AM protocol to obtain a post-quantum secure AKE protocol suitable for pre-shared key applications. Fig. 13 shows the resulting optimised protocol.

5 Concrete post-quantum secure AKE protocols

In the previous section we have presented optimised generic AKE protocols which will be secure as long as the KEM, signature and MAC primitives are instantiated with secure instances. Even restricting to a handful of currently best-trusted post-quantum primitives, this leads to hundreds of potential concrete protocols, bearing in mind that we have shown that different KEMs and signatures can be mixed in the same protocol and observing that the generic protocols are not symmetric between initiator and responder. The question of whether the concrete instantiated protocols are practical in terms of computational efficiency and message size is a natural one.

5.1 Comparison of different implementations

Tables 1a and 1b present the computational efficiency of various KEMs and signatures from the NIST competition. The figures are taken from two recent reports from ETSI [Sec21a, Sec21b]. They are not intended as definitive efficiency comparisons — indeed some of the figures have already been improved upon — but rather to illustrate

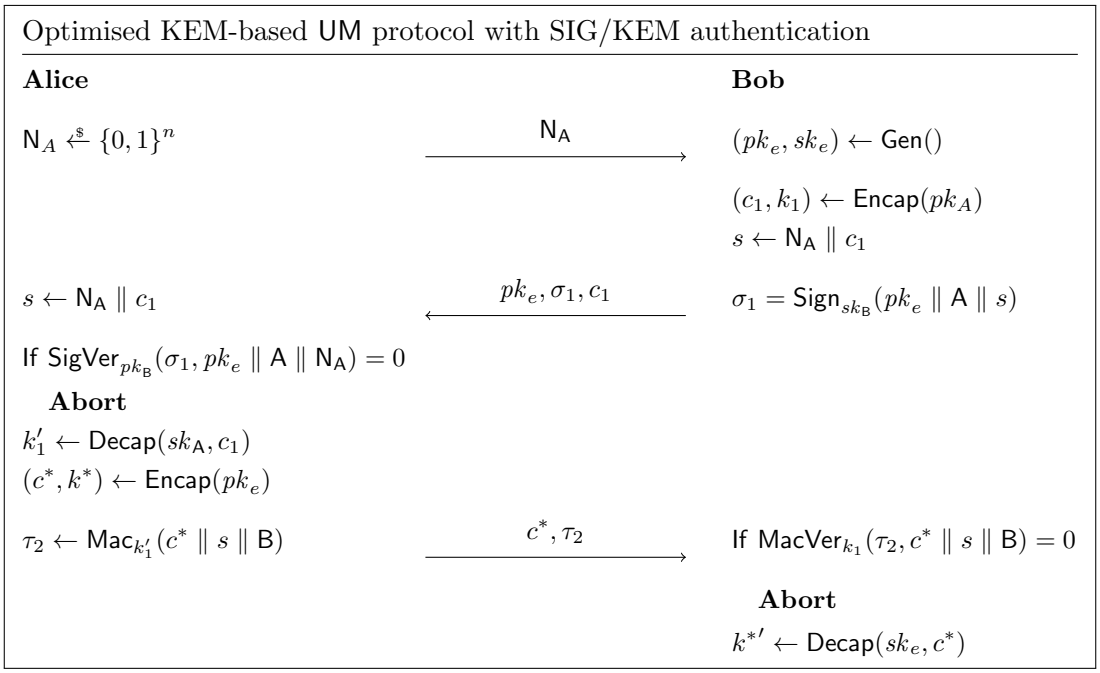

Figure 11: Optimised UM protocol from the KEM-based AM protocol and the signature-based MT-authenticator used for authenticating the first message, and the kem-based MT-authenticator authenticating the second message.

typical ballpark figures and highlight the big variation between many of the existing proposals.

5.2 Computational cost

To give an impression of the computational costs of our new protocols we summarize the number of public key operations needed in each of our optimised protocols in the upper part of Table 2. The lower part of the same table includes the number of similar operations for some prominent existing protocols.

All of the protocols in Table 2 use three passes and three rounds. However, they do not all have the same goals or assumptions. TLS and KEMTLS only aim for server-side authentication while our protocol in Fig. 13 assumes pre-shared keys. PQ-WireGuard [HNS+20] is a variant of the WireGuard protocol using only KEMs. Its design is based on the FSXY protocol [FSXY12]. All of the protocols in the lower half of Table 2 use only KEMs, both for authentication and key exchange. When comparing with our KEM-only protocol of Fig. 8 we see that the main computational effort is the same as in the three bottom protocols which are all KEM-only protocols. We conclude that our protocols are comparable in computation to existing ones. Another difference between the various protocols is on which side most computations are performed, e.g.

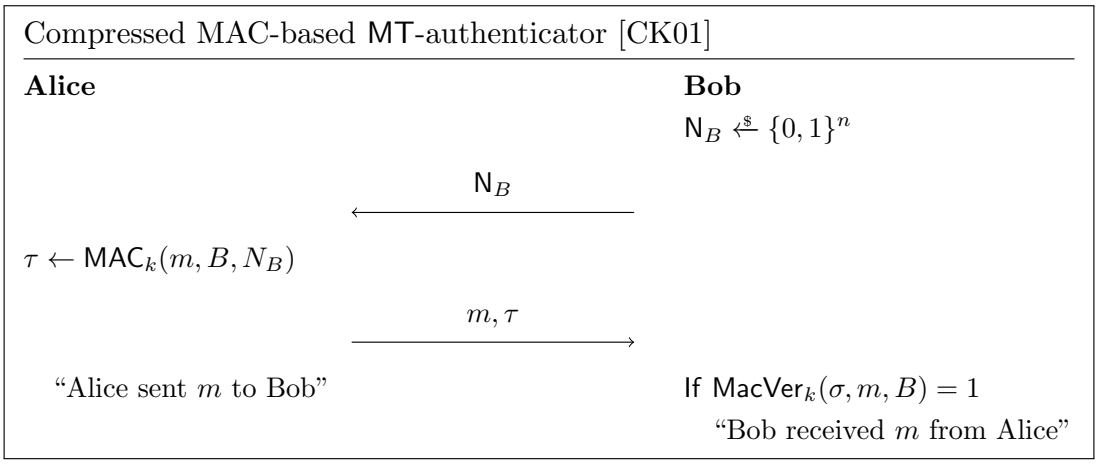

Figure 12: λ_{MAC}, a compressed MAC-based MT-authenticator with shared key k.

in Fig. 14 the initiator Alice encapsulates twice while Bob performs computationally heavier decapsulations and generation of the ephemeral key.

The most obvious difference between the upper and lower part of the table is that our designs have the responder generating the ephemeral KEM key while all existing protocols shown give this task to the initiating party. We do not believe that either option is inherently better, rather it depends on the relative costs of generation, encapsulation and decapsulation of the instantiating ephemeral KEM. For some well-known KEMs (Table 1a), key generation is far more costly than encapsulation or decapsulation. To minimise the overall protocol cost to both parties it seems better to use an algorithm with more uniform cost for the three KEM operations, but if it is desired to reduce the cost of one party at the expense of the other then different algorithms can be better.

It can be argued that implementation is most efficient when the same concrete KEM is used for all three of the KEM instances in the all-KEM protocols. This should be true at least with regard to the codebase needed in any implementation. However, this may not be the case when it comes to counting computation cycles. Recall that the AM protocol includes generation of an ephemeral public key, while the long-term keys are generated only once before the protocol runs. Therefore it can make sense to use a KEM with an efficient key generation algorithm for the ephemeral KEM, and a different one with a much less efficient key generation algorithm for the KEM using the long-term keys. PQ-WireGuard [HNS+20] does exactly this, using Classic McEliece for the long-term KEM and a variant of Saber for the ephemeral KEM. The size of its public key (Table 1a) shows why using Classic McEliece for the ephemeral KEM seems to be a bad idea.

Current known post-quantum signatures tend to be computationally less efficient than KEM constructions (Table 1b) where signing is much more expensive than decapsulation in known algorithms. It is therefore natural that KEM-based authentication

Optimised KEM-based UM protocol with MAC authentication using pre-shared MAC key k

Alice		**Bob**
$\mathsf{N}_A \xleftarrow{\$} \{0,1\}^n$	$\xrightarrow{\mathsf{N}_A}$	$(pk_e, sk_e) \leftarrow \mathsf{Gen}()$
		$\mathsf{N}_B \xleftarrow{\$} \{0,1\}^n$ $s \leftarrow \mathsf{N}_A \parallel \mathsf{N}_B$
$s \leftarrow \mathsf{N}_A \parallel \mathsf{N}_B$	$\xleftarrow{pk_e, N_B, \tau_1}$	$\tau_1 \leftarrow \mathsf{MAC}_k(pk_e \parallel s \parallel A)$
If $\mathsf{MacVer}_k(\tau_1, pk_e \parallel s \parallel A) = 0$ **Abort**		
$(c^*, k^*) \leftarrow \mathsf{Encap}(pk_e)$		
$\tau_2 \leftarrow \mathsf{MAC}_k(c^* \parallel s \parallel B)$	$\xrightarrow{c^*, \tau_2}$	If $\mathsf{MacVer}_k(\tau_2, c^* \parallel s \parallel B) = 0$ **Abort**
		$k^* \leftarrow \mathsf{Decap}(sk_e, c^*)$

Figure 13: Optimised UM protocol from the KEM-based AM protocol and the MAC-based MT-authenticator.

currently is seen favourably. This can change in the future, and the NIST focus on new post-quantum signature proposals may well lead to more efficient post-quantum secure signature algorithms. To our knowledge, there is no analog to our KEM/Sig or Sig/KEM protocols in the literature, neither are we aware of post-quantum proposals for the pre-shared key case.

Remark. We find it interesting to remark on a major difference regarding the *symmetry* of the computation requirements between Diffie–Hellman and the AM protocol (Fig. 6) which can be regarded as a generalization. The computational requirements for Diffie–Hellman are the same for both initiator and responder. In the AM protocol the initiator runs Gen and Decap while the responder runs only Encap. Of itself this is not significant, since Encap really has two purposes: to generate the new key for the responder and to generate the encapsulation for the initiator. Thus in the Diffie–Hellman case the cost of Encap is the same as the cost of Gen plus the cost of Decap. However, in all the examples in Table 1a this is nowhere close to being true. Indeed Encap is always significantly cheaper than Gen plus Decap, which may be important when deciding which party take the role of initiator in a protocol run.

5.3 Communications cost

In Table 3 we take an inventory of the message fields in each of our abstract protocols. Due to the optimisation techniques explained earlier, the number of fields sent and received by each party is three in all cases. Informally, at least, this is a minimum since

Optimised KEM-based UM protocol with KEM/SIG authentication

Alice		**Bob**
$(c_1, k_1) \leftarrow \mathsf{Encap}(pk_B)$	$\xrightarrow{c_1}$	$(pk_e, sk_e) \leftarrow \mathsf{Gen}()$
		$\mathsf{N}_B \xleftarrow{\$} \{0,1\}^n$
		$s \leftarrow c_1 \parallel \mathsf{N}_B$
		$k_1 \leftarrow \mathsf{Decap}(sk_\mathsf{B}, c_1)$
$s \leftarrow c_1 \parallel \mathsf{N_B}$	$\xleftarrow{pk_e, N_B, \tau_1}$	$\tau_1 \leftarrow \mathsf{Mac}_{k_1}(pk_e \parallel s \parallel A)$
If $\mathsf{MacVer}_{k_1}(\tau_1, pk_e \parallel s \parallel A) = 0$		
Abort		
$(c^*, k^*) \leftarrow \mathsf{Encap}(pk_e)$		
$\sigma_2 \leftarrow \mathsf{Sign}_{sk_\mathsf{A}}(c^* \parallel s \parallel \mathsf{B})$	$\xrightarrow{c^*, \sigma_2}$	If $\mathsf{SigVer}_{pk_A}(\sigma_2, c^* \parallel s \parallel \mathsf{B}) = 0$
		Abort
		$k^* \leftarrow \mathsf{Decap}(sk_e, c^*)$

Figure 14: Optimised UM protocol from the KEM-based AM protocol and the KEM-based MT-authenticator used for the first message, and the signature-based MT-authenticator for the second message.

the ephemeral public key needs to communicated and then used in the response, and each of these two messages must be authenticated using a fresh value chosen by the other party.

The size of these fields depends on the parameters of the concrete primitives chosen. In July 2022, NIST announced a first list of selected candidates as a result of its Post-Quantum Cryptography competition [AAC+22], pointing out CRYSTALS-Kyber as their selected KEM and CRYSTALS-Dilithium as their selected signature algorithm. Using the real-world efficiency of the Kyber KEM and the Dilithium signature scheme, in Tables 1a and 1b and naively adding up these numbers, all messages in our Fig. 14 protocol would be under 5 kB for Kyber-1024, which definitely is practical. Another look at the ephemeral public key sizes in Table 1a shows why the choice of Saber in PQ-WireGuard [HNS+20] is an obvious one. We note that before its recent demise, SIKE looked an even more promising candidate to minimise the ephemeral key size.

Just as for computation efficiency, currently accepted post-quantum secure signature candidates do not look attractive for communications efficiency as shown in Table 1b. To minimise signature size FALCON is a better choice than Dilithium, but requires a trade-off with computation.

We reiterate that Table 3 assumes that authentic long-term public keys are available to all parties by some external channel. This fits some real-world protocols (such as

Table 1: The efficiency of selected post-quantum algorithm proposals. Algorithms Gen, Encap, Decap, Sign and SigVer, are measured in clock cycles on a standard processor. Parameters public key size (pk), ciphertexts (encapsulations) (ct) and signatures (σ) are measured in bytes.

(a) The efficiency of various KEMs [Sec21a].

	Gen	Encap	Decap	pk	ct	NIST security category
mceliece348864	36641040	44 350	134 745	261120	128	1
mceliece460896	117067765	117 782	271 694	524160	188	3
KYBER512	33856	45 200	59 088	800	768	1
KYBER1024	73544	97 324	115 332	1568	1568	3
ntruhps2048677	309216	83 519	59 729	930	930	1
ntruhps4096821	431667	98 809	75 384	1230	1230	3
LightSaber	45152	49 948	47 852	672	736	1
Saber	66727	79 064	76 612	992	1088	3

(b) The efficiency of various signature schemes [Sec21b].

	Sign	SigVer	σ	NIST security category
Dilithium-3	269 000	118 000	3293	3
Dilithium-5	429 000	179 000	4595	5
FALCON-512	386 678	82 340	666	1
FALCON-1025	789 564	168 498	1280	5

WireGuard) but not others (such as TLS). Post-quantum signatures used to certify the long-term public keys can be chosen independently of other concrete choices in the protocol. This choice will obviously affect both the computation for each party and the size of the protocol messages. Although registration of public keys can avoid use of post-quantum signatures [GHL+22], it seems necessary to use signatures to achieve usual certificate properties.

6 Conclusion and future work

As summarized in Sec. 1.2, the main contributions of this work are the new KEM-based authenticator and corresponding security proof, the new proofs in the AM-model and the derivation of several new generic AKE protocols. We hope that the flexibility of protocol designs can be useful in fitting AKE protocols to different application use cases and that as new concrete KEMs and signature schemes are developed the generic protocols will yield new and interesting instantiations. Some of the ways to extend the work are the

Table 2: The number of public key operations for ours and existing protocols.

	Initiator					Responder				
	Gen_e	Encap	Decap	Sign	SigVer	Gen_e	Encap	Decap	Sign	SigVer
Fig 8. KEM/KEM	0	2	1	0	0	1	1	2	0	0
Fig 10. Sig/Sig	0	1	0	1	1	1	1	0	1	1
Fig 14. KEM/Sig	0	2	0	1	0	1	0	2	0	1
Fig 11. Sig/KEM	0	1	1	0	1	1	1	1	1	0
Fig 13. MAC/MAC	0	1	0	0	0	1	0	1	0	0
TLS 1.3[3]	1	0	1	0	1	0	1	0	1	0
KEMTLS[4] [SSW20]	1	1	1	0	0	0	1	1	0	0
KEMTLS-pdk [SSW21]	1	1	2	0	0	0	2	1	0	0
PQ-WireGuard [HNS+20]	1	1	2	0	0	0	2	1	0	0
SSW17[5][dSW17]	1	1[6]	2	0	0	0	2	1	0	0

Table 3: What comprises the messages sent in each protocol.

	Message 1	Message 2	Message 3
Fig 8. KEM/KEM	ct	pk, ct, MAC	ct, MAC
Fig 10. Sig/Sig	N	pk, N, σ	ct, σ
Fig 14. KEM/Sig	ct	pk, N, MAC	pk, MAC
Fig 11. Sig/KEM	N	pk, σ, ct	ct, MAC
Fig 13. MAC/MAC	N	pk, N, MAC	ct, MAC

following.

- Adding additional security properties; for example, application of the twisted-PRF trick [FSXY12] can likely be added to an AM-protocol to secure against ephemeral leakage.
- Check whether use of hybid-KEMs (secure against conventional adversaries based on traditional assumptions and secure against post-quantum adversaries based on new assumptions) can be usefully applied to obtain hybrid-secure AKE [BBF+19].
- Since the modular approach does not naturally lead to tight reductions, it would be useful to improve on this, although in the end it may be necessary to complement new protocol designs obtained from the modular approach with a monolithic proof in a stronger model for some concrete protocol.

[3]Using Diffie-Hellman as an ephemeral KEM. Unilateral authentication
[4]Unilateral authentication
[5]Assuming that our KEM-based AM protocol is used as the base protocol.
[6]Encryption is needed in the full protocol, not encapsulation

- Applying an authenticator just to one message (from a two-message AM protocol) will allow for unilateral authentication such as is common in TLS. The security and efficiency of such a generic protocol deserves analysis.
- Real-world implementations of the generic protocols is also left to future work — obviously this would be a prerequisite for future adaptation, and give us a more concrete comparison of their efficiency in the real world.

References

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/publications/detail/nistir/8413/final`.

[AASA+20] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/publications/detail/nistir/8309/final`.

[ADH+22] Yawning Angel, Benjamin Dowling, Andreas Hülsing, Peter Schwabe, and Florian Weber. Post quantum noise. Cryptology ePrint Archive, Report 2022/539, 2022. `https://eprint.iacr.org/2022/539`.

[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`.

[BCD+16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1006–1018. ACM Press, October 2016. `doi:10.1145/2976749.2978425`.

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols

(extended abstract). In *30th ACM STOC*, pages 419–428. ACM Press, May 1998. doi:10.1145/276698.276854.

[BHMS17] Nina Bindel, Udyani Herath, Matthew McKague, and Douglas Stebila. Transitioning to a quantum-resistant public key infrastructure. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 384–405. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-59879-6_22.

[BJS15] Florian Bergsma, Tibor Jager, and Jörg Schwenk. One-round key exchange with strong security: An efficient and generic construction in the standard model. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 477–494. Springer, Heidelberg, March / April 2015. doi:10.1007/978-3-662-46447-2_21.

[BL17] Daniel J Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[CFS+21] Sofía Celi, Armando Faz-Hernández, Nick Sullivan, Goutam Tamvada, Luke Valenta, Thom Wiggers, Bas Westerbaan, and Christopher A. Wood. Implementing and measuring KEMTLS. Cryptology ePrint Archive, Report 2021/1019, 2021. https://eprint.iacr.org/2021/1019.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, Heidelberg, May 2001. doi:10.1007/3-540-44987-6_28.

[Cre11] Cas Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *ASIACCS 11*, pages 80–91. ACM Press, March 2011.

[DAL+17] Jintai Ding, Saed Alsayigh, Jean Lancrenon, Saraswathy RV, and Michael Snook. Provably secure password authenticated key exchange based on RLWE for the post-quantum world. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 183–204. Springer, Heidelberg, February 2017. doi:10.1007/978-3-319-52153-4_11.

[dSW17] Cyprien Delpech de Saint Guilhem, Nigel P. Smart, and Bogdan Warinschi. Generic forward-secure key agreement without signatures. In Phong Q. Nguyen and Jianying Zhou, editors, *ISC 2017*, volume 10599 of *LNCS*, pages 114–133. Springer, Heidelberg, November 2017.

[DXL12] Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology

ePrint Archive, Paper 2012/688, 2012. https://eprint.iacr.org/2012/688. URL: https://eprint.iacr.org/2012/688.

[FSXY12] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 467–484. Springer, Heidelberg, May 2012. doi:10.1007/978-3-642-30057-8_28.

[GHL+22] Tim Güneysu, Philip Hodges, Georg Land, Mike Ounsworth, Douglas Stebila, and Greg Zaverucha. Proof-of-possession for KEM certificates using verifiable generation. Cryptology ePrint Archive, Report 2022/703, 2022. https://eprint.iacr.org/2022/703.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 190–218. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5_7.

[HBN06] Yvonne Hitchcock, Colin Boyd, and Juan Manuel González Nieto. Modular proofs for key exchange: rigorous optimizations in the Canetti-Krawczyk model. *Appl. Algebra Eng. Commun. Comput.*, 16(6):405–438, 2006. doi:10.1007/s00200-005-0185-9.

[HNS+20] Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R. Zimmermann. Post-quantum WireGuard. Cryptology ePrint Archive, Report 2020/379, 2020. https://eprint.iacr.org/2020/379.

[JKRS21] Tibor Jager, Eike Kiltz, Doreen Riepel, and Sven Schäge. Tightly-secure authenticated key exchange, revisited. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 117–146. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77870-5_5.

[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.

[Kra05] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, Heidelberg, August 2005. doi:10.1007/11535218_33.

[Pei15] Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Paper 2015/939, 2015. https://eprint.iacr.org/2015/939. URL: https://eprint.iacr.org/2015/939.

[Sec21a] ETSI Technical Committee Cyber Security. Quantum-safe public-key encryption and key encapsulation. ETSI TR 103823, ETSI, October 2021. URL: https://www.etsi.org/deliver/etsi_tr/103800_103899/103823/01.01.01_60/tr_103823v010101p.pdf.

[Sec21b] ETSI Technical Committee Cyber Security. Quantum-safe signatures. ETSI TR 103616, ETSI, September 2021. URL: https://www.etsi.org/deliver/etsi_tr/103600_103699/103616/01.01.01_60/tr_103616v010101p.pdf.

[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.

[SSW21] Peter Schwabe, Douglas Stebila, and Thom Wiggers. More efficient post-quantum KEMTLS with pre-distributed public keys. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *ESORICS 2021, Part I*, volume 12972 of *LNCS*, pages 3–22. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-88418-5_1.